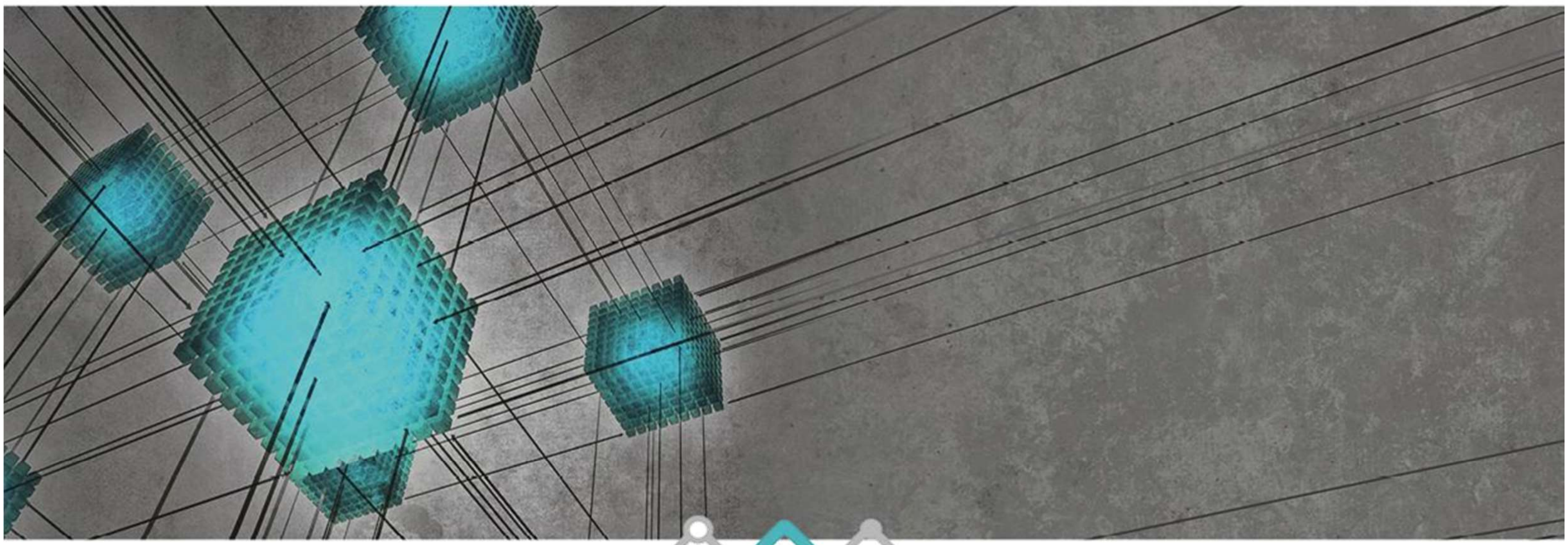


interiot

INTEROPERABILITY
OF HETEREOGENEUS
IOT PLATFORMS.

D6.2

Factory Aceptance Test Plan

Version: 1.0

February 2018

INTER-IoT

INTER-IoT aim is to design, implement and test a framework that will allow interoperability among different Internet of Things (IoT) platforms.

Most current existing IoT developments are based on “closed-loop” concepts, focusing on a specific purpose and being isolated from the rest of the world. Integration between heterogeneous elements is usually done at device or network level, and is just limited to data gathering. Our belief is that a multi-layered approach integrating different IoT devices, networks, platforms, services and applications will allow a global continuum of data, infrastructures and services that will enhance different IoT scenarios. Moreover, reuse and integration of existing and future IoT systems will be facilitated, creating a de facto global ecosystem of interoperable IoT platforms.

In the absence of global IoT standards, the INTER-IoT results will allow any company to design and develop new IoT devices or services, leveraging on the existing ecosystem, and bring them to market as fast as possible.

INTER-IoT has been financed by the Horizon 2020 initiative of the European Commission, contract 687283.

INTER-IoT

Factory Aceptance Test Plan

Version: V1.0

Security: Confidential

6, February 2018

The INTER-IoT project has been financed by the Horizon 2020 initiative of the European Commission, contract 687283

Executive Summary

This document describes the Factory Acceptance Test (FAT) plans of the INTER-IoT project.

During development units and components are tested and validated by the developers. This applies for the IoT framework components as well as for the project specific units and components. When all the components needed for the project are complete, tested and validated they are integrated into the system as defined for this project.

This system will then undergo this Factory Acceptance Test to test the readiness of the system. This is done in a LAB setup which approaches the actual field deployment as much as possible. When the FAT has been successfully executed and has been approved the system can advance to field integration and undergo the Site Acceptance Test (SAT).

This document describes all aspects of the FAT, form defining the versions of the used components and deliverable checklist up, test setup, tooling, test description, etc. to be able to test the readiness of the system under test. The document contain the working documents of each pilot, formatted to be included in a formal deliverable. The documents handled by each pilot and each third party wll be handled independently and separatedly.

It has to be considered that the FATs coming from the third parties have different structure as only a template and some guidance has been provided, in order to get creativity and different tests from each of the collaboration. This has led to a different size and content of the documents. Some homogeneization work has been performed, however following the strong expertise of NEWAYS in FAT processes and trying to provide a more industrial nature to the work in WP6, we have allowed some heterogeneity in the contributions.

D6.2 includes a section with the summary report of the mid term evaluation of the third parties of the open call evaluation has been satisfactory and some recommendations are issued

Finally, the structure of this document is divided into the following sections:

- Section A: Introduction
- Section B: Test strategy and approach
- Section I: FAT INTER-LogP
- Section II: FAT INTER-Health
- Section III: OpenCall FAT
- Section IV: Open Call Third Parties Evaluation
- Section V: Conclusions

List of Authors

Organisation	Authors	Main organisations' contributions
Neways	Dennis Engbers	Document structure, Executive summary, Introduction, Test setup and integration, Chapter templates
Neways	Johan Schabbink	Edit & reviews of final document
Neways	Arnoud Groote-Venema	Document review and preparation
VPF	Pablo Giménez, Joan Meseguer, Jordi Arjona	INTER-LogP pilots and tests
UPV-SABIEN	Gema Ibáñez	Test descriptions
E3tcity	Javier Escalera Casillas	Completed information with production process of the e3tcity company
XLAB d.o.o.	Flavio Fuart, Marija Gorenc Novak	Coordinating, editing, drafting.
AUEB	Nikos Fotiou, George C. Polyzos	Contributed test plans and document editing.
E3tcity	Javier Escalera Casillas	Completed information with production process of the e3tcity company
UPF	Toni Adame	System description, Deliverables and version overview, Requirements, scenario and Use cases to test, Test environment, Test description
UPF	Albert Bel	Ethics
TU Wien	Hong-Linh Truong	INTER-HINC Document structure, INTER-HINC test description
TU Wien	Bunjamin Memishi	INTER-HINC Requirements, Test Description
U.Twente	João Moreira	Document content.
Nemergent Solutions	Iñigo Ruiz, Jose Oscar Fajardo	Project specific descriptions.
Vrije Universiteit Brussel	Kris Steenhaut An Braeken	Documentation on OM2M framework, OM2M bridge and Sigfox tracers
AvailabilityPlus	Dr. Günther Hoffmann	Added details for SecurIoTy

/ SecurIoTy		
CNR-ITIA	Gianfranco Modoni	FAT addition
CNR-ITIA	Enrico Caldarola	FAT addition
Irideon	Bastian Faulhaber	Contribution specific content
CEA	Levent Gurgen, Jander Nascimento	Initial contribution on SensiNact platform description
INFOLYSiS	Harilaos Koumaras, Vaios Koumaras, Ch. Sakkas	Contributions related to SOFOS experiment.

Change control datasheet

Version	Changes	Chapters	Pages
0.1	Creation and completion	All	
0.2	Initial contributions	All	
0.3	Application of the review dated from 12nd December 2017 by Groote-Venema, Arnaud and Schabbink, Johan	All	
0.4	Complete Test Environment section	All	
1.0	INTER-IoT Review	All	
1.1	Final document	All	
0.1	Creation and completion	All	

Contents

Executive Summary 3
List of Authors 4
Change control datasheet 6
Contents 7
List of Figures 9
List of Tables 13
Acronyms 16
1 Introduction 17
2 Test strategy and approach 18
2.1 Testing strategy 18
2.2 Entrance criteria 18
2.2.1 Integration of the tested and validated system components 18
2.2.2 Validation and Test reports of the system components 19
2.2.3 FAT document 19
2.2.4 System test setup, Test applications and tooling 19
2.3 Acceptance Criteria 19
2.4 Testing types 19
2.5 Suspension and resumption criteria 21
2.6 Change Control Board 21
2.7 Defect Reporting 21
3 Factory Acceptance Tests 22
3.1 INTER-LogP FAT 23
3.1.1 System description 23
3.1.2 Use case oriented pilots 27
3.1.3 Deliverables and version overview 30
3.1.4 Requirements and scenarios 31
3.1.5 Test environment 31
3.1.6 Test description 35
3.1.7 Test outcome overview 36
3.1.8 Integration ethics and security 37
3.2 INTER-Health FAT 38
3.2.1 System description 38
3.2.2 Integration of IoT framework 40
3.2.3 Deliverables and version overview 41
3.2.4 Requirements, scenarios and use cases 42
3.2.5 Test environment 43

3.2.6 Test setups, tools, hooks and probes 43
3.2.7 Test description 44
3.2.8 Test outcome 49
3.2.9 Integration ethics and security 50
3.3 Open Call FAT’s 51
3.3.1 Third Party: SensiNact 51
3.3.2 Third Party: OM2M 83
3.3.3 Third Party: INTER-HARE 92
3.3.4 Third Party: Mission Critical operations based on IoT analytics 181
3.3.5 Third Party: Early Warning System (EWS) 194
3.3.6 Third Party: Senshook 225
3.3.7 Third Party: SOFOS 239
3.3.8 Third Party: ACHILLES 248
3.3.9 Third Party: Inter-HINC 260
3.3.10 Third Party: Semantic Middleware 275
3.3.11 Third Party: SecurIoTy 290
3.3.12 Third Party: E3City 313
4 Open Call Third Parties Evaluation 323
4.1 Introduction 323
4.2 Second Evaluation 325
4.3 Large contributions 326
4.4 Small contributions 330
5 Conclusions 354

List of Figures

Figure 1: INTER-LogP high level design.....23
Figure 2: Port IoT platform and integration.....24
Figure 3: Terminal IoT platform and integration.....25
Figure 4. High-level view of the pilot 1.....27
Figure 5. Representation of the pilot 3.....28
Figure 6. High-level view of the pilot 4.....29
Figure 7. High-level view of the pilot 2.....30
Figure 8. SOAP UI test setup example.....33
Figure 9. WSO2 OAuth based client authentication test suite.....33
Figure 10. SoapUI test hook example.....35
Figure 11: INTER-Health Hardware overview.....38
Figure 12: INTER-Health Software overview.....39
Figure 13: SensiNact Gateway overall architecture......51
Figure 14 SensiNact Southbound and Northbound bridges.....53
Figure 15 SensiNact Gateway internal architecture.....53
Figure 16 SensiNact Service and Resource model.....54
Figure 17 sensiNact’s service oriented approach.....55
Figure 18 SecuredAccess Sequence Diagram.....60
Figure 19 Access right inheritance diagram example.....61
Figure 20: Architecture of a sNa component.....62
Figure 21: JSON example of a sNa component.....63
Figure 22: Lifecycle of an application.....64
Figure 23: sensiNact Studio Graphical User Interface......65
Figure 24: Gateway configuration......66
Figure 25: Gateway connection......66
Figure 26: Application creation......67
Figure 27: Application deployment.....68
Figure 28: Application management resources......69
Figure 29: Application start-up......70
Figure 30 StudioWeb initial screen.....71
Figure 31 StudioWeb connect.....72
Figure 32 StudioWeb gateway content.....72
Figure 33 StudioWeb: Sensor data.....73
Figure 34 StudioWeb gateway disconnection.....73
Figure 35 Integrating sensiNact and Inter-IoT catalog.....74
Figure 36: Test set-up.....84
Figure 37: INTER-HARE network environment.....92
Figure 38: Example of INTER-HARE transport network.....93
Figure 39: Ring structure of the LPWAN......93
Figure 40: Zolertia RE-Mote......95
Figure 41: CC1200 – TX Current Consumption vs TX Power at Different Temperatures.....99
Figure 42: Path Loss, Shadowing and Multipath versus Distance [1].....100
Figure 43: Range coverage for the TI CC1200 transceiver.....101
Figure 44: Range coverage for the TI CC2538 transceiver.....101
Figure 45: MAC sublayer consisting of TDMA slots and CSMA/CA technique inside them.....103
Figure 46: Beaconing system in the INTER-HARE platform.....104
Figure 47: Primary and secondary beacon periods.....105

Figure 48: Example of a staggered wakeup pattern in a 3-ring LPWAN performing uplink communications. 105
Figure 49: Simplified algorithm of CSMA/CA 106
Figure 50: X-MAC's short preamble approach 107
Figure 51: Unicast transmission in ContikiMAC 107
Figure 52: Broadcast transmission in ContikiMAC 108
Figure 53: Diagram of the query-driven data delivery model operation 108
Figure 54: Diagram of the event-driven data delivery model operation 109
Figure 55: Diagram of the continuous data delivery model operation....109
Figure 56: Example of address allocation for a typical INTER-HARE deployment 113
Figure 57: Diagram of the network association mechanism operation 114
Figure 58: Diagram of the STA association operation....114
Figure 59: e2e ACK operation 116
Figure 60: Network topology of the multi-hop LPWAN from Figure 59 117
Figure 61: State of the network from Figure 60 after the corresponding e2e ACK 117
Figure 62: STA's decision flowchart to stay awake or go to sleep before the start of a new transmission window 118
Figure 63: Mechanism of RTT averaging....127
Figure 64: Computation of RTT link between a parent and a child when sending more than one data segment 127
Figure 65: Gateway architecture split into two parts 130
Figure 66: Zolertia RE-Mote 137
Figure 67: Zolertia Orion Router 137
Figure 68: 3D model of UPF facilities 139
Figure 69: Office 'A' detail and dimensions....140
Figure 70: Office 'B' detail and dimensions....141
Figure 71: Office 'C' detail and dimensions....141
Figure 72: Integration and Factory test setup overview....142
Figure 73: TS_01 network topology....143
Figure 74: TS_02 network topology....144
Figure 75: TS_03 network topology....144
Figure 76: TS_04 network topology....145
Figure 77: TS_05 network topology....145
Figure 78: TS_06 network topology....146
Figure 79: TS_07 network topology....146
Figure 80: TS_08 network topology....147
Figure 81: INTER-HARE monitoring tool based on Java 148
Figure 82: A Zolertia RE-Mote device with its led switched on in red 149
Figure 83: Aaronia Spectran HF-6065 spectrum analyzer 149
Figure 84: Screenshot of MCS spectrum analyzer....150
Figure 85: Zolertia RE-Mote front view 172
Figure 86: Zolertia RE-Mote back view....172
Figure 87: Zolertia RE-Mote schematic front view 174
Figure 88: Zolertia RE-Mote schematic back view....174
Figure 89: Zolertia Orion Router schematic view 177
Figure 90: BeagleBone Black front view 177
Figure 91: BeagleBone Black rear view....177
Figure 92: BeagleBone Black Rev. A5C technical specifications 178
Figure 93: DHT22 temperature and humidity sensor 179
Figure 94: Main features of the DHT22 temperature and humidity sensor 179
Figure 95: Electrical parameters of the DHT22 temperature and humidity sensor 180

Figure 96: Zolertia Re-mote's analog connector pin-out 180
Figure 97: Scope of MiCrOBIoTa activities. 181
Figure 98: Overall system description...181
Figure 99. IoT-aided Mission Critical operations scenario..182
Figure 100. High-level perspective of the integration. 183
Figure 101. Use case perspective of the integration. 184
Figure 102: Typical EWS architecture (top) and the SEMIoTICS architecture (bottom). 196
Figure 103: EWS to detect accident risks and accidents at the port of Valencia. 197
Figure 104: Mosquitto trap system architecture. 225
Figure 105: Web Application of Senscape Connector...227
Figure 106: Virtual gateway bundles architecture. 228
Figure 107: Diagram of the Web Interface Front-end testing. 232
Figure 108: The proposed SDN/NFV end-to-end IoT Gateway overview............................239
Figure 109: SOFOS Integration and Factory test setup overview. 240
Figure 110: SOFOS Integration and Factory test setup logical topology.............................241
Figure 111: Collaboration approach of the proposed SDN/NFV end-to-end IoT infrastructure within the INTER-IoT architecture...241
Figure 112: Detailed approach of SDN applicability in SOFOS experiment on top of the INTER-IoT testbed. 242
Figure 113: FAT test setup of SOFOS solution..244
Figure 114: The concept of the ACHILLES project. 248
Figure 115: Testing System..249
Figure 116: Message sequence diagram of the core functions of the testing system..........250
Figure 117: INTER-HINC Architecture overview. 260
Figure 118: Overview of components in INTER-HINC. 262
Figure 119: An example of infrastructure configuration for INTER-HINC as the system under test (SUT)..263
Figure 120: Overall architecture and its interaction with INTER.IoT. 275
Figure 121: Subscription and notification workflow. 276
Figure 122: Workflow of the cancellation of the subscription. 277
Figure 123. Integration of the Semantic Middleware with the IoT framework 278
Figure 124. Workflow of the scenario 282
Figure 125: SecurIoTy overview. 293
Figure 126: SecurIoTy architecture with the DocRAID crypto proxy 293
Figure 127: test architecture. 296
Figure 128: Bug reporting process 297
Figure 129: System e3tcity description. 313
Figure 130: Diagram of solution e3t..316
Figure 131: INTER-IoT areas covered by the Small Contribution third parties. 324
Figure 132: INTER-IoT areas covered by the Large Contribution third parties....................325
Figure 133: OM2M bridge integration in INTER-IoT architecture. 327
Figure 134. sensiNact bridge integration in INTER-IoT architecture. 329
Figure 135: INTER-HINC architecture. 331
Figure 136: SOFOS proposed SDN/NFV end-to-end IoT Gateway. 333
Figure 137: D2D interoperability in INTER-HARE...335
Figure 138: MiCrOBIoTa integration in INTER-IoT architecture...337
Figure 139: SENSHOOK Block Diagram...340
Figure 140: SENSHOOK use case..340
Figure 141: ACHILLES concept architecture 343
Figure 142: ACHILLES integration in INTER-IoT Gateway 343

Figure 143: Conceptual architecture of semantic brokering for context-aware decision support 346
Figure 144: Integration in INTER-IoT....346
Figure 145: Integration in INTER-IoT....349
Figure 146: SecurIoTy Integration in INTER-IoT....351
Figure 147: E3TCITY D2D interoperability....352
Figure 148: E3TCITY MW2MW interoperability....353

List of Tables

Table 1: FAT test execution sign-off 20
Table 2: Deliverable checklist. 30
Table 3: Component version overview. 31
Table 4: Requirements vs test mapping. 31
Table 5: Scenario vs test mapping. 31
Table 6: Test outcome overview 37
Table 7: Local Server software components 40
Table 8: Android Phones applications 40
Table 9: Healthcare Professional browsers 40
Table 10: Integration components. 41
Table 11: Deliverable checklist 41
Table 12: Component version overview 41
Table 13: Requirements vs test mapping 42
Table 14: Scenario vs test mapping. 43
Table 15: Test outcome overview 50
Table 16: Resource types. 55
Table 17: Resource's access methods. 56
Table 18: Types used in the JSON component. 63
Table 19: Functions supported by the plugins of the AppManager. 64
Table 20: SensiNact Domain specific language basic syntax 68
Table 21: Application management resources 69
Table 22: Deliverable checklist 75
Table 23: Component version overview 75
Table 24: Requirements vs test mapping 76
Table 25: Test outcome overview 82
Table 26: Test for each OM2M component. 85
Table 27: Deliverable checklist 85
Table 28: Component version overview 85
Table 29: Requirements vs test mapping 86
Table 30: Scenario vs test mapping. 86
Table 31: Test outcome overview 91
Table 32: Key characteristics of Contiki, TinyOS, FreeRTOS, and RIOT 95
Table 33: Latest releases of the considered Operating Systems 96
Table 34: Current consumption value of the transmitting operational mode depending on the selected output power 98
Table 35: Current consumption value of the transmitting operational mode depending on the selected output power 99
Table 36: Range coverage computation for the TI CC1200 transceiver 102
Table 37: Range coverage computation for the TI CC2538 transceiver 103
Table 38: Link layers used in the INTER-HARE platform 103
Table 39: Main features of the INTER-HARE addressing system 112
Table 40: Available space for the application layer 119
Table 41: Packet headers encoding in INTER-HARE 120
Table 42: Field content of management frames 120
Table 43: Type descriptions of management frames 121
Table 44: Field content of beacons 122

Table 45: Content of the ‘association_info’ field....122
Table 46: Field content of a data packet....123
Table 47: Field content of a statistics packet....124
Table 48: Network performance metrics generated by the INTER-HARE gateway....125
Table 49: Field content of a link ACK packet....128
Table 50: Field content of an e2e ACK packet....128
Table 51: Deliverable checklist....131
Table 52: Component version overview....131
Table 53: Requirements vs test mapping....134
Table 54: Scenario vs test mapping....135
Table 55: List of INTER-HARE testbed components....136
Table 56: Comparison table among Arduino UNO, Raspberry Pi and BeagleBone....137
Table 57: Estimated pilot equipment....139
Table 58: Summary of offices’ dimensions....140
Table 59: Test setups summary....143
Table 60: Test tools summary....147
Table 61: Test hooks summary....151
Table 62: Definition of error configurations....152
Table 63: Test probes summary....152
Table 64: Summary of FAT tests and definition....154
Table 65: Diagram compiling the different test setups....156
Table 66: List of requirements to be analyzed in each test....157
Table 67: Test outcome overview....169
Table 68: Zolertia RE-Mote operational values....173
Table 69: Deliverable checklist....185
Table 70: Component version overview....185
Table 71: Requirements vs test mapping....186
Table 72: Scenario vs test mapping....186
Table 73. Data sources.....198
Table 74: Deliverable checklist....199
Table 75: Requirements vs test mapping....200
Table 76: Scenario vs test mapping....201
Table 77: Test outcome overview....223
Table 78: Deliverable checklist....229
Table 79: Component version overview....229
Table 80: Requirements vs test mapping....230
Table 81: Scenario vs test mapping....230
Table 82: Test outcome overview....237
Table 83: Deliverable checklist....242
Table 84: Component version overview....242
Table 85: Requirements vs test mapping....243
Table 86: Scenario vs test mapping....243
Table 87: Test outcome overview....246
Table 88: List of GW components that will be used during the Tests....250
Table 89: Deliverable checklist....251
Table 90: Component version overview.....251
Table 91: Requirements vs test mapping.....252
Table 92: Scenario vs. test mapping....252
Table 93: Test outcome overview.....259
Table 94: Requirements vs test mapping....261
Table 95: Scenario vs test mapping....261

Table 96. Table containing the components and interface overview of the used IoT components and the tests that will test these IoT components 278
Table 97: Deliverable checklist 279
Table 98: Component version overview..........279
Table 99: Requirements vs test mapping 280
Table 100: Scenario vs test mapping..........280
Table 101: Test outcome overview 288
Table 102: Test categories. 291
Table 103: Tested components. 291
Table 104: Security measures. 292
Table 105: Deliverable checklist 294
Table 106: Component version overview..........294
Table 107: Requirements vs test mapping..........295
Table 108: Requirements vs test mapping..........298
Table 109: Alpha version quality criteria 298
Table 110: Beta version quality criteria..........299
Table 111: Release version quality criteria 299
Table 112: Criticality description..........300
Table 113: Priority description 300
Table 114: Test outcome overview 311
Table 115: Description of the components..........314
Table 116: Deliverable checklist 314
Table 117: Component version overview..........315
Table 118: Requirements vs test mapping..........315
Table 119: Test outcome overview 322
Table 120: INTER-OM2M Evaluation. 328
Table 121: sensiNact platform Evaluation..........330
Table 122: INTER-HINC Evaluation. 332
Table 123: SOFOS Evaluation. 334
Table 124: INTER-HARE Evaluation. 336
Table 125:MiCrOBIoTA Evaluation. 339
Table 126: SENSHOOK Evaluation. 341
Table 127: ACHILLES Evaluation. 344
Table 128: INTER-EWS evaluation. 347
Table 129: Semantic Middleware Evaluation. 349
Table 130: SecurIoTy Evaluation. 352
Table 131: E3TCity Evaluation. 353

Acronyms

AIOTI	Alliance for Internet of Things Innovation
API	Application Programming Interface
CCB	Change Control Board
EC	European Commission
FAT	Factory Acceptance Test
ICT	Information and Communication Technology
IEEE	Institute of Electrical and Electronics Engineers
IPR	Intellectual property rights
IoT	Internet of Things
IoT-EPI	IoT European Platform Initiative
REST	Representational State Transfer
SAT	Site Acceptance Test
EWS	Emergency Warning System

1 Introduction

INTER-IoT project is aiming at the design, implementation and experimentation of an open cross-layer framework, an associated methodology and tools to enable voluntary interoperability among heterogeneous Internet of Things (IoT) platforms. The proposal will allow effective and efficient development of adaptive, smart IoT applications and services, atop different heterogeneous IoT platforms, spanning single and/or multiple application domains.

Most current existing sensor networks and IoT device deployments work as independent entities of homogenous elements that serve a specific purpose, and are isolated from “the rest of the world”. In a few cases where heterogeneous elements are integrated, this is done either at device or network level, and focused mostly on unidirectional gathering of information. A multi-layered approach to integrating heterogeneous IoT devices, networks, platforms, services and applications will allow heterogeneous elements to cooperate seamlessly to share data, infrastructures and services as in a homogenous scenario.

This document describes the Factory Acceptance Test plan for the INTER-IoT projects which is part of the experimentation step.

One of the main goals of INTER-IoT is to overcome fragmentation caused by typical IoT platforms being oriented to a specific solution, stakeholder and application domain. The cross-domain use case will show how verticality is avoided in INTER-IoT. The rationale behind this use case is that future IoT applications will not aim at a single application domain but multiple domains in which devices, networks, platforms, services or generated data will interact.

The scenarios defined in the cross application domain use case will integrate platforms from the two application domains in consideration, and also from different application domains (e.g. smart grid or smart cities). This use case will prove the extendibility of the project outcomes, achieving interoperability between IoT platforms from different application domains. Several scenarios have been foreseen in which IoT platforms from different application domains may be required to interoperate, e.g. logistics and health monitoring of transport workers for labour risk prevention, however new cross domain scenarios will be defined during the execution of the project and after the resolution of the Open Call, including e.g. road IoT ecosystems; supply chains or emergency response services IoT ecosystems used in fire brigades, ambulances or security forces.

According to the Grant Agreement the field trials will be successful once the following conditions are met: “Trials of INTER-IoT concept, with involvement of 400 smart objects in the logistics use case and 200 subjects (with wearable devices) in the m-Health use case, and ~500 IoT units in the cross domain use case. Extensive testing of results of application of the INTER-IoT framework to instantiate multi-IoT-platform systems in real-world scenarios, validated by the corresponding stakeholders.”

2 Test strategy and approach

This Factory Acceptance Test is performed to test and prove the system is operational and functional and complies with the defined requirements. The FAT takes place in a lab environment and tests the solution before it is deployed in the field. The FAT tests if the solution meets the specification, and if it is fully functional. It includes a check of completeness, verifies requirements, and proves functionality. This can be either by simulation or a conventional functional test. The FAT Document describes the Factory Acceptance Testing plan and describes or points to previously defined test plans, use cases and test scenarios used during testing. The test outcomes will be placed in the FAT document itself.

2.1 Testing strategy

This document will describe the system, test setup, tooling, test strategy, test activities and test results for the lab setup. During these tests the system readiness for field deployment will be tested and proven. For this test, the following stakeholders should be present:

- Project managers (From manufacturer and customer)
- Key engineering personal (System Architect, Lead Engineer, Integrator)
- Operators
- Maintenance personal

During FAT testing the readiness of the system is shown to the customer and the customer can use the system for the first time in an actual system setup to get a better feeling for the new system. The operators and maintenance personal can get a view of how the system is operated and maintained. During the FAT the customer will work with the system for the first time which will most likely lead to some minor changes to the system to fix operation inconveniences before the system is deployed in the field. During testing the result should be written in the FAT document which should be signed off by all the attendees at the end of the test. A FAT will be conducted 2 to 4 weeks before actual deployment in the field.

2.2 Entrance criteria

To start a FAT for this project the following deliverables should be present/ready:

- Integration of the tested and validated system component into a to test system
- Validation and Test reports of the system components
- FAT document
- System test setup (as much actual hardware as possible)
- Test applications and tooling (e.g. for performance testing)

2.2.1 Integration of the tested and validated system components

The system components that make up this system should all be tested and validated. The units that make up the components should be unit tested and validated on interface level, normal use and boundary checking, error handling, performance, etc. After integration into components the components should be integration tested the same way on interface level.

This applies for the pilot specific components as well as for the used IoT components. For each component a test/validation report for the used version should be available stating that the component has been tested and validated and passed this test.

2.2.2 Validation and Test reports of the system components

The test and validation reports of the system components used in this project should provide an overview of the tests done on the used system components and the outcome of these tests. Each component should have passed the tests and validation before used in this FAT. The version in these reports shall match the version used in the system release to be FAT tested.

2.2.3 FAT document

A printed version of this document which should be checked before start of the test. Any issue found should be manually corrected. The outcome and remarks of each test as well as the final outcome should be written in the copy of this document during testing and be signed at the end of the FAT. The signed copy of this document will serve as an acceptance on the system to start field integration.

2.2.4 System test setup, Test applications and tooling

The test setup as described in this document should be present and checked for completeness. See 0 **Test environment** for the system setup.

2.3 Acceptance Criteria

The FAT acceptance criteria is a signed copy of this FAT document as this contains all the needed deliverables and tests to complete the factory acceptance testing. After a successful FAT the system can be integrated in the field and proceed to SAT testing.

2.4 Testing types

The FAT document will define the testing types per project in detail. The following list provides an example of testing types one could think of:

- manual data load
- interface using scripted data
- interface bounds checks
- converted data load
- converted data inspection
- backup and recovery
- database auditing
- data archival
- security
- locking
- batch response time
- online response time
- network stress
- stress testing
- security
- live data
- live environment
- error handling

The following table shall contain the test attendance list. Each individual in this list will sign of on his/her presence, the deliverables and the outcome of the tests.

Test execution date			
Test execution time			
System version	**V1.0**		
Name	**Company**	**Function title**	**Signature**

Table 1: FAT test execution sign-off

2.5 Suspension and resumption criteria

When one of the entrance criteria is not met at the start of the test the complete or part of the tests will be suspended until it/they are met.

When during testing one of the entrance criteria's is found to be unsatisfactory parts of the tests or the complete test can be suspended until it is met. In most case though the test will be executed completely to prove the rest of the system.

When a test is suspended or after execution is not accepted, the found issues shall be solved and a new FAT test shall be performed. The new FAT will then again run all tests to confirm the issues are solved and no new issues have been introduced.

2.6 Change Control Board

A Change Control Board (CCB) will be defined for each project.

The change control board will consist of the following persons:

• Carlos Palau	(UPV)
• Eneko Olivares	(UPV)
• Flavio Fuart	(XLAB)
• Johan Schabbink	(NEWAYS)
• Dennis Engbers	(NEWAYS)
• Pablo Giménez	(VPF)
• Gema Ibáñez	(SABIEN)

2.7 Defect Reporting

Please see D6.1 System Integration Plan for defect reporting.

3 Factory Acceptance Tests

This section describes the Factory Acceptance Test (FAT) plan of the two large use cases of INTER-IoT, namely INTER-LogP and INTER-Health and the the twelve third parties involved in the INTER-DOMAIN use case, as separate pilots. During development units and components are tested and validated by the developers. This applies for the IoT framework components as well as for the project specific units and components. When all the components needed for the project are complete, tested and validated they are integrated into the system as defined for this project.

This system will then undergo this Factory Acceptance Test to test the readiness of the system. This is done in a LAB setup which approaches the actual field deployment as much as possible. When the FAT has been successfully executed and has been approved the system can advance to field integration and undergo the Site Acceptance Test (SAT).

This section describes all aspects of the FAT, form defining the versions of the used components and deliverable checklist up, test setup, tooling, test description, etc. to be able to test the readiness of the system under test.

For each pilot the structure of the FAT document is divided into the following sections:

- Section 1: System description
- Section 2: Use case oriented pilots
- Section 3: Deliverables and version overview
- Section 4: Requirements, scenario and Use cases to test
- Section 5: Test environment
- Section 6: Test description
- Section 7: Test outcome overview
- Section 8: Ethics

The information provided in this document will be handled separately for each of the pilots, but for simplicity all the FAT information is compiled in a single document.

3.1 INTER-LogP FAT

3.1.1 System description

In this chapter are described the systems of the main actors involved in INTER-LogP.

3.1.1.1 Port systems description

In recent years, there is a need to share real-time data between different companies in order to offer new services to their clients. In the port environment, there are many different companies, each with its own independent system. Nowadays they only exchange some logistic documentation and not sensor data. This new exchange of data should be done in a secure and robust way

The goal of INTER-LogP pilot is to demonstrate the need for a system that allows the exchange of data and messages among the different actors of the port community. In this case, as can be seen in Figure 1, there are three main actors: the port, the terminal and the haulier company. INTER-IoT has to provide interoperability between the IoT platforms of the port and the terminal, and give access to other devices from other companies, like trucks.

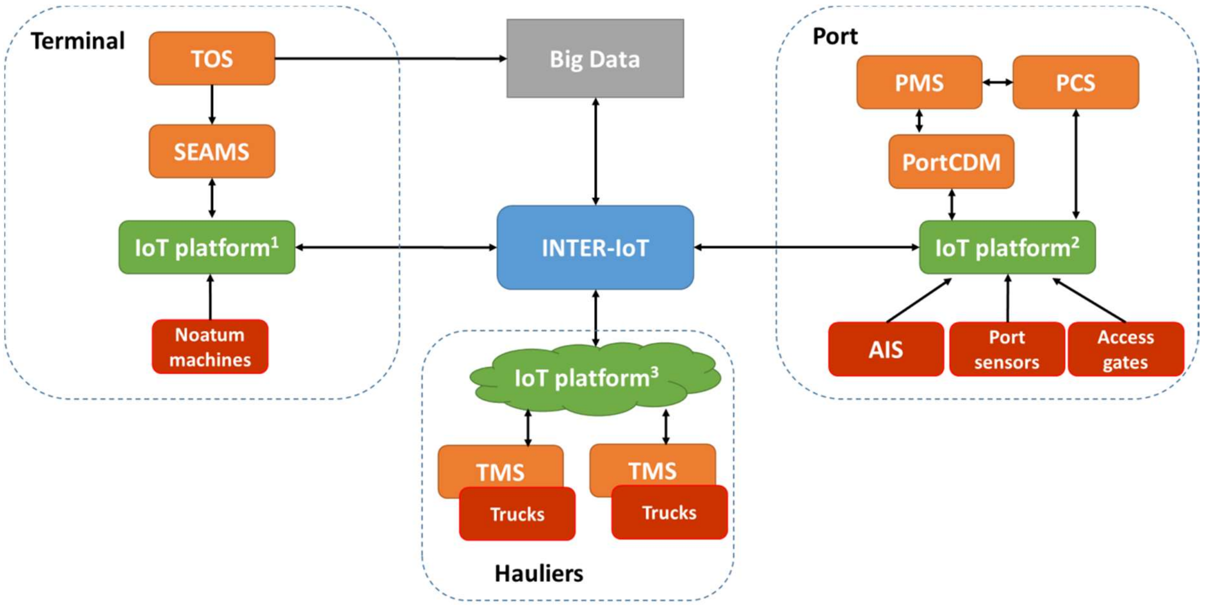


Figure 1: INTER-LogP high level design.

Both the port and the terminal have a large number of sensors and devices that produce large amounts of data, which can be interesting for other entities. Furthermore, they need data from other companies to provide a better service to their clients.

3.1.1.2 Port authority

The port authority has several sensors distributed throughout the port that provides data for management and operation. Most of that data is confidential, but other can be shared, adding value to other companies.

The architecture for providing interoperability from the existing infrastructure is the one that can be seen in Figure 1. Currently, the port authority has a common database where all the data is stored coming from different systems (in red). It uses WSO2 to provide an IoT architecture in two ways: data in real time through the Message broker and historic data through the Data services server and Enterprise service bus.

Because the port has its own platform, the integration with the INTER-IoT is done through the INTER-MW. It needs a bridge in the middleware layer in order to interoperate with other platforms. Another integration could be done if you need to deploy new devices or sensor in a place without wired connection. In that case, the INTER-IoT gateway can be used to connect with the IoT platform all the sensors.

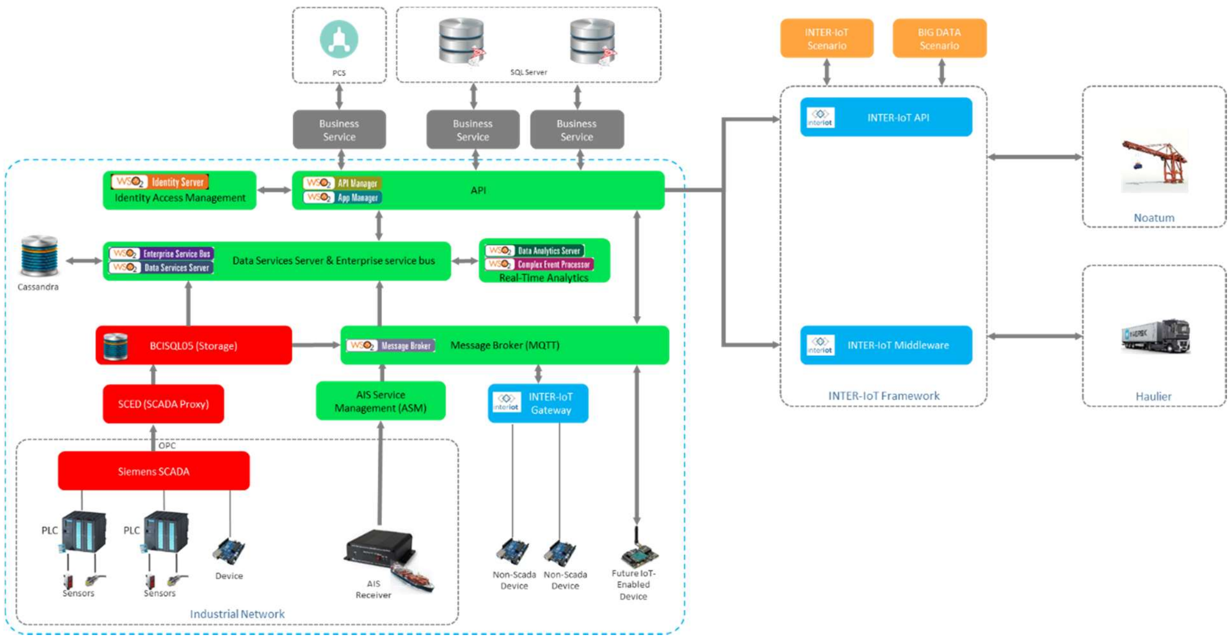


Figure 2: Port IoT platform and integration

WSO2 is an open source service-oriented architecture (SOA) middleware. It is designed with independent components, so it can be adapted for a lean targeted solution to enterprise applications. WSO2 products use Java technology and are built on top of WSO2 Carbon, a SOA middleware platform. Carbon makes use of Apache Axis2 and encapsulates SOA functionality such as data services, business process management, ESB routing/transformation, rules, security, throttling, caching, logging and monitoring.

Not all components are used as stand-alone implementations. Many of them are used to supplement the capabilities or add functionality to an implementation of the Enterprise Service Bus. The main WSO2 components deployed in the IoT platform are:

Integration

- Enterprise Service Bus: Allows developers to connect and manage systems and software in accordance with SOA Governance principles.
- Data Services Server: Provides a Web service interface for data stores.
- Message Broker: Translates, validates and routes messages between systems.

API Management

- API Manager: API management platform for creating, deploying and managing APIs to expose data and functionality of backend systems.

Identity Management and Security

- Identity Server: Connects and manages multiple identities across applications, APIs, the cloud, mobile, and Internet of Things devices.

Management and Governance

- App Manager: Facilitates the process of creating, deploying and managing applications.

Analytics

- Data Analytics Server: Real-time, batch, interactive and predictive analytics using enterprise data.
- Complex Event Processor: Real-time event processing and detection. Identify patterns from multiple data sources, analyse their impacts. Uses WSO2 Siddhi and Apache Storm.

3.1.1.3 Container terminal

The correct management of resources in a container terminal implies the monitorization of all the machinery, to be able to manage the resources properly. For that reason, in the Noatum terminal each of the machines (vehicles, cranes, etc.) provide massive data about up to 80 sensors per machine and second. There are around 300 monitored devices among machines and dynamic lighting on lamp posts.

As can be seen in Figure 3, data is sent from the machinery to the IoT Platform in two ways. Legacy sensors are collected once per second and inserted in the IoT Platform. New IoT devices are configured to send directly through MQTT or REST interfaces real-time data. In addition, the data will be stored in a non-relational database, providing faster access to information.

As in the case of the port, the IoT platform of the terminal will be integrated with INTER-IoT through the middleware layer and the API layer.

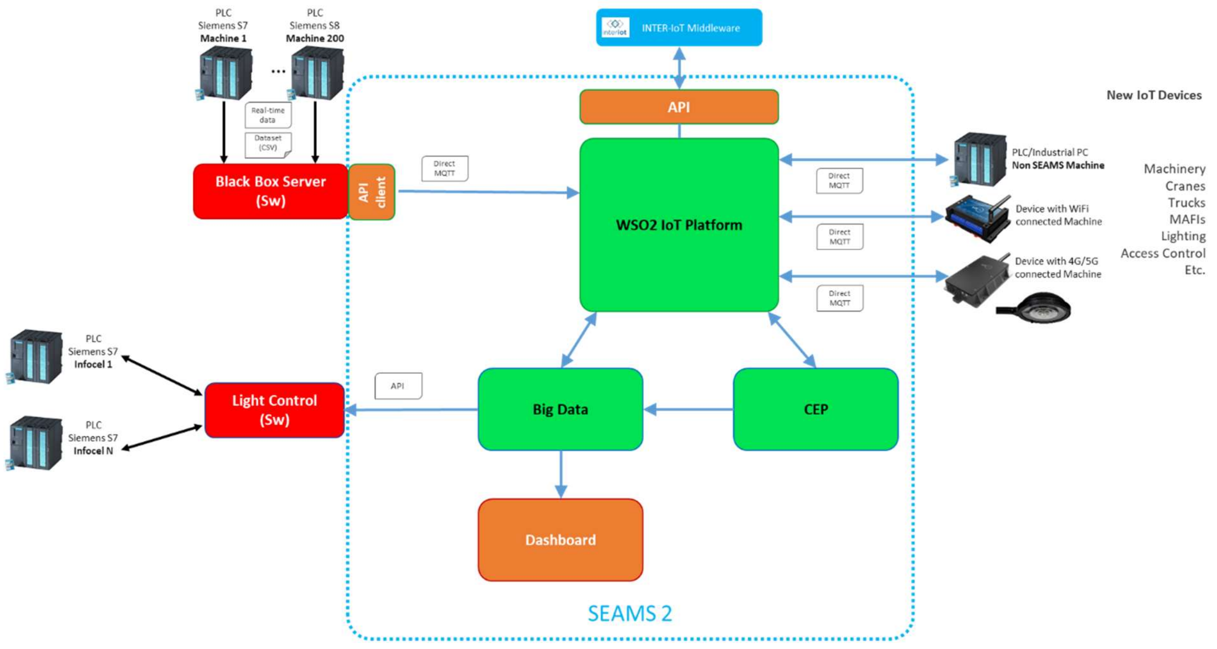


Figure 3: Terminal IoT platform and integration

The container terminal has its own server with its IoT platform. They are mainly interested in knowing the estimated time of arrival of the truck to the terminal to be able to manage its resources. Furthermore, the terminal gives access to other companies to some of their own data, such as the entry and exit of trucks by their access.

3.1.1.4 Haulier Company

The haulier company has a large fleet of trucks, which access the port daily. Each of them has a mobile app (MyDriving) installed in a mobile or a tablet that acts as a bridge between the vehicle and the IoT platform of the company. All the devices in the truck and the driver send the data to the IoT platform through the mobile app via Bluetooth.

The haulier company has an Azure IoT platform in the cloud, where their trucks send all the data from the vehicle. These data will be accessible to other companies as long as they are authorized and certain conditions are met, such as being inside the port area.

3.1.2 Use case oriented pilots

In this chapter are described the main pilots where all the functionalities of the developed components are shown.

3.1.2.1 INTER-LogP pilots

We have defined three pilots where we demonstrate some of the products developed in during the project. These pilots are:

- Pilot IoT access control, traffic and operational assistance
- Pilot Dynamic lighting
- Pilot Wind gusts detection

Pilot IoT access control, traffic and operational assistance

The main objective in the defined pilot is a service to control access, monitor traffic and assist the operations at the port. Several systems will be able to identify trucks and drivers using different devices. This information can be shared under certain predefined rules through interoperability between the platforms involved. This information can be used to monitor the truck inside the port by the Port Authority platform (security and safety purposes) and to manage more efficiently resources in the terminal. This also will allow avoid queues in the access gates to the port and the terminal.

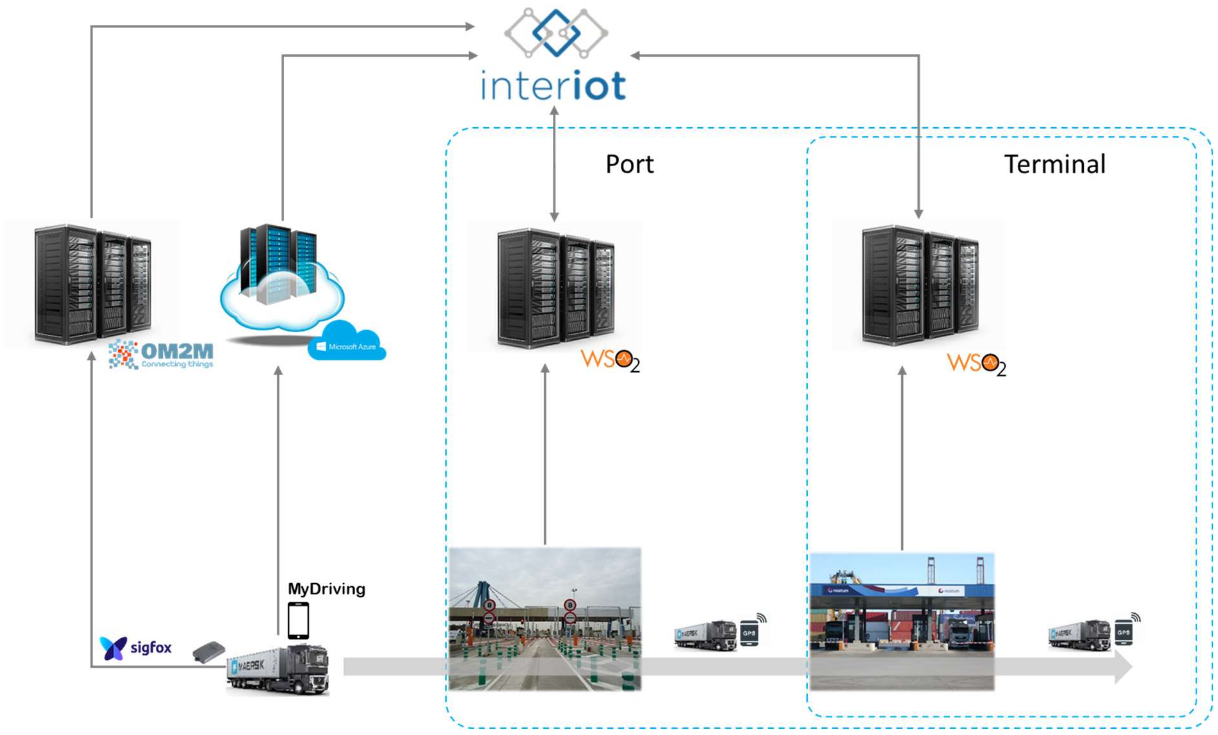


Figure 4. High-level view of the pilot 1

The main benefits we can get from this scenario is to obtain data regarding queues, congestion and temporary distribution of traffic, to manage efficiently the resources. Other important data is the position of the trucks while they are inside the port facilities, for safety and security. All these data can be shared between the port authority and the port terminals to improve the operation.

Pilot Dynamic lighting

The goal of this pilot is expand the smart illumination (Dynamic Illumination) in the yard of Noatum for the rail yard. In this case, the lighting posts are of the port authority of Valencia but the machinery is from Noatum, so it is needed an exchange of data between both companies to illuminate it properly during the operation.

As can be seen in the Figure 5, currently the rail yard (green area) is dimly lit with the container yard lighting posts. The objective is replace the road lighting posts (blue area) with a dynamic lighting system, which receive data from the terminal to change the degree of illumination.

The dynamic lighting system is based in the GPS position of the Noatum port equipment and long range PIR Sensors (presence sensors).

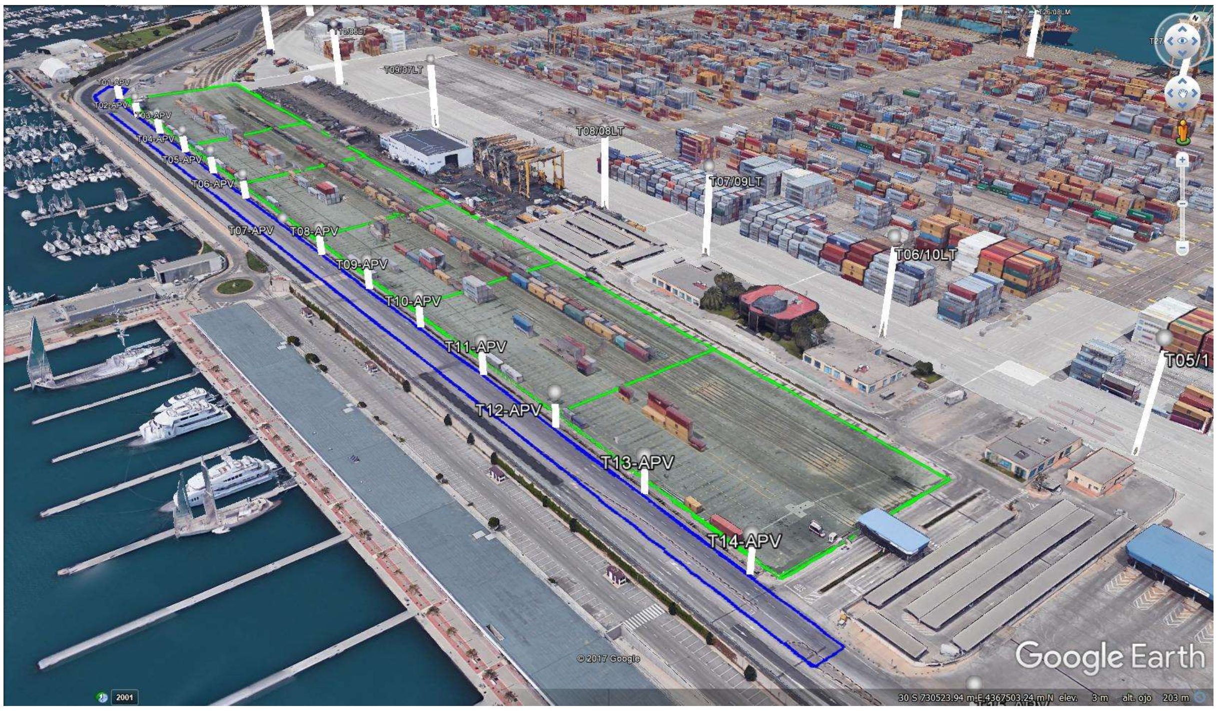


Figure 5. Representation of the pilot 3

The main benefits we can get from this scenario is an energy saving due to the adaptation of lighting to traffic and operation, and an improvement of the safety and security in the railway infrastructure due a better illumination.

This pilot will be deployed with two different approaches. Firstly, a pilot will be deployed only with the participation of the INTER-IoT partners. Then, there will be a second version in the INTER-DOMAIN pilots, which integrates the technology of the company involved in the Open call, called E3TCity.

Pilot Wind gusts detection

Currently, both the port authority and each of the container terminals in the port have anemometers to detect wind gusts. In situations where the wind speed exceeds a threshold, operations must be stopped for safety reasons. However, each terminal can only measure information in its presmises, so that there is not data of surrounding areas, making impossible to predict when the stronger wind gusts can be expected. This makes the first detection of strong wind a risky situation for the operators, since the hazard is only detected when there is already active. If they were able to receive this information before, they would stop the operation in a safer way.

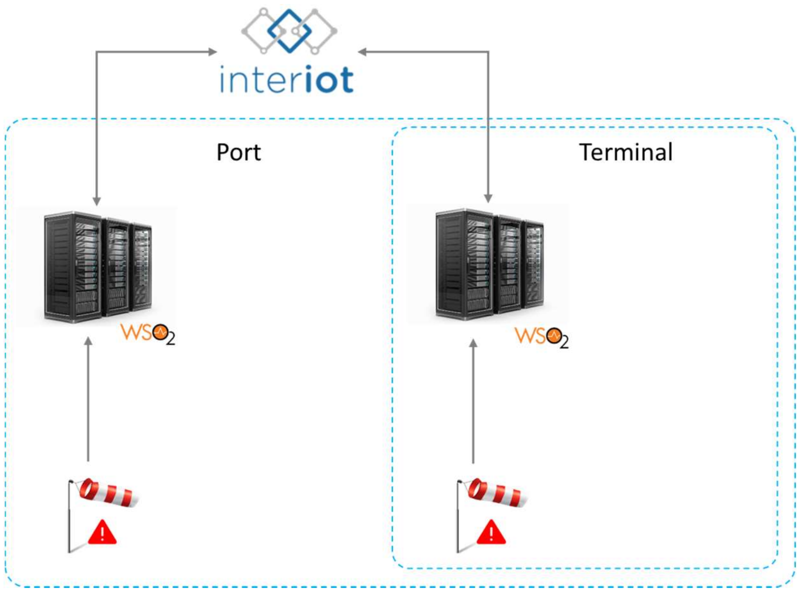


Figure 6. High-level view of the pilot 4

The main benefit we can get from this scenario is improve operational safety at terminals, enabling an early awareness system that could end up in less accidents due to environmental phenomena.

3.1.2.2 INTER-Domain pilots

In the open call some companies started to work in the INTER-IoT project as third parties in order to integrate and test the different INTER-IoT products. As a result of this work, several scenarios have been proposed where they will participate. This integration, together with the results, will be analyzed during this document and in the future versions of this one.

Pilot Health accident at the port area

Starting from the access control pilot, the trucks will be monitored once they are in the port facilities. The Emergency Warning System (EWS) will be monitoring the data coming from the truck and the driver. In case, it detect an accident or a medical problem, it will publish a notification to the port authority in a standard format (EDXL). Once the emergence control centre receive the notification, it can start communication with the driver with a push to talk protocol in the driver’s mobile.

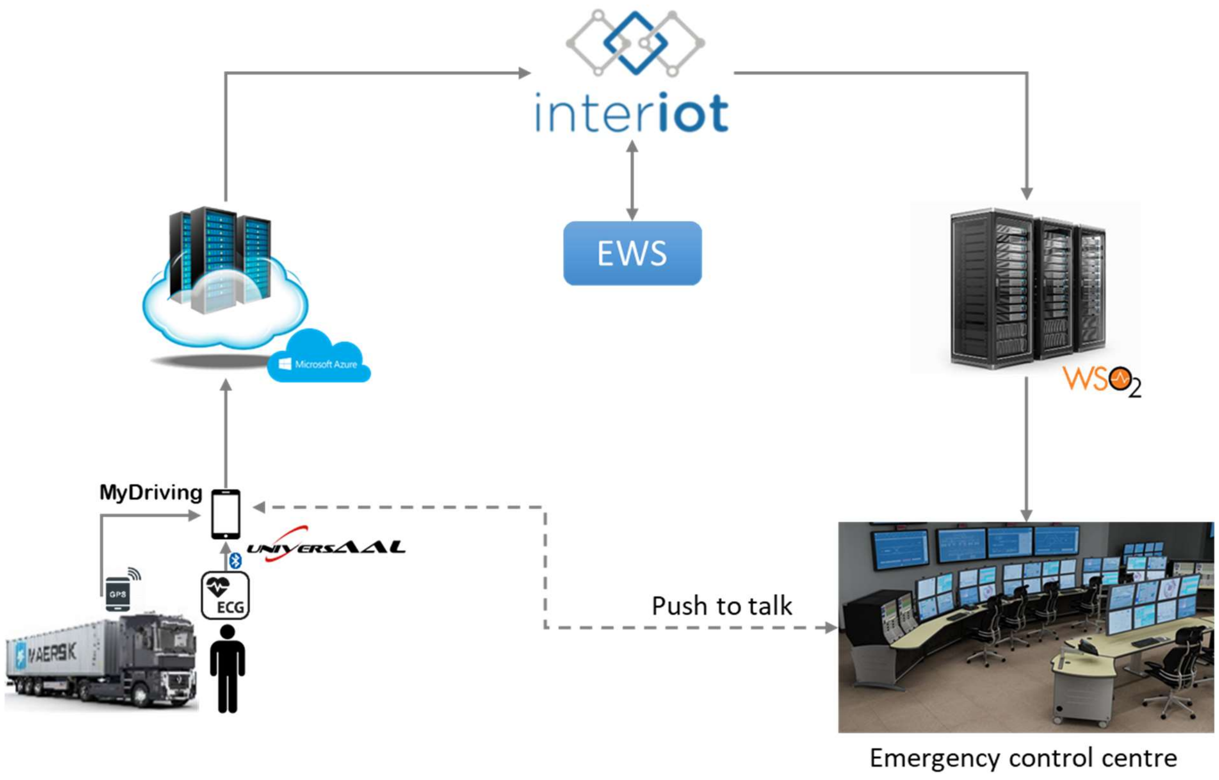


Figure 7. High-level view of the pilot 2

The main benefits we can get from this scenario are: apply in the port communications a standard format in accident reporting like EDXL, real time identification of the location of the accident, direct communication with the closest control centre when an accident occurs and monitoring driver's health if it is necessary.

In the development of this pilot are participating two of the companies involved in the Open call: University of Twente and Nemergent Solutions.

3.1.3 Deliverables and version overview

The following table contains a deliverable list which need to be signed of before FAT testing commences.

ID	Description	Check
Documents		
1	D6.2 Filled in FAT document	
Software		
4	WSO2 IoT platform	
5	Seams IoT platform	
6	Azure platform	
7	Spotlights and controllers	
Tools		
8	Wireshark	
9	SoapUI	

Table 2: Deliverable checklist.

The following table shows the software components and version of which the system release version consists of.

ID	Description	Version	Check
1	Physical gateway		
2	Middleware		

Table 3: Component version overview.

3.1.4 Requirements and scenarios

In this section, there are lists of requirements and scenarios covered in INTER-LogP.

3.1.4.1 Requirements

The following table provides an overview of the relation between the requirements and the test(s) that validate their implementation.

ID	Description	Covered by
Application		
166	Detection of passive physical entities to start communication with other platforms	TS_02
195	Provide the creation and monitoring of geofences	TS_01, TT_01 TT_02
246	Identification of an object through multiple techniques	TS_02
251	Ability of IoT platforms to coordinate with emergency systems.	TT_02

Table 4: Requirements vs test mapping.

3.1.4.2 Scenarios

The following table provides an overview of the relation between the scenarios and the test(s) that validate their implementation.

ID	Description	Covered by
6	Dynamic lighting in the port	T1.2
7	SCADA port sensor system integration with IoT platforms	T1.1, T1.2
8	SEAMS integration with IoT platforms	T1.1, T1.2
30	IoT access control, traffic and operational assistance	T1.1

Table 5: Scenario vs test mapping.

3.1.5 Test environment

Introduction

To test the functionality of the INTER-LogP in combination with the IoT framework a representative test system is needed. The test system needs to approach the "real world" as much as possible. The pilot setups must be recreated and proven. This chapter will describe this environment and the used hardware, software, tools and platforms.

Test environment

This paragraph describes the test environment and the complete system setup used during this FAT. Each of the IoT platforms has been tested independently and from now on, tests of interoperability among them will be made.

3.1.5.1 Test setups, tools, hooks and probes

This paragraph describes the test setups, tools, hooks and probes used in the by this document defined tests.

A test setup defines a setup used for a test, and describe the used system parts, interface etc.

Tools define the used COTS tooling that will be used during testing e.g. Logic analyzers, packet sniffer like e.g. Whireshark, developed scripts, etc.

Hooks are used to inject data or control parts of the system used to get the system to handle a needed scenario. This can be hidden, debug or service interfaces which contain the needed functionality.

Probes are used to gather logging from parts of the system under test to provide system feedback and to prove that the system is handling the scenario as it should.

The following paragraphs will define the used setups, tools, hooks and probes which will be referred to by the test descriptions.

TS_01 Publication of data from a legacy data source

One of the tests we designed consists in having a legacy database as an event generator and allowing different users to subscribe to those events. To do so, we designed a data injector, injected data in the database, detected these changes, and published them classified per topics in real time to a shared mean where a set of end users can subscribe to these topics and receive the information. In this test, we used MicroSoft (MS) SQL Server as legacy database, MS SQL Server Service Broker (MSSB) to detect the changes in the database, MSSB External activator to behave as a MQTT publisher, and Mosquitto as a MQTT broker and its client as subcriber.

TS_02 WSO2 Platform

In the development phase of INTER-LogP pilots and related testbeds we tried to follow the TDD (test-driven development) process. This is a well-known practice that relies on the delivery of tested and ready-to-use logic in form of chunks. TDD best practices mainly relies on following the TDD's life-cycle: write the tests, run tests, implement logic, runt tests, refactor, run tests, and so on. In fact, our implementation mainly focuses on the logic necessary for the integration of several port systems as well as development of several web-services to access data related to access control, traffic, meteorological information, etc. Therefore, we chose SoapUI as a suitable tool that allowed us to apply as much as we could the TDD life-cycle. In our test setup we can distinguish three types of tests: unit tests, integration tests and functional tests. First one focuses on testing the functionality of a small piece of logic regardless its relation with other logic. Second one proves that some implemented logic chunk or set of chunks can inter-work with existing logic of hosting platform (e.g. integration with web-services or client applications). Finally, the third one, checks the end-to-end functionality from the user perspective. Usually, unit tests are merely single requests in some messaging format (REST, SOAP, Mqtt, etc.). Integration tests are very similar to unit tests but their requests hit a part of the logic that depends on some interaction with a third application, database, etc. Functional tests may be composed of several test steps and they are grouped in test suites. Test suites may be composed of requests, test scripts to check validate some response or even mock services. Mock services are used to emulate some web service or functionality. It just runs a kind of simple servlet capable to send responses when a request hits the mock service. An example of functional test could be a test suit with necessary test requests emulating a client application authentication process. An example of test-setup in SOAP UI is shown in Figure 8

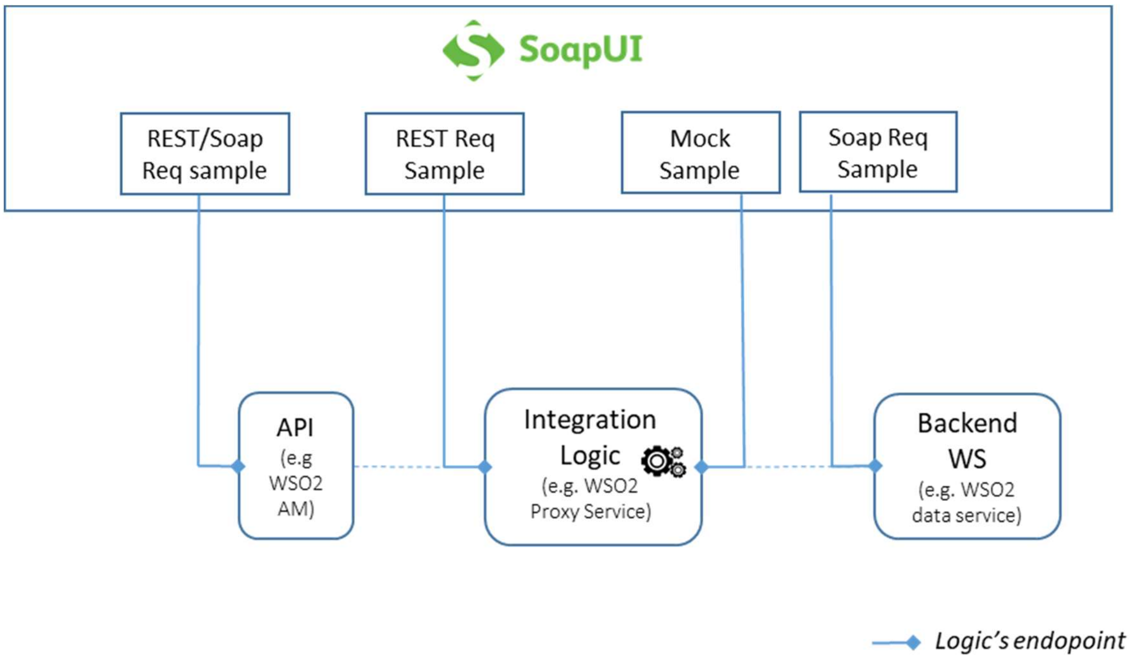


Figure 8. SOAP UI test setup example

The following picture shows an example of a test project that includes a test suite aiming to test the client authorization/authentication process of an application using the Oauth2 protocol:

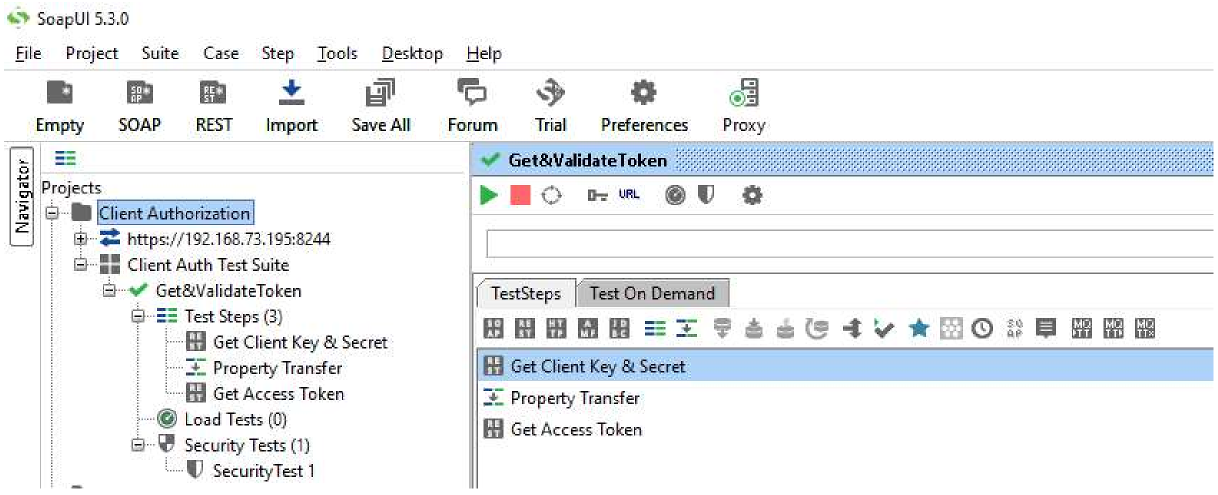


Figure 9. WSO2 OAuth based client authentication test suite

In above figure, the Client Auth Test Suite has only one test case: Get&Validate Token. This test case includes three test steps:

- Get Client Key & Secret: this is a REST request that invokes a WSO2 Identity Server endpoint responsible of providing a client Identifier and secret key.
- Property Transfer: above identifier and key is transferred to the Get Access Token request using this test step. For that *jsonPath* notation is used for the navigation across the received response.

Get Access Token: this is again a REST request that is finally used to get a valid token that will be used for accessing some API secured by Oauth2.

TT_01 MS SQL Server

Microsoft SQL database environment. We used it as example of legacy database. Test setup has been already described in previous sections. The main purpose of using this tool is to prove the requirement 188.

TT_02 MS Service Broker and Service Broker External activator

Two of the tools provided by MicroSoft SQL Server. The service broker allows capturing different events happening in the database (insertions, deletions, alterations, etc.) and perform actions on or with those. The external activator allows to execute a particular application external to MS SQL Server with the data that triggered the event.

TT_03 Mosquitto MQTT Broker

Mosquitto is a free lightweight Message Broker used. Mosquitto offers not only the broker itself, but also a publisher and a subscriber. It can be used as both the endpoints and the communication channel in our tests.

TT_04 SoapUI

This is a very advanced tool for test automation of applications and web-services deployed in SOA based configurations. This is a perfect tool to perform unit, functional and integration tests. One can hit APIs following several messaging protocols: REST, SOAP, Mqtt, JDBC, etc. Although this a commercial tool it also provides a functionally limited free version. In this project, the free version is used.

It allows the emulation of real user stories with web-services, APIs, database calls (via JDBC), etc. User stories are emulated by test suites. A test suite may include the following main test steps:

- Test Request: this is an http request template where different protocols can be used like SOAP, REST or JDBC.
- Property Transfer: a set of properties are taken from some response and placed into some fields in the following request.
- Groovy Script: a script may be implemented in order to make more complex processing of requests and/or responses.
- Delay: this is used to add some delay within the test suite execution.

TH_01 Restclient

Tool to debug restful web services. It allows sending customized messages using any of the methods in RFC2616 (HTTP/1.1) and RFC2518 (WebDAV) as well as receiving the associated responses. Messages can be built as HTTP requests with different combinations of headers, customized body, and addressed to a given URI. RestClient is integrated in Firefox as a free add-on.

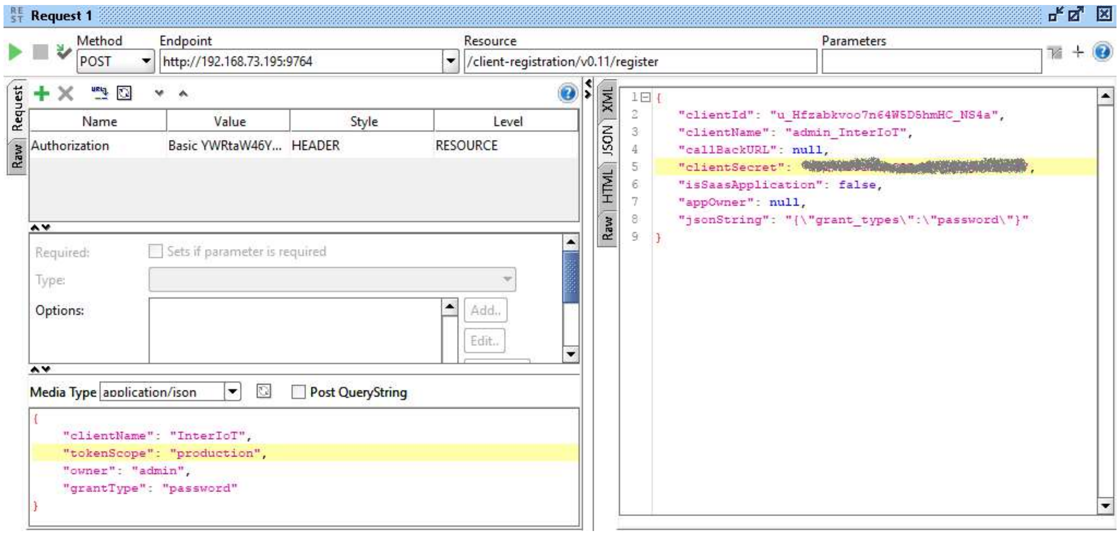
We use RestClient to test different Rest web services and tools. With RestClient we can send a particular input to the tool/service under test and verify, afterwards, that the result of performing a task matches what was expected. Similarly, we also use it to test the target behaviour against incorrect or incomplete messages.

TH_02 Self-designed data injector

We designed a data injector able to read data in a customizable way from a legacy data source and inject it into a targeted legacy data source. This injection is customizable and can be performed replicating the speed at which data was injected in the original data source or altering this speed. This injector is written in Java.

TH_03 SOAP UI

Regarding to SOAP UI, we could consider as test hooks all the configured SOAP and REST requests used to invoke implemented logic service. In our test scenarios, requests enter the WSO2 platform throughout API, proxy service or data service endpoints. The following picture shows an example of a SOAP UI hook (HTTP/REST request):



Method	Endpoint	Resource	Parameters
POST	http://192.168.73.195:9764	/client-registration/v0.11/register	

Name	Value	Style	Level
Authorization	Basic YWRtaW46Y...	HEADER	RESOURCE

Required: ☐ Sets if parameter is required
Type:
Options:

Media Type application/json ☐ Post QueryString

```
{
    "clientName": "InterIoT",
    "tokenScope": "production",
    "owner": "admin",
    "grantType": "password"
}
```

```
{
    "clientId": "u_Hfzabkvoo7n64W5D5hmHC_NS4a",
    "clientName": "admin_InterIoT",
    "callBackURL": null,
    "clientSecret": [redacted],
    "isSaasApplication": false,
    "appOwner": null,
    "jsonString": "{\"grant_types\":\"password\"}"
}
```

Figure 10. SoapUI test hook example

TP_01 TCPdump

Probably the most popular packet analyzer. It is a free tool available in most UNIX based systems (Linux, OS, etc.) as well as Windows (WinDump). TCPDump allows to capture or visualize the data flowing inwards or outwards a network interface with different levels of detail. For instance, data packets can be displayed at different levels (network, transport, etc.) or filtered by different parameters like MAC, IP, port, or protocol, among other.

We use TCPDump as a probe. In particular, we can use it to revise whether the content of the messages that we deliver to a platform, or that the platform outputs, are what were expected to be. A classical example of how TCPDump can be used is validating the messages received at the platform, finding whether a problem in the test is derived from an erroneous input or an erroneous processing of a correct input. The same procedure can be applied on its output.

3.1.6 Test description

3.1.6.1 IoT access control, traffic and operational assistance

Truck triggers information

ID	T1.1

Test	Verify the integration of all the components in the IoT access control, traffic and operational assistance pilot. The main objective in the defined pilot is a service to control access, monitor traffic and assist the operations at the port.
Setup	Deployment, installation and configuration of all the components.
Start	A truck is arriving to the port.
Req.	[27], [28], [166], [194], [195], [198], [245], [246], [268]
Input	Truck data
Output	Exchange of access data between the port and the terminal
Logs	INTER-LogP`1.1.log`
Outcome	Pass / Fail

Pilot Dynamic lighting

ID	T1.2
Test	Verify the integration of all the components in the Idynamic lighting pilot. The goal of this pilot is develop a smart illumination (Dynamic Illumination) in the yard of Noatum for the rail yard.
Setup	Deployment, installation and configuration of all the components.
Start	A truck or machinery is accessing to the rail yard area in the terminal.
Req.	[27], [28], [168], [198], [245]
Input	Data from PIR sensors
Output	The light level in the rail yard terminal is adjusted to the operation.
Logs	INTER-LogP`1.2.log`
Outcome	Pass / Fail

3.1.7 Test outcome overview

Test outcome

The following table will provide an overview of the test result of all the performed tests in this FAT.

Test	Description	Outcome
TS_01	Publication of data from a legacy data source	Pass / Fail
TS_02	WSO2 Platform	Pass / Fail
TT_01	MS SQL Server	Pass / Fail
TT_02	MS Service Broker and Service Broker External activator	Pass / Fail
TT_03	Mosquitto MQTT Broker	Pass / Fail
TT_04	SoapUI	Pass / Fail

TH_01	Restclient	Pass / Fail
TH_02	Self-designed data injector	Pass / Fail
TH_03	SOAP UI	Pass / Fail
TP_01	TCPdump	Pass / Fail
FAT Outcome		**Pass / Fail**

Table 6: Test outcome overview

3.1.8 Integration ethics and security

Introduction

During the Ethical Review it was requested that ethics had to be included within the project scope. This section addresses recommendations and proposed solutions to the ethical and security points of the integration.

For each pilot the ethics is discussed in last sections. The security aspects of each layer is also discussed after the ethics paragraphs as both normally are related and are important aspects to define for the success of the pilots.

The information for the pilots for both ethics and security comes from the partners and may be included in other documents as well.

INTER-LogP

The objective of this project is the interoperability, so it’s needed the exchange of data between components and platforms. But in the port sector, this data could be sensitive for the owner’s company, so agreements are necessary among the organizations to use the data only for the agreed purpose. And do not share the information with anyone else.

Furthermore, this process requires a high level of security. The security must be guaranteed in communications and in each of the intermediate components. This must be done through the use of secure and encrypted communication channels and with high security in the middleware.

3.2 INTER-Health FAT

3.2.1 System description

The following scheme shows the deployment of hardware equipment:

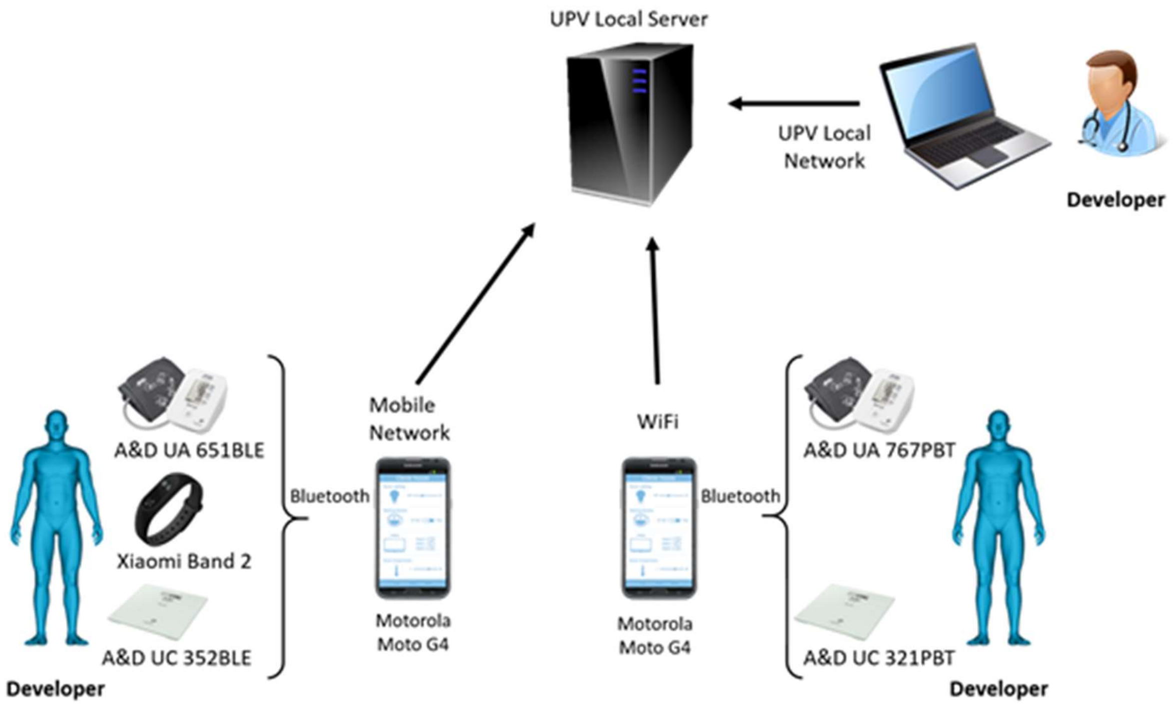


Figure 11: INTER-Health Hardware overview

The network in this test is owned by UPV-SABIEN so it will be possible to configure it to our needs. It will be set up to mimic the network available in the pilot but there is always a possibility of irreproducible, unexpected or unknown characteristics which later pose problems.

The Local Server is a virtual machine run at one of UPV-SABIEN server machines. It runs a Windows Server 2016 OS. It also runs several other non-related virtual machines so its performance may not be indicative of final performance in the field.

The PC used to access the Professional Web Tool (PWT) will be a developer PC, which most likely will differ greatly from the PCs used in the final integration environment. Since access to the PWT is through a Web Browser it will most likely not have any impact, but it is still worth taking into account.

The Android Phones used are owned by UPV-SABIEN. The model will be a Motorola Moto G4, but UPV-SABIEN will attempt to try as many other models as possible.

The sensor devices are the same models to be used in the field. Weight scales: A&D UC 352BLE, A&D UC 321PBT. Blood pressure sensors: A&D UA 651BLE, A&D UA 767PBT. Wrist band: Xiaomi Band 2.

The following scheme shows the deployment of main software subsystems:

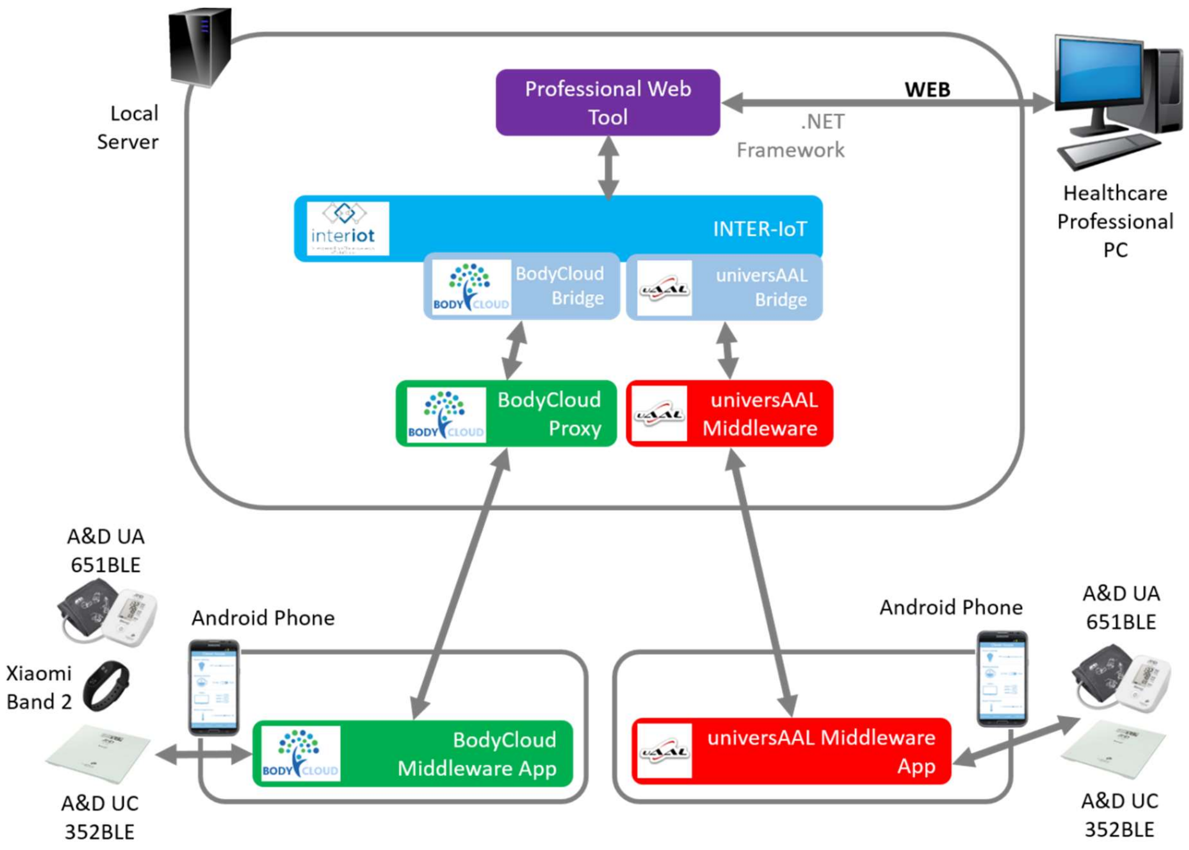


Figure 12: INTER-Health Software overview

The virtual machine at the Local Server runs Windows Server 2016 OS with nested virtualization capabilities. The following table shows the installed software components that form each of the above subsystems inside the Local Server. Some have to be run on top of a container software component that is also installed:

Component	Subsystem	Version	Container
INTERMW	INTER-IoT	0.0.1-SNAPSHOT	Apache Tomcat 8.5.23, Java JDK 8
RabbitMQ	INTER-IoT	-	Docker
WSO2 API Manager	INTER-IoT	2.1.0	Docker
Kafka	INTER-IoT	0.11.0.1	Docker
IPSM	INTER-IoT	0.3.0.0	Docker
Parliament DB	INTER-IoT	-	Docker
Professional Web Tool Application	Professional Web Tool	1.0.0	IIS, .NET Framework 4.7
MS SQL Server	Professional Web Tool	2014 Version 12.0.5204.0	Native (Windows)

MySQL	BodyCloud Proxy	15.1	XAMPP v7.1.10
BodyCloud Proxy	BodyCloud Proxy	0.0.1-SNAPSHOT	Apache Tomcat 8.5.23
UniversAAL server	UniversAAL Middleware	3.4.1-SNAPSHOT	Karaf OSGi 3.0.8, Java JDK 8

Table 7: Local Server software components

The Android phones used will run versions 6.0 to 7.0 of Android. They will run either one of the applications in the following table. No phone will run both applications at the same time. This accounts for representing patients' phones and doctors' phones, which accomplish different use cases.

Application	Vesion
BodyCloud INTER-Health App	0.0.1-SNAPSHOT
UniversAAL INTER-Health App	0.0.1-SNAPSHOT

Table 8: Android Phones applications

The browsers used by the HealthCare Professional PC (role played by a UPV-SABIEN developer PC) will be the same that will be used in the field deployment:

Application	Vesion
Mozilla Firefox	52.2
Google Chrome (if needed)	Latest at the time of testing

Table 9: Healthcare Professional browsers

3.2.2 Integration of IoT framework

As depicted above, INTER-IoT, in particular INTER-Framework, is used to allow the Professional Web Tool to access data from two different IoT platforms. For this reason, only the INTER-Middleware (or MW2MW) layer of INTER-Layer is of interest for the pilot.

All modules required to run INTER-Middleware are installed in the server, as listed above, but no other layers are required. The goal is that the Professional Web Tool uses INTER-API to access the required data through INTER-Framework, but in the FAT it will use a mix of API and some hardcoded values, for simplicity and speed of development.

The following table lists the components of INTER-IoT that are installed in the FAT and, if they do, how they interface with external components:

INTER-IoT Component	Interface	With component
INTER-Middleware (BodyCloud Bridge)	BodyCloud's local REST-API and callback	BodyCloud Proxy
INTER-Middleware (universAAL Bridge)	universAAL's local REST-API and callback	universAAL Middleware
INTER-API	INTER-IoT local REST-API	Professional Web tool

Table 10: Integration components.

3.2.3 Deliverables and version overview

The following table contains a deliverable list which need to be signed of before FAT testing commences.

ID	Description	Check
Documents		
1	Validation and Test reports of the IoT system components	
2	Validation and Test reports of the Professional Web Tool	
3	Validation and Test reports of universAAL	
4	Validation and Test reports of BodyCloud	
Hardware		
4	Local Server	
5	Healthcare Professional PC	
6	Android Phones	
Tools		
7	MS Windows Remote Desktop	

Table 11: Deliverable checklist

The following table shows the software components and version of which the system release version consists of.

ID	Description	Version	Check
INTER-IoT			
1	Java JDK	1.8	
2	Apache Tomcat	8.5.23	
3	Docker	17.09.0-ce	
4	Parliament DB		
5	IPSM	0.3.0.0	
6	Kafka	0.11.0.1	
7	WSO2 API Manager	2.1.0	
8	RabbitMQ		
9	INTER-MW	0.0.1-SNAPSHOT	
Professional Web Tool			
10	MS SQL Server	12.0.5204.0	
11	.NET Framework	4.7	
12	.NET Core Windows Server Hosting	1.0.4 & 1.1.1	
13	Professional Web Tool Application	1.0.0	
BodyCloud			
15	XAMPP	7.1.10	
16	MySQL	15.1	
17	BodyCloud Proxy	0.0.1-SNAPSHOT	
UniversAAL			
18	Karaf OSGi	3.0.8	
19	UniversAAL Server (Middleware + REST API)	3.4.1-SNAPSHOT	

Table 12: Component version overview

3.2.4 Requirements, scenarios and use cases

The following table provides an overview of the relation between the requirements and the test(s) that validate their implementation.

ID	Description	Covered by
62	Constraints on processing of personal and health data	T1.1.1, T1.1.2, T1.1.3, T1.2.1, T1.3.1, T1.3.2, T1.3.3, T1.4.1, T1.4.2
71	Application response time	T1.1.1, T1.1.2, T1.1.3, T1.2.1, T1.3.1, T1.3.2, T1.3.3, T1.4.1, T1.4.2
101	Exchanging discrete medical measures across platforms	T1.3.1, T1.3.2, T1.3.3
102	Exchanging complex medical measures across platforms	T1.3.1, T1.3.2
103	User Authentication to access INTER-Health services	T1.1.1, T1.1.2, T1.1.3, T1.2.1, T1.3.1, T1.3.2, T1.3.3, T1.4.1, T1.4.2
104	Personal data and user profile management	T1.1.1, T1.1.2, T1.1.3, T1.2.1, T1.3.1, T1.3.2, T1.3.3, T1.4.1, T1.4.2
106	Definition of reference meaning for health information	(Theoretical, not testable practically)
107	Exchanging synthetic or statistical health information between platforms	T1.3.1, T1.3.2, T1.3.3
127	Availability of sensor data	T1.3.1, T1.3.2
145	Informed consent. Processing of personal data	T1.1.1, T1.1.2, T1.2.1
146	Information sheet. Processing of personal data	T1.1.1, T1.1.2, T1.2.1
157	Seamless patient monitoring	T1.3.1, T1.3.2, T1.3.3, T1.4.1, T1.4.2
158	National, regional and local Bioethic Committee	T1.1.1, T1.1.2, T1.2.1
164	Medical Device informatics	T1.3.1, T1.3.2
172	User Access Service for Patients	T1.1.3, T1.3.2, T1.3.3
173	User Access Service for Doctors	T1.1.1, T1.1.2, T1.2.1, T1.3.1, T1.4.1, T1.4.2
174	User Access Service for Administrators	T1.1.1, T1.1.2, T1.2.1
176	User Access Gateway for Patients	T1.1.3, T1.3.2
177	User Access Gateway for Caregivers	T1.3.1
188	Integration with legacy systems	T1.1.1, T1.1.2, T1.2.1, T1.4.1, T1.4.2
217	Wearable devices support	T1.3.2
218	Personal data collected on Computerized Nutritional Folder	T1.2.1, T1.3.3, T1.4.2

Table 13: Requirements vs test mapping

The following table provides an overview of the relation between the scenarios and the test(s) that validate their implementation.

ID	Scenario name	Covered by
1	Chronic disease prevention	All
11	Primary prevention of cognitive decline	
12	Health failure disease and mild Alzheimer disease	
15	Surveillance systems for prevention programs	All
16	Elderly monitoring	All
21	Low risk of developing chronic diseases.	All

22	Increased risk of developing chronic diseases	All
23	High risk of developing chronic diseases	All
24	Very high risk of developing chronic diseases	All
25	Extremely high risk of developing chronic diseases	All
27	Vitamins intake analyser	
28	Calories / nutrition mixer / cookware counter	

Table 14: Scenario vs test mapping

3.2.5 Test environment

Introduction

To test the functionality of the INTER-Health in combination with the IoT framework a representative test system is needed. The test system needs to approach the “real world” as much as possible. The pilot setups must be recreated and proven. This chapter will describe this environment and the used hardware, software, tools and platforms.

Test environment

For details, refer to system description in section 3. The environment for the test comprises:

The server machine of UPV-SABIEN. This computer is located in the premises of UPV-SABIEN, in a separate server room with restricted access allowed only to UPV-SABIEN members.

The local network used in the test that stands in for the Healthcare center that will be used in the real deployment is that of the UPV. It is managed by UPV staff, not UPV-SABIEN directly, but the proper sub-networks, firewalls and other nodes have already been configured to simulate, to a certain extent the pilot environment.

The computers emulating the Healthcare professional’s desktops can be any of the ones used by UPV-SABIEN staff while performing the tests. They are all Windows 10 Enterprise edition with both Firefox and Chrome installed to do the end-user tests of the Professional Web Tool.

All access to tools, hooks and probes are performed through those same computers.

The mobile phones used are pre-existing development phones at UPV-SABIEN. The model is Motorola Moto G4 running Android 6.0 or 7.0. It is possible however that during the course of the test other phone models are tested, if available.

The sensor device models used in the test will be the same as those used by patients: A&D UA 651BLE and A&D UC 352BLE, and Xiaomi Band 2.

The role of patients and healthcare providers will be performed indistinctively by any of the UPV-SABIEN staff at convenience.

3.2.6 Test setups, tools, hooks and probes

This paragraph describes the test setups, tools, hooks and probes used in the by this document defined tests.

A test setup defines a setup used for a test, and describe the used system parts, interface etc.

Tools define the used COTS tooling that will be used during testing e.g. Logic analyzers, packet sniffer like e.g. Wireshark, etc.

Hooks are used to inject data or control parts of the system used to get the system to handle a needed scenario. This can be hidden, debug or service interfaces which contain the needed functionality.

Probes are used to gather logging from parts of the system under test to provide system feedback and to prove that the system is handling the scenario as it should.

The following paragraphs will define the used setups, tools, hooks and probes which will be referred to by the test descriptions.

TS_01 Overall test setup

This generic setup is used in all tests: A UPV-SABIEN tester plays the role of patient and has access to one (or several, depending on the success of the tests) mobile phones, already paired to each of the available sensor devices. The tester interfaces directly with the sensors to make measurements, and with the applications in the phone.

Another tester, playing the role of Healthcare professional, has access, through his/her computer, to the Professional Web Tool.

Finally, another tester will act as observer and accesses the server through remote desktop to monitor the running programs and obtain the required results.

TT_01 Microsoft Windows Remote Desktop

Access to the software running in the server is performed through Remote Desktop. UPV-SABIEN testers have access to this server and monitor it in real time while the tests are performed, and obtain all necessary outputs.

TH_01 INTER-IoT Message Emulators

There are emulators that publish sensor data as if it came from either BodyCloud or universAAL. These are not officially part of any test, but may be used in case of issues with real data, in order to compare as a baseline or to test the overall system when needed.

TP_01 IoT Platforms consoles

Both universAAL and BodyCloud are installed in the system server. Some of their outputs can be observed in real time through their consoles. This can help identify crashes or problems as they happen.

TP_02 IoT Platforms output logs

Both universAAL and BodyCloud are installed in the system server. They generate output log files stored in the server that can be obtained after running, or even while the system is running, depending on the log editor used. This can help identify crashes or problems after they happen.

TP_03 Android Studio

Console output and logs from the mobile phones can be accessed through Android Studio. In order to enable this, the phones used in the tests have to enable their “developer options”, connect to a PC (of one of UPV-SABIEN testers) through USB and allow debugging.

3.2.7 Test description

3.2.7.1 U1 – Creates and operates users/services

T1.1.1 Professional creates user

ID	T1.1.1
Test	The Healthcare Professional enters the system and creates a new patient user
Type	Manual test
Setup	TS_01, TT_01
Start	First time run, empty records, Patient has been given phone and documentation
Req.	[62], [71], [103], [104], [145], [146], [158], [173], [174]
Input	Healthcare Prof. enters PWT, authenticates. Healthcare Prof. enters option to create new patient Healthcare Prof. enters patient data and confirms creation
Output	The new Patient is registered in the system The Patient can now use the system as intended
Logs	<Define where the output log is stored, e.g.: Folder "`Tx_Output`", prefix "`Tx.y.1_logname`" >
Outcome	Pass / Fail

T1.1.2 Professional modifies user

ID	T1.1.2
Test	The Healthcare Professional enters the system and updates a patient's data
Type	Manual test
Setup	TS_01, TT_01
Start	Patient was already created in the system, Patient has been given phone and documentation
Req.	[62], [71], [103], [104], [145], [146], [158], [173], [174]
Input	Healthcare Prof. enters PWT, authenticates. Healthcare Prof. enters option to update patient Healthcare Prof. enters new patient data and confirms update
Output	The Patient's new data is registered in the system The Patient can continue to use the system as usual
Logs	<Define where the output log is stored, e.g.: Folder "`Tx_Output`", prefix "`Tx.y.1_logname`" >
Outcome	Pass / Fail

T1.1.3 Patient logs with their profile

ID	T1.1.3
Test	Patient enters the system through his/her mobile phone apps to access data
Type	Manual test
Setup	TS_01, TT_01, TP_01, TP_02, TP_03
Start	Patient is in possession of mobile phone with installed app Patient has been registered in the system with proper data
Req.	[62], [71], [103], [104], [172], [176]
Input	Patient enters BC app, authenticates, checks data
Output	Patient successfully accesses app Patient can check up-to-date data
Logs	<Define where the output log is stored, e.g.: Folder "`Tx_Output`", prefix "`Tx.y.1_logname`">
Outcome	Pass / Fail

3.2.7.2 U2 – Set patient protocol parameters

T1.2.1 Professional sets protocol

ID	T1.2.1
Test	The Healthcare Professional enters the system and updates a patient's protocol
Type	Manual test
Setup	TS_01, TT_01
Start	Patient was already created in the system, Patient has been given phone and documentation
Req.	[62], [71], [103], [104], [145], [146], [158], [173], [174], [218]
Input	Healthcare Prof. enters PWT, authenticates. Healthcare Prof. enters option to update patient Healthcare Prof. enters new patient protocol and confirms update
Output	The Patient's new data is registered in the system The Patient can continue to use the system as usual, according to new protocol
Logs	<Define where the output log is stored, e.g.: Folder "`Tx_Output`", prefix "`Tx.y.1_logname`">

Outcome	Pass / Fail

3.2.7.3 U3 – Perform objective and subjective measures

T1.3.1 Professional collects measures (objective)

ID	T1.3.1
Test	Healthcare Professional takes measures from Patient at centre using devices
Type	Manual test
Setup	TS_01, TT_01, TH_01, TP_01, TP_02, TP_03
Start	Patient has been registered in the system with proper data Healthcare prof. is in possession of mobile phone with installed app Sensor devices are paired to mobile phone
Req.	[62], [71], [101], [102], [103], [104], [107], [127], [157], [164], [173], [177]
Input	Healthcare Prof. enters PWT, authenticates. Healthcare Prof. enters option to update patient measures Healthcare Prof. enters universAAL app, authenticates Patient takes measurement on centre sensor device
Output	Patient measure appears on uAAL app and PWT, allowing Healthcare Prof. to update patient measure data
Logs	<Define where the output log is stored, e.g.: Folder "`Tx_Output`", prefix "`Tx.y.1_logname`" >
Outcome	Pass / Fail

T1.3.2 Patient performs measures (objective)

ID	T1.3.2
Test	Patient takes measures at home using devices
Type	Manual test
Setup	TS_01, TT_01, TH_01, TP_01, TP_02, TP_03
Start	Patient has been registered in the system with proper data Patient is in possession of mobile phone with installed app Sensor devices are paired to mobile phone
Req.	[62], [71], [101], [102], [103], [104], [107], [127], [157], [164], [172], [176], [217]
Input	Patient successfully accesses app

	Patient takes measurement on home sensor device
Output	The measure is registered in the system at the Healthcare centre and can be checked by Healthcare Prof. in PWT.
Logs	<Define where the output log is stored, e.g.: Folder "`Tx_Output`", prefix "`Tx.y.1_logname`">
Outcome	Pass / Fail

T1.3.3 Patient performs measures (subjective)

ID	**T1.3.3**
Test	Patient answers questionnaire about habits
Type	Manual test
Setup	TS_01, TT_01, TP_01, TP_02, TP_03
Start	Patient has been registered in the system with proper data Patient is in possession of mobile phone with installed app Healthcare Prof. has set protocol
Req.	[62], [71], [101], [102], [103], [104], [107], [157], [172], [218]
Input	App notifies Patient about questionnaire Patient successfully accesses app Patient takes questionnaire
Output	The measure is registered in the system at the Healthcare centre and can be checked by Healthcare Prof. in PWT.
Logs	<Define where the output log is stored, e.g.: Folder "`Tx_Output`", prefix "`Tx.y.1_logname`">
Outcome	Pass / Fail

3.2.7.4 U4 – Monitors subjective and objective parameters

T1.4.1 Professional monitors parameters (objective)

ID	**T1.4.1**
Test	Healthcare Professional accesses Patient data recorded through sensors
Type	Manual test
Setup	TS_01, TT_01, TH_01, TP_01, TP_02
Start	Patient has been registered in the system with proper data Patient has recorded objective measures

	Healthcare prof. has recorded objective measures of patient
Req.	[61], [71], [103], [104], [157], [173]
Input	Healthcare Prof. enters PWT, authenticates. Healthcare Prof. enters option to observe patient measures
Output	The PWT displays the measures taken with the sensors
Logs	<Define where the output log is stored, e.g.: Folder "`Tx_Output`", prefix "`Tx.y.1_logname`">
Outcome	Pass / Fail

T1.4.2 Professional monitors parameters (subjective)

ID	T1.4.2
Test	Healthcare Professional accesses Patient data recorded through questionnaires
Type	Manual test
Setup	TS_01, TT_01, TH_01, TP_01, TP_02
Start	Patient has been registered in the system with proper data Patient has recorded objective measures Healthcare prof. has recorded objective measures of patient
Req.	[61], [71], [103], [104], [157], [173], [218]
Input	Healthcare Prof. enters PWT, authenticates. Healthcare Prof. enters option to observe patient measures
Output	The PWT displays the measures taken with the questionnaires
Logs	<Define where the output log is stored, e.g.: Folder "`Tx_Output`", prefix "`Tx.y.1_logname`">
Outcome	Pass / Fail

3.2.8 Test outcome

The following table will provide an overview of the test result of all the performed tests in this FAT.

Test	Description	Outcome
T1.1.1	Professional creates user	Pass / Fail
T1.1.2	Professional modifies user	Pass / Fail
T1.1.3	Patient logs with their profile	Pass / Fail

T1.2.1	Professional sets protocol	Pass / Fail
T1.3.1	Professional collects measures (objective)	Pass / Fail
T1.3.2	Patient performs measures (objective)	Pass / Fail
T1.3.3	Patient performs measures (subjective)	Pass / Fail
T1.4.1	Professional monitors parameters (objective)	Pass / Fail
T1.4.2	Professional monitors parameters (subjective)	Pass / Fail
FAT Outcome		**Pass / Fail**

Table 15: Test outcome overview

3.2.9 Integration ethics and security

Introduction

During the Ethical Review it was requested that ethics had to be included within the project scope. This section addresses recommendations and proposed solutions to the ethical and security points of the integration.

For each pilot the ethics is discussed in paragraphs 8.2 until 8.5. The security aspects of each layer is discussed in paragraph 8.7 and 8.8.

The information for the pilots for both ethics and security comes from the partners and may be included in other documents as well.

INTER-Health

Regarding ethics, this FAT procedure does not require additional restrictions over what is already defined in the overall project. The stand-ins for patients are UPV-SABIEN personnel, and the data they will input in the system is of either one of three types:

- Simulated data: does not require additional security or ethics considerations
- Fake or spoofed data from sensors, which does not represent actual personnel data: does not require additional security or ethics considerations
- Real data from sensors: In this case the data would be subject to similar considerations as real patient data, but the entire FAT setup is secured in an equivalent level to that of the final pilot, and in addition to this, all data generated during the tests is removed after verification of the tests success or failure. The personnel at UPV-SABIEN are confirmed to not have any health condition that could affect the test outcomes.

Regarding security, the FAT setup is similarly protected as the pilot deployment. The local network can only be accessed by UPV personnel, and the FAT server can only be accessed by UPV-SABIEN personnel (and other INTER-IoT partners upon request, with appropriate security measures).

3.3 Open Call FAT's

3.3.1 Third Party: SensiNact

The sensiNact Gateway allows interconnection of different networks to achieve access and communication with embedded devices and/or cloud-based services. It is composed of a set of functional groups and their relative interfaces; both can be seen in the Figure 13 Below it is a non-exhaustive list of the components present in the platform.

- The *Device Protocol Adapter* abstracts the specific connectivity technology of wireless sensor networks. It is composed of the bridges associated to protocol stacks. All the bridges comply with a generic Device Access API used to interact with northbound sensiNact's services.
- The *Smart Object Access and Control* implements the core functionalities of sensiNact like discovering devices, resources and securing communication among devices services and their consumers.
- The *Consumer API* is protocol agnostic and exposes services of the Smart Object Access and Control functional to Consumers.
- The *Consumer Protocol Adapter* consists of a set of protocol bridges, translating the Consumer API interface into specific application protocols.
- The *Gateway Management* functional group includes all the components needed to ease management of devices connected to sensiNact, regardless of their underlying technologies. A Device Management API is used for this purpose. This functional group also contains the components managing cache, resource directory and security services. These management features are exposed by means of the *Manager API*.
- *Manager Protocol Adapter* allows adapting the *Gateway Management API* to the specific protocols used by different external management entities.

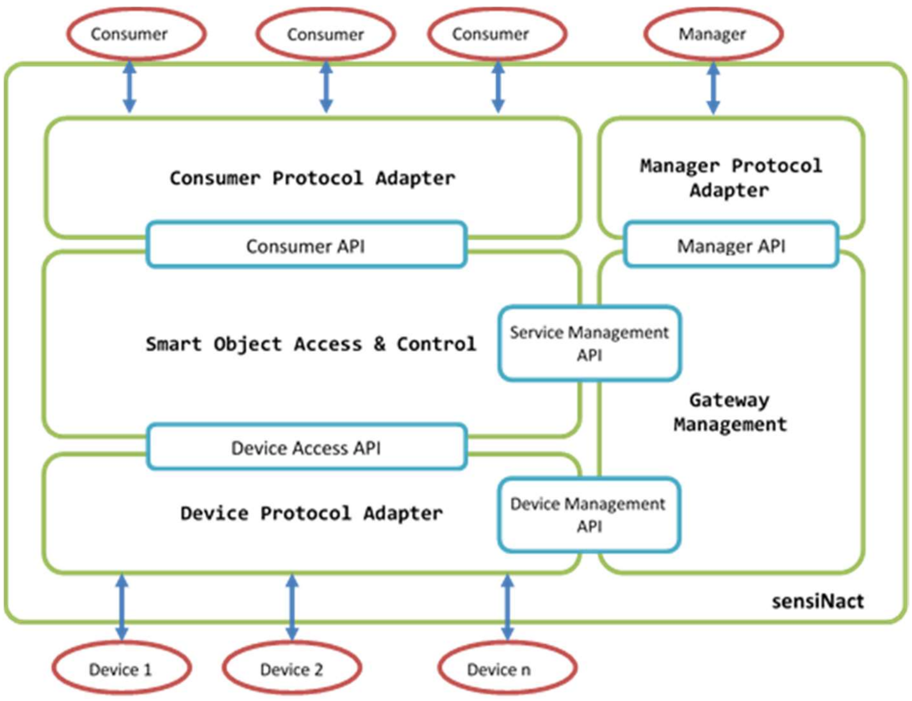


Figure 13: SensiNact Gateway overall architecture.

In terms of connectivity:

On the southbound side sensiNact gateway allows to cope with *physical device* protocols and *virtual device*, allowing a uniform and transparent access to given protocol, for example ZigBee network, HTTP Restful web service. Below it is a list of supported protocols:

- **EnOcean**, concerting energy harvesting wireless sensor technology (ultra-low-power radio technology for free wireless sensors), and protocols in use to interact with those sensors;
- **BLE**, Bluetooth Low Energy , which is a WPAN, low power protocol designed mainly for healthcare or entertainment applications;
- **MQTT**, which is a machine-to-machine protocol, lightweight publish/subscribe messaging transport, useful for connections with remote locations where a small code footprint is required and/or network bandwidth is at a premium;
- **ZigBee** radio communication protocol for low consumption short range designed for WPAN (XBee for example);
- **CoAP** which is REST application protocol, designed to be “the HTTP for constrained networks and devices” whose concept originated from the idea that "the Internet Protocol could and should be applied even to the smallest devices," and that low-power devices with limited processing capabilities should be able to participate in the Internet of Things; it is usually used on top of a 6LoWPAN network, but it may travel regular IP networks as well (it is used by the OMA LWM2M protocol, for instance);
- **EchoNet,** Japanese communication protocol designed to create the “smart houses” of the future. Today, with Wi-Fi and other wireless networks readily available in ordinary homes, there is a growing demand for air-conditioning, lighting and other equipment inside the home to be controlled using smartphones or controllers, or for electricity usage to be monitored in order to avoid wasting energy.

On the northbound side the sensiNact gateway provides both client/server and publish/subscribe access protocols:

- **MQTT**;
- **JSON-RPC** (1.0 and 2.0);
- **HTTP RESTful**
- **CDMI**

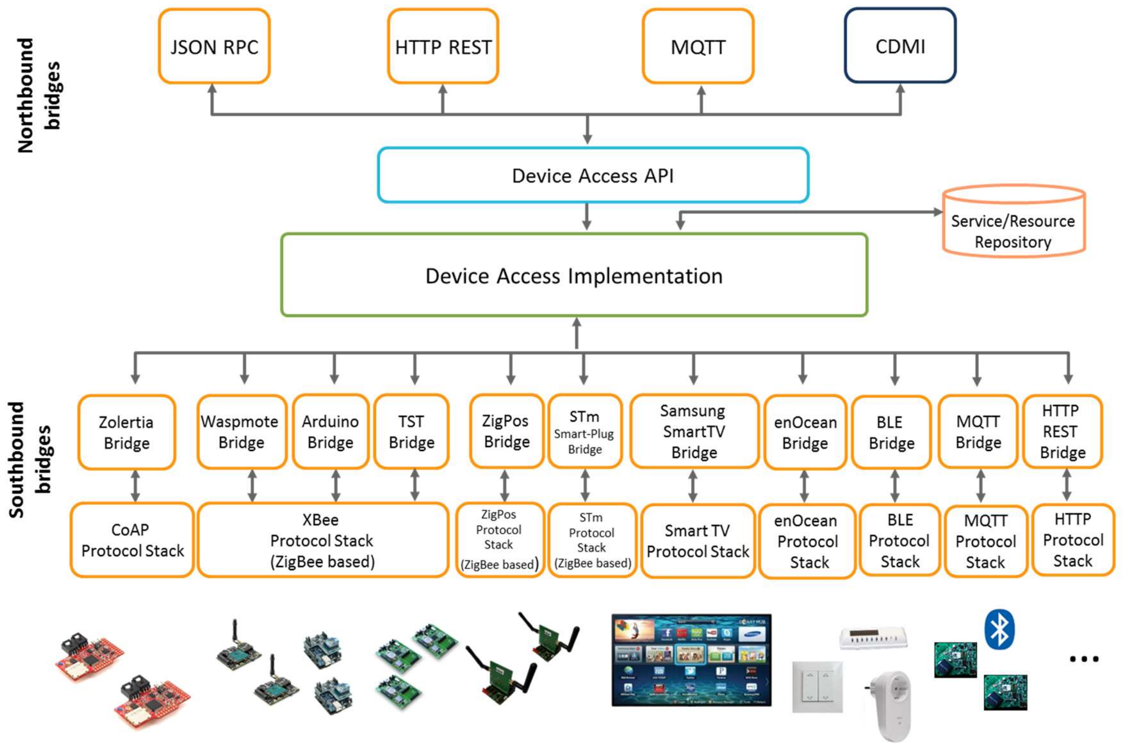


Figure 14 SensiNact Southbound and Northbound bridges

The **Smart Object Access and Control** functional group described in this previous section includes a large number of functionalities, among them:

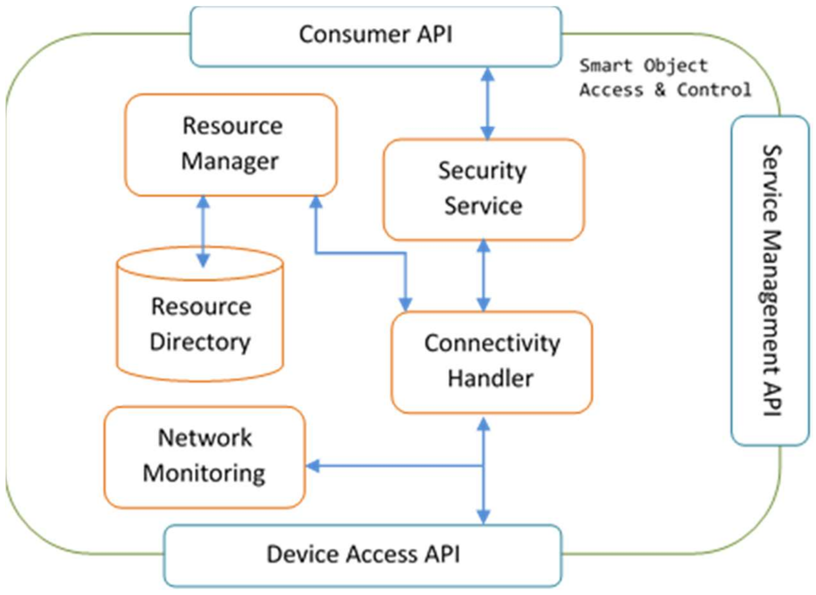


Figure 15 SensiNact Gateway internal architecture

- It handles the communication with the Consumer Protocol Adapter (REST API, JSON RPC, etc.) and IoT (and non-IoT) devices, providing URI mapping, incoming data/messages translation in an internal format and outgoing data/messages translation in Consumer format. Whenever a Consumer tries to access a resource via Consumer API, the requested URI is forwarded to the Resource Manager in order to check if a specific resource descriptor exists or not inside the Resource Directory and to verify its accessibility status. If a resource descriptor doesn't exist, a message response with error code is returned to the Consumer API. Otherwise, the request is forwarded to the right interface. At the same time whenever response is originated from IoT device (or abstract IoT device), it will be also forwarded to its logical counterpart in order to update the resource representation in the gateway.

- It manages the subscription/notification phases towards the Consumer, if it is not handled by the targeted device (service) itself
- It supports Devices and Resource Discovery and Resource Management capabilities, to keep track of IoT Resource descriptions that reflect those resources that are reachable via the gateway. These can be both IoT Resources, or resources hosted by legacy devices that are exposed as abstracted IoT Resources. Moreover, resources can be hosted on the gateway itself. The Resource Management functionality enables to publish resources in sensiNact, and also for the Consumer to discover what resources are actually available from the gateway; sensiNact Service and Resource model allows exposing the resources provided by an individual service. The latter, characterized by a service identifier, represents a concrete physical device or a logical entity not directly bound to any device. Each service exposes resources and could use resources provided by other services. Figure 16 below depicts the Service and Resource model:

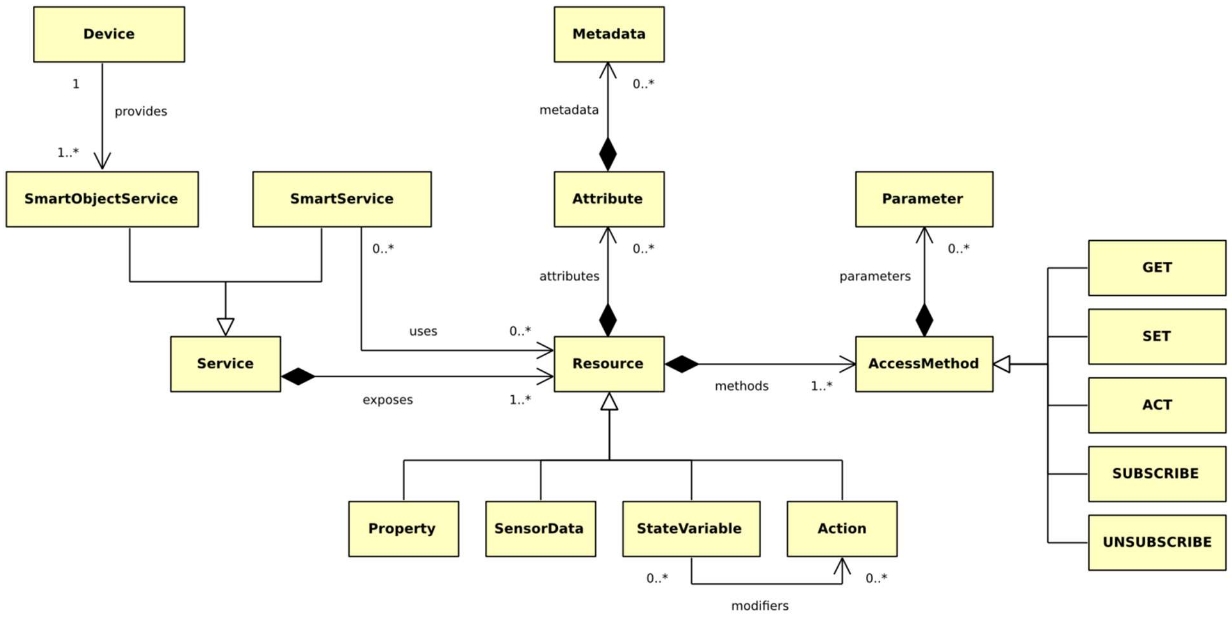


Figure 16 SensiNact Service and Resource model

The **Resource Directory** allows storing information, *i.e.* resource descriptions, about the resources provided by individual devices connected to sensiNact. It also supports resource description look up, as well as publishing, updating and removing resource descriptions.

Discovering and using resources exposed by Services is the preferred approach for avoiding using static service interfaces, thus increasing interoperability. Therefore, sensiNact Services and their exposed resources are registered into Service/Resource repositories. The gateway uses the OSGi service registry as Service/Resource repository, where resources are registered as service properties. Clients ask the Service/Resource repository for resources fulfilling a set of specified properties (defined by LDAP filters). In response, the Service/Resource repository sends clients the list of service references that expose the requested and authorized resources. Clients can then access/manipulate the resources exposed by their selected service objects.

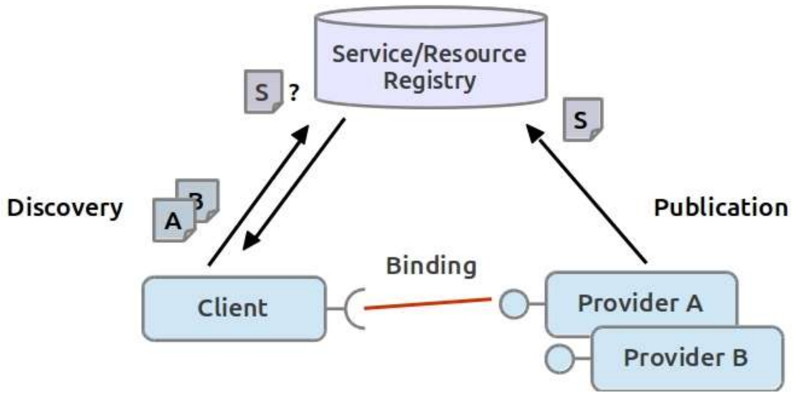


Figure 17 sensiNact's service oriented approach

Resources and services can be available for remote discovery and access using different communication protocols, such as HTTP REST, JSON-RPC, etc. Advanced features may also be supported (as semantic-based lookup). Resources can be classified as shown in Table 16 while the access methods are described in Table 17.

Type	Description
SensorData	Sensory data provided by a service. This is real-time information provided, for example, by the SmartObject that measures physical quantities.
Action	Functionality provided by a service. This is mostly an action on the physical environment via a SmartObject actuator supporting this functionality (turn on light, open door, etc.) but can also be a request to do a virtual action (play a multimedia on a TV, make a parking space reservation, etc.)
StateVariable	Information representing a SmartObject state variable of the service. This variable is most likely to be modified by an action (turn on light modifies the light state, opening door changes the door state, etc.) but also to intrinsic conditions associated to the working procedure of the service
Property	Property exposed by a service. This is information which is likely to be static (owner, model, vendor, static location, etc.). In some cases, this property can be allowed to be modified.

Table 16: Resource types.

Type	Description
GET	Get the value attribute of the resource

SET	Sets a given new value as the data value of the resource
ACT	Invokes the resource (method execution) with a set of defined parameters
SUBSCRIBE	Subscribes to the resource with optional condition and periodicity
UNSUBSCRIBE	Remove an existing subscription

Table 17: Resource's access methods.

The access methods that can be associated to a resource depend on the resource type, for example, a GET method can only be associated to resources of type Property, StateVariable and SensorData. A SET method can only be associated to StateVariable and modifiable Property resources. An ACT method can only be associated to Action resources. SUBSCRIBE and UNSUBSCRIBE methods can be associated to any resource type.

3.3.1.1 Protocols and data formats

The SensiNact gateway uses JSON formatted data. The resource model is a hierarchical five-tiered tree: A Device owns Services which in turn own Resources, which hold Attributes, and its *metadata*. To describe one element of this tree there is no restriction about how many sub-elements it can contain. The description of a resource and its value (result of an access method execution) are distinct from one to the other. The choice of this separation is to lighten the work of components whose work is to process the result of an access method execution, by avoiding the reification of high level data structures to only extract the content of one (or two) attribute(s).

Device Description

As only the resources are the containers of information, those which target the device are grouped in a specific service which is the administration one (« AdminService » prefixed). Those resources can be one specifying the location, or the vendor of the device, or any other data that are common to all provided services (and so resources). Formally, a device is a JSON object containing an array of services. The list of the services a device provides can be summarized or detailed. If it is summarized, only the name of the services are part of the description (otherwise each service is completely described).

```
{
    "serial-number":"fake-1234",
    "services":[
        {
            "ID":"AdminService_f1To4"
        },
        {
            "ID":"temperature_f1To4"
        }
    ]
}
```

Service Description

It gathers resources, and it references the unique identifier of the device holding it. It represents the entry point to access to resources through the OSGi context. The list of the resources a service provides can be summarized or detailed. If it is summarized only the name and the type of the resource are part of the description (otherwise each resource is completely described).

```
{
   "ID":"AdminService_f1To4",
   "properties":[
      { "device.serial-number":"fake-1234"}
   ],
   "resources":[
      {"name":"location","type":"property"},
      {"name":"owner","type":"property"},
      {"name":"vendor","type":"property"},
      {"name":"SLEEP","type":"action"}
   ]
}
```

Resource description

- The data structures are mainly nested in triplets : name, type and value;
- The type of the resource itself can be : property, variable, sensor, or action;
- The type key of a 'name-type-value' data structure (embedded in the resource description) can have a primitive as value (byte, short, int, long, double, char, boolean, [string]) or the canonical name of the java class used to reify it in the gateway;
- For each resource access method signatures are also described in a JSON array. Some of them can be shortcuts to other ones: a GET method without parameter is a shortcut to the GET method whose unique parameter "attributeName" has for value "value", for example. A parameter of an access method can be completed with the constraints which apply on it (« min », « max », « fixed », regular expression « pattern », « enumeration » of allowed values) or the JSON schema of the expected JSON object from which to reify the appropriate Java object in the gateway;
- At least two metadata exist for each attribute: the "hidden" one defining whether the attribute has to be specified in the description of the resource, and the "modifiable" one defining whether the value of the attribute can be modified by the client. By default, the « hidden » attribute is not visible in the description (if the attribute is visible that's mean that this metadata value is set to false, and if it is set to true the client is, at the end, not aware of that);
- A metadata specified as « dynamic » will be added to the result of an access method execution
- Timestamps are « epoch » formated (number of seconds since 1970 January 1st); to avoid the reification of high level objects to make calculations (that are at least as easy with this format). High level programming languages handle this format. It is also possible to multiply it by 1000 if handling of milliseconds is needed (what is done natively by java for example).

```
{
“name":"temperature",
    "type":"sensor",
    "attributes":[
        {  "name":"value","type":"int",
          "metadata":[
                {"name":"modifiable",“type":"boolean","value":false,"dynamic":fal
se},
                {"name":"timestamp","type":"long","value":1418541626,"dynamic":tr
ue},
                { "name":"description","type":"string","value":"temperature measu
re","dynamic":false},
                {  "name":"unit","type":"string","value":"celsius degree","dynami
c":false}
            ]
        }
    ],
    "accessMethods":[
        {
            "name":"GET",
            "parameters":[
            ]   },
        {
            "name":"GET",
            "parameters":[
                { "name":"attributeName","type":"string"}
            ]   },
        {
            "name":"SUBSCRIBE",
            "parameters":[
                {  "name":"listener","type":"object",
                    "schema-id":"http://fr.cea.sensinact/subscription/listener",
                    "description":"parameter value example: '{\"callback\":\"<uri
>\"}' "
                },
                {
                    "name":"condition",
                    "type":"object",
                    "schema-id":"http://fr.cea.sensinact/subscription/condition",
                    "description":"parameter value example: '{ \"condition\":\"le
ss\", \"value\":\"5\"}'"
                },
                {
                    "name":"lifetime",
                    "type":"long"
                }
            ]
        },
        {
            "name":"SUBSCRIBE",
            "parameters":[
                {
                    "name":"listener",
                    "type":"object",
                    "schema-id":"http://fr.cea.sensinact/subscription/listener",
                    "description":"parameter value example: '{\"callback\":\"<uri
>\"}' "
                }  ]  },
        {
            "name":"UNSUBSCRIBE",
            "parameters":[
                {
                    "name":"subscriptionID",
                    "type":"string"
                }   ]   }
    ]  }
```

Access Method result

```
{
    "name":"temperature",
    "type":"int",
    "value":22,
    "timestamp":1418541626
}
```

As it is the « default » attribute, asking for the value of a resource providing one returns a JSON formatted data structure in which the "name" key has the name of the resource as value (instead of "value" as it could have been expected).

Location resource

The location of a device (service, resource) is frequently a needed context information. By default a device always contains one (its administration service in fact), and a link to it is created in all services it provides. If needed, a link to this resource could also be created as an attribute of all resources (mainly if this location is supposed to change frequently and so to avoid to require the complete device description to update the information). Its content is not restricted (as it is the case for the others) and can so contain attributes defining longitude, latitude, altitude, a friendly name or whatever is needed to specify it (for now we are using « <latitude>,<longitude> » formatted string as value)

3.3.1.2 Device Access Control

This section explains the security definition and method into sensiNact architecture.

Security and access policies

A first level of security is reached by the way of some of available security "tools" in the *OSGi* environment: *ServicePermission* and *ConditionalPermissionAdmin.*

The ServicePermission is a module's authority to register or use a service.

- The register action allows a module to register a service on the specified names.
- The get action allows a module to detect a service and use it.

Permission to use a service is required in order to detect events regarding the service. Untrusted modules should not be able to detect the presence of certain services unless they have the appropriate *ServicePermission* to use the specific one.

The *ConditionalPermissionAdmin* is framework service to administer conditional permissions that can be added to, retrieved from, and removed from the framework.

The sensiNact gateway defines service permissions in such a way that access to the ones it provides is forbidden excepted if a specific condition is met (a sensiNact specific conditional permission). This condition being that the requirer is the sensiNact *SecuredAccess* service. Even sensiNact services have to use the *SecuredAccess* one to be able to “talk” to each other’s; Modalities of such exchanges depend on the UserProfile of the user of these services (the user can be the system itself). A *UserProfile* can be defined at each level of the hierarchical sensiNact resource model: *ServiceProvider*, Service, and Resource. Five *UserProfiles* exist for which predefined access rights are defined: Owner, Administrator, Authenticated, Anonymous, and Unauthorized.

When asking for a data structure of the sensiNact resource model, the access rights of the user are retrieved; the set of this user's accessible *AccessMethods* for the specific data structure is built and returned as part of the description object. Each future potential interaction of the user on the data structure will be made by the way of this description object. For a remote access a security token is also generated and transmitted to the user, to avoid repeating the security policy processing. A token is defined for a user and a data structure (and so it previously created description object).

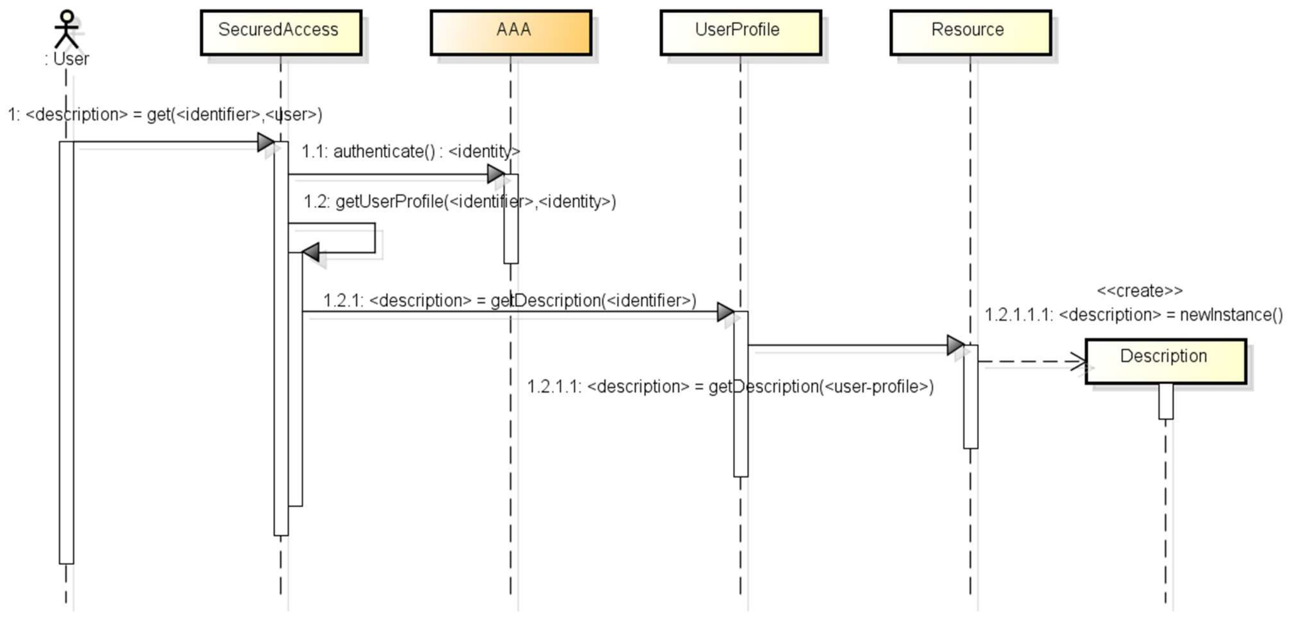

: User
SecuredAccess
AAA
UserProfile
Resource
1: <description> = get(<identifier>,<user>)
1.1: authenticate() : <identity>
1.2: getUserProfile(<identifier>,<identity>)
1.2.1: <description> = getDescription(<identifier>)
<<create>>
1.2.1.1.1: <description> = newInstance()
Description
1.2.1.1: <description> = getDescription(<user-profile>)

Figure 18 SecuredAccess Sequence Diagram

The Authenticaton Authorization Access service can be externalized; It is used to retrieve identity material from which it is be possible to associate a user and a sensiNact resource model data structure to a *UserProfile* (the sensiNact platform manages a database linking this identity to a *UserProfile* for a specific data structure). For all data structures for which the user has not been registered the Anonymous *UserProfile* is used by default (except if the owner of a resource has defined this default profile to another one). The internal database also gathers information relative to the minimum required *UserProfile* to access to data structures. This definition can be made at each level of the resource model, knowing that if no *UserProfile* is defined for a data structure, the one specified for its parent is used.

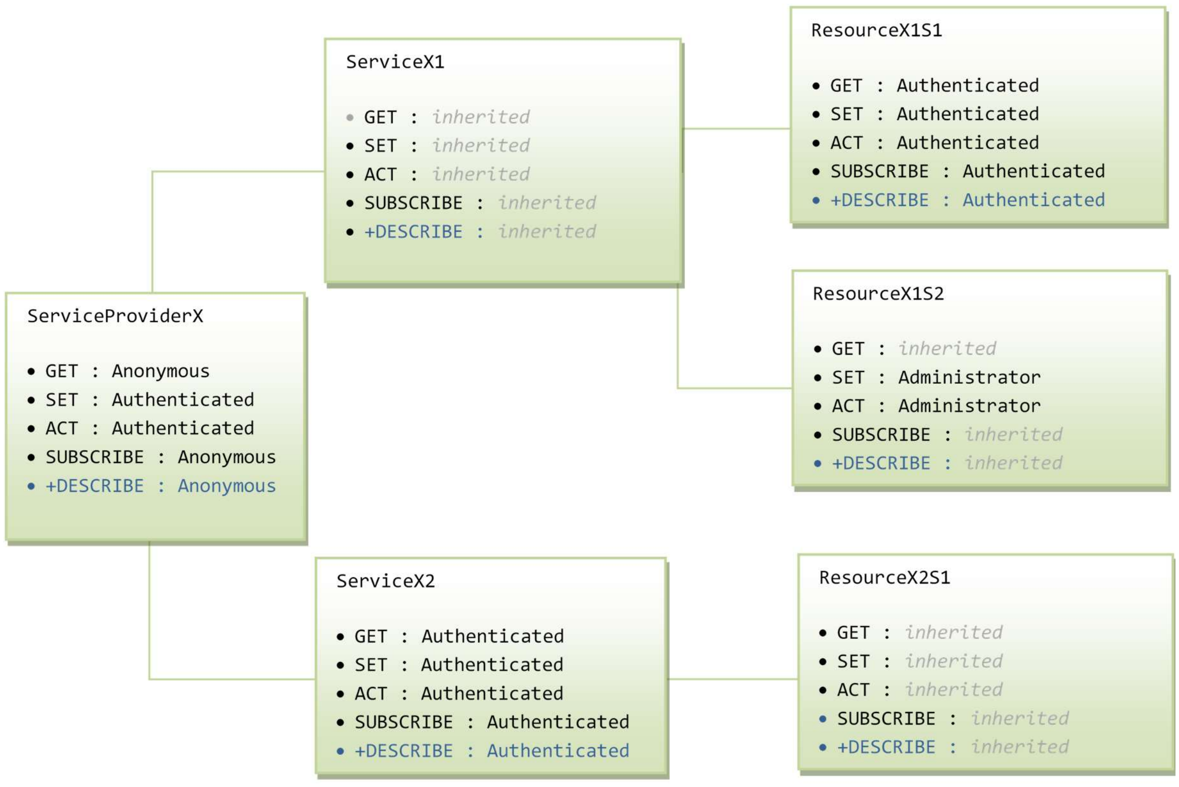


Figure 19 Access right inheritance diagram example

For example, according to the diagram shown above, a user trying to access to the ServiceProviderX for which its UserProfile is Anonymous will receive a description object in which only one Service will be referenced (ServiceX1), containing a single Resource (ResourceX1S2) providing two AccessMethods, GET and SUBSCRIBE.

Federation approach

SensiNact is based on a service-oriented approach where its functionalities are exported in terms of services, which allows easy integration of those features within the federated FESTIVAL's Experimentation Testbeds as a Service.

3.3.1.3 Application Manager

SensiNact has a component named AppManager, this component aims to create higher level applications based on the resources provided by the sensiNact gateway and which the life-cycle can controlled by the sensinact gateway.

AppManager provides a way to develop event driven applications, i.e., based on the Event-Condition-Actions (ECA) axiom. Thus, the application is only triggered when the required events occur. Then, if all conditions are satisfied, the actions are done. Events are created from sNa sensors and the actions are performed using the sNa actuators available in the environment.

3.3.1.4 Data model and JSON format

The AppManager assumes that an application is a set of bound components. Each component processes a single function (e.g., addition, comparison, action). The result of this function is stored in a variable in the current instance of the application. The components using this result as input listen to the corresponding variable. When the variable changes, they are notified and can process their own function, leading to a new result.

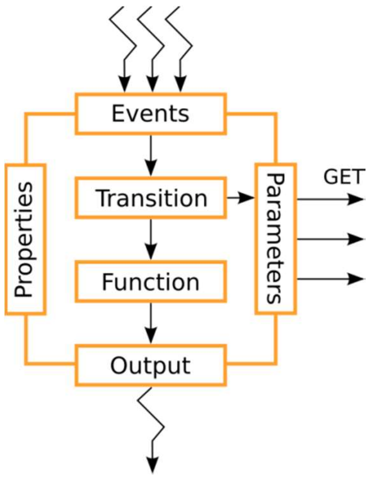


Figure 20: Architecture of a sNa component.

The component is the atomic element of an application. Thus, an application can consider a single component to perform an action. It holds the minimal requirements to create an ECA application:

- Events: events that trigger the process of a component. Trigger can be conditioned to a specific event or a specific value of the event (e.g., when the value of the sensor reach a threshold);
- Function: function wrapped in the component (e.g., addition, comparison, action). The acquisition of the parameters is realized in the transition block before the function block;
- Parameters: parameters of the function that are not available in the event (e.g., static value, sensors values).
- Output: result of the function that is stored in a variable and that triggers a new event.
- Properties: non-functional properties of the component (e.g., register the result as a new resource in sNa).

The AppManager is a sNa service provider. Thus like any other resource it provides a set of resources; in this specific case an INSTALL and an UNINSTALL resources, enabling a client to install/uninstall an application.

A sNa application is described using a JSON file. We developed a specific Architecture Description Language (ADL) to describe the components used in an application and the bindings between the components.

The following JSON code example corresponds to the code of a single component:

```
{
"events": [{
"value": "resource1",
"type": "resource",
"condition": {
"operator": ">=",
"value": 100,
"type": "integer"
}
}],
"function": "function_name",
"parameters": [{
"value": "ON",
"type": "string"
}],
"properties": {},
"output": "output_name"
}
```

Figure 21: JSON example of a sNa component

This component specifies that when the resource1 is greater or equals to 100, the function_name is called with the string parameter "ON". The result of the function is stored in the output_name variable and triggers a new event that may be used by others components.

The supported types are:

- Primitives types: integer, boolean, long, double, float, string. This is used to described a static variable;
- Resource type: resource. This is used to refer to a resource. If this is set in the JSON Event section of the JSON, a SUBSCRIBE is done on the resource. If this is done in any JSON Parameters section, a GET is done on the resource and returns the current value;
- Variable type: variable. This is used to refer to the output of a previously processed component;
- Event type: event. This is used to refer to the value of the event that triggers the function. This type is never used in the condition of the JSON Event section.

Here after is a synthesis of the type that can be used in the different parts of the JSON file.

	Primitive types	Resource type	Variable type	Event type
In event type	No	Yes	Yes	No
In event/condition type	Yes	Yes	Yes	No
In parameters type	Yes	Yes	Yes	Yes

Table 18: Types used in the JSON component.

The AppManager supports the validation of the JSON files against a JSON schema. Schemas exist in the plugins and may be used by the developers of the applications.

3.3.1.5 Architecture

The AppManager is designed to be used as any sNa service provider. Thus it provides an "Install" and an "Uninstall" resource, enabling a client to install/uninstall an application. These resources are accessible using different bridges, such as any actuators.

The AppManager architecture is also designed to easily add new functions and to handle the lifecycle of applications in order to perform checks.

Plugins

Plugins enable to add new function to the AppManager. New plugins require to implements the mandatory interfaces Java interface to be found in the OSGi registry and thus be used by the AppManager. The AppManager is currently supporting the following functions.

Plugin	Functions supported
Basic Plugin	various operators (e.g., equals, greater than, lesser than, different), addition, subtraction, division, multiplication, modulo, concatenation, substring, ACT and SET methods on resources
CEP Plugin	after, before, coincides, average, average deviation, count, max, min, median, standard deviation, sum

Table 19: Functions supported by the plugins of the AppManager.

Lifecycle

The AppManager provides a lifecycle to manage the applications. It enables to process various checks during different steps of the lifecycle of the application (e.g., ADL consistency, resource permissions). The first step is to install the application, i.e., send the ADL of the application. If there is a problem, the AppManager returns an error. Once the application is installed, it can be started and its state changes to "Resolving". If there is a problem during this step, the application enters in the "Unresolved" state. Otherwise, the application is active until it is stopped or an exception occurs.

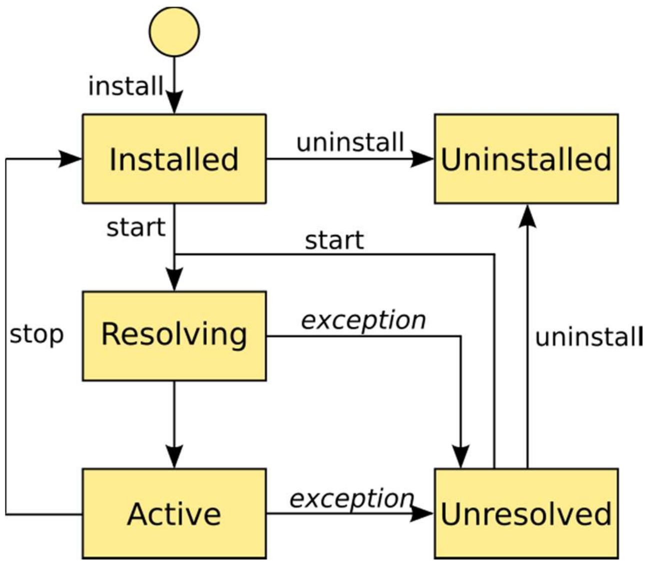


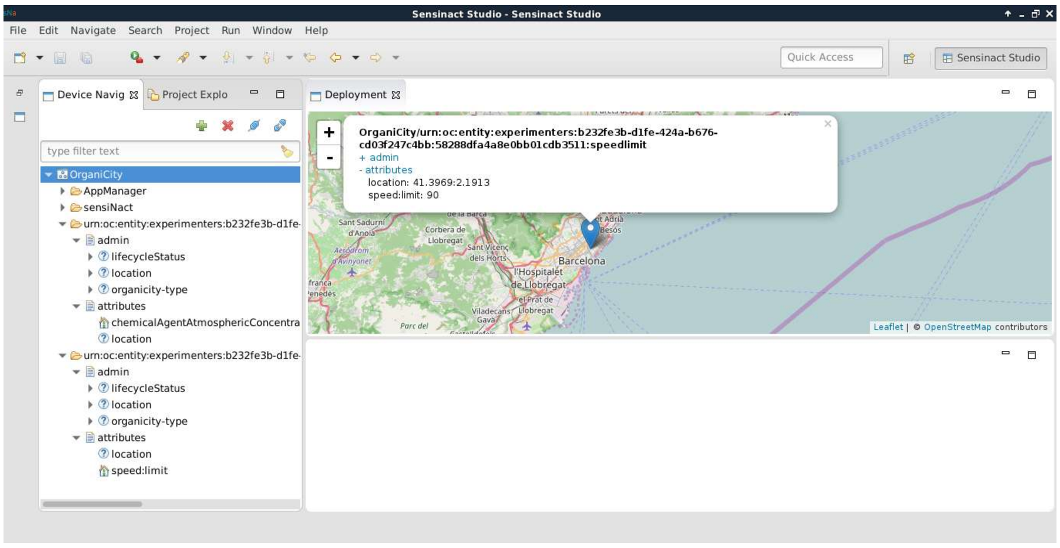
Figure 22: Lifecycle of an application.

Instances

The AppManager allows multiple instances of the same application to run in parallel. When an event occurs, the InstanceFactory of the application instantiates a new set of components and passes the event to the first component. The number of instances can be set in the application properties. If, there is more events than available instances, events are stored and processed when an instance ends.

SensiNact Studio

SensiNact Studio allows an easy interaction with the IoT devices and the creation of applications. The Studio is based on the Eclipse platform and built as a rich client platform application. The Graphical User Interface (GUI) is developed using the views mechanism from Eclipse. Thus, it proposes views for browsing devices, locating devices on a map and interacting with them, i.e., getting value from sensors or performing actions on actuators. The Studio is also targeted to ease the creation of IoT application following the Event-Condition-Action (ECA) pattern.



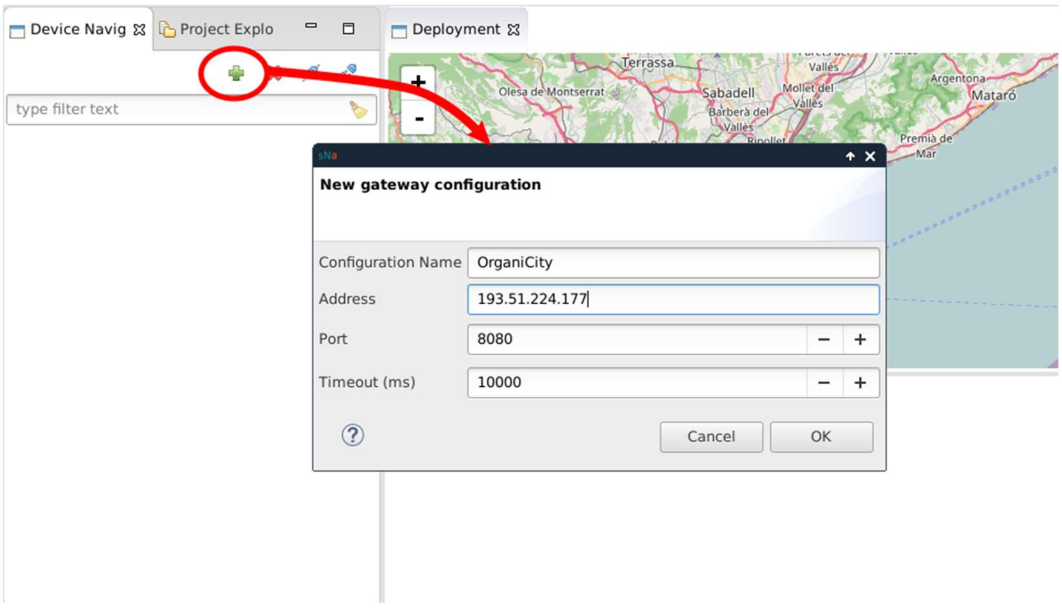
```
OrganiCity/urn:oc:entity:experimenters:b232fe3b-d1fe-424a-b676-
cd03f247c4bb:58288dfa4a8e0bb01cdb3511:speedlimit
+ admin
- attributes
  location: 41.3969:2.1913
  speed:limit: 90
```

Figure 23: sensiNact Studio Graphical User Interface.

The GUI (Figure 23) includes different views: navigator, deployment, properties views, as well as a Domain Specific Language (DSL) editor.

Browsing devices

Before users can use the studio for managing devices and applications, they need to connect a sensiNact gateway. This action is performed by clicking on the plus sign icon on the device navigator. Then, gateway information have to be provided (Figure 24).



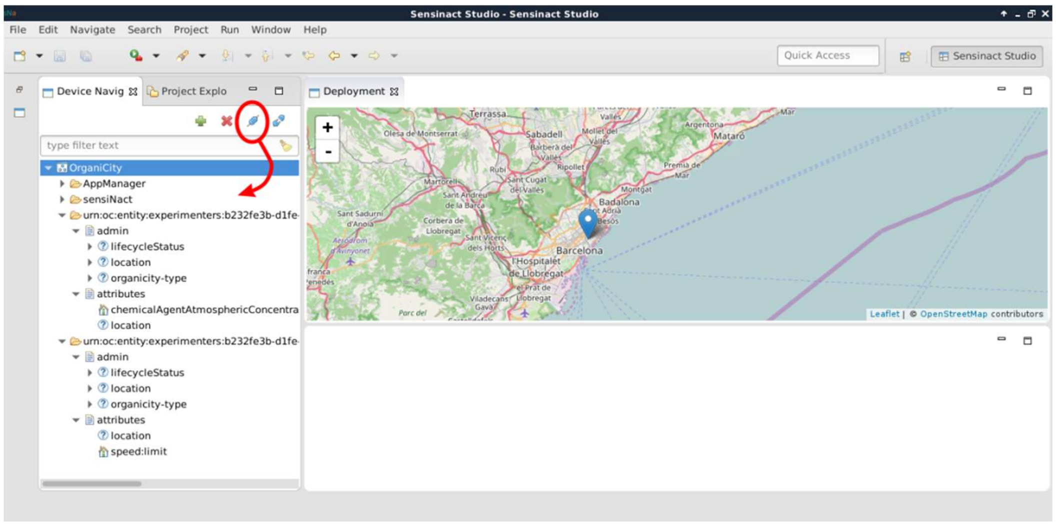
New gateway configuration

Configuration Name	OrganiCity
Address	193.51.224.177
Port	8080
Timeout (ms)	10000

Cancel OK

Figure 24: Gateway configuration.

Once the information has been provided and the dialog validated, the Gateway is added to the Navigator View. To display and browse the available devices imported by this gateway, connecting to it is needed. This action is performed using the connect button (Figure 25).

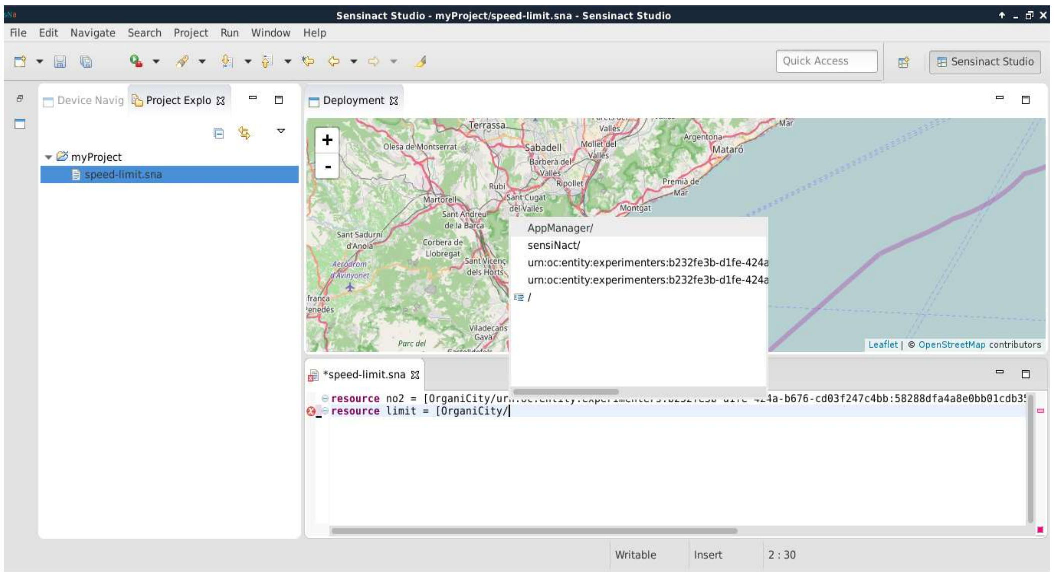

Sensinact Studio - Sensinact Studio
File Edit Navigate Search Project Run Window Help
Quick Access
Sensinact Studio
Device Navig
Project Explo
Deployment
type filter text
Barcelona
Leaflet | © OpenStreetMap contributors

```
OrganiCity
  AppManager
  sensiNact
  urn:oc:entity:experimenters:b232fe3b-d1fe-
    admin
      lifecycleStatus
      location
      organicity-type
    attributes
      chemicalAgentAtmosphericConcentra
      location
  urn:oc:entity:experimenters:b232fe3b-d1fe-
    admin
      lifecycleStatus
      location
      organicity-type
    attributes
      location
      speed:limit
```

Figure 25: Gateway connection.

The device Navigator View is then populated, and pin points are displayed on the map. By clicking on attributes names, it is possible to get the current value for the considered attribute. It is also possible to see attributes values on the map, clicking on the pin points (see Figure 25).

Application creation


Sensinact Studio - myProject/speed-limit.sna - Sensinact Studio
File Edit Navigate Search Project Run Window Help
Quick Access
Sensinact Studio
Device Navig
Project Explo
myProject
speed-limit.sna
Deployment
*speed-limit.sna
AppManager/
sensiNact/
urn:oc:entity:experimenters:b232fe3b-d1fe-424a
urn:oc:entity:experimenters:b232fe3b-d1fe-424a
/
Leaflet | © OpenStreetMap contributors
Writable
Insert
2 : 30

```
resource no2 = [OrganiCity/ur[illegible]4a-b676-cd03f247c4bb:58288dfa4a8e0bb01cdb3
resource limit = [OrganiCity/
```

Figure 26: Application creation.

The SensiNact Studio allows the creation of applications to be executed on the gateway. Creating an application is performed by writing a script using a dedicated syntax, and deploying this script to the gateway.

On Figure 26, a project has been created on the project explorer view (on the left). In this project, a script named *speed-limit.sna* has been created, and is being edited. As the figures shows, the editor provides code highlighting (some keywords are displayed in a special font), code completion (with popups) and a syntax validator which displays red crosses on the script margin in case of error.

The dedicated syntax, a Domain Specific Language, is composed by the following blocks:

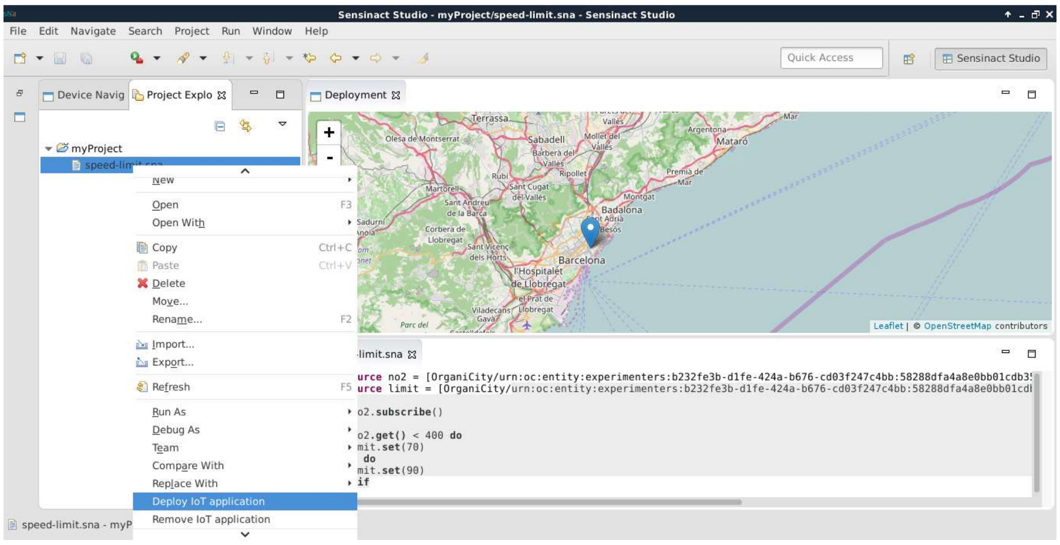
- The shortcut block: each resource is accessible through a unique URI, which can be quite long. This block aims at creating shortcut for the next blocks.
- The event block: the developer defines on which resources the application is triggered. When an event is thrown and is a valid trigger, the conditional block is executed.
- The conditional block: once the application has been triggered, and before any action can be executed, the data from the resource has to satisfy the conditional block. The keyword for this block is if followed by the conditions to be validated.
- The actions statements: if the conditional block is satisfied, actions are performed in the order that they are listed. The actions can be physical actions on actuators or virtual actions such as changing the format of a data using a mathematical function. The available actions, also named functions, are listed below:
- Basic functions: addition, subtraction, division, multiplication, modulo, string concatenation, substring, various operators (e.g., equals, greater than, lesser than, different), ACT and SET methods on sensiNact resources.
- Complex Event Processing functions: after, before, coincides, average, average deviation, count, max, min, median, standard deviation and sum.

Table 20 shows the basic structure for writing a script.

[resource<resource>]+	Shortcut block, which must contains at least one statement.
on<events>+	The event block, lists the events triggering the script. At least one event must be provided.
[if<condition>do]+ **[<actions>]+** **[else do]?** **[<actions>]?+** **end if;**	The conditional block, which lists actions to be performed based on conditions.

Table 20: SensiNact Domain specific language basic syntax

Once the script has been written, it can be deployed to the gateway where it will be executed. This is performed using a right click on the script file (see Figure 27).

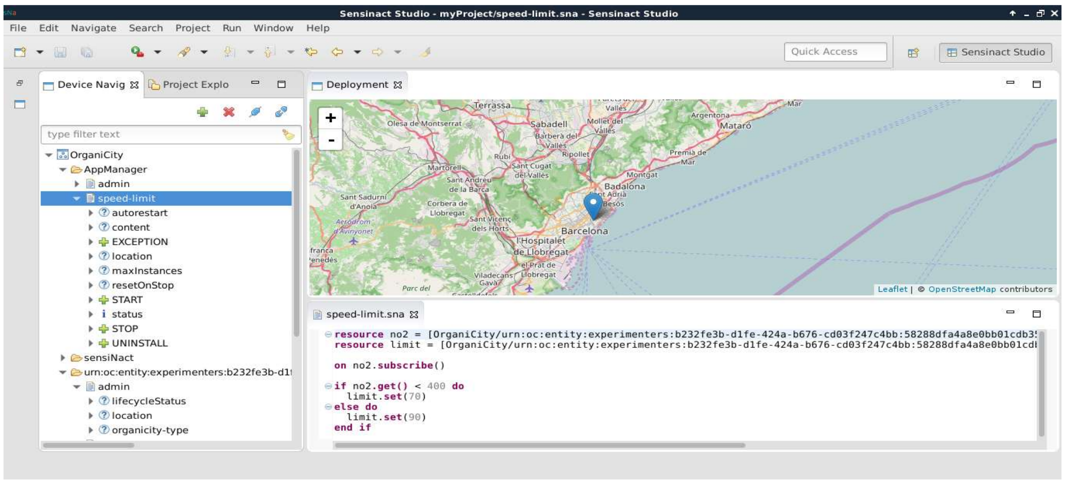


```
urce no2 = [OrganiCity/urn:oc:entity:experimenters:b232fe3b-d1fe-424a-b676-cd03f247c4bb:58288dfa4a8e0bb01cdb3
urce limit = [OrganiCity/urn:oc:entity:experimenters:b232fe3b-d1fe-424a-b676-cd03f247c4bb:58288dfa4a8e0bb01cd

o2.subscribe()

o2.get() < 400 do
mit.set(70)
 do
mit.set(90)
if
```

Figure 27: Application deployment.

Application monitoring

After the application has been deployed, a new set of resources is automatically created under the AppManager device. You can browse those resources into the Device Navigator View (Figure 28).



```
resource no2 = [OrganiCity/urn:oc:entity:experimenters:b232fe3b-d1fe-424a-b676-cd03f247c4bb:58288dfa4a8e0bb01cdb3
resource limit = [OrganiCity/urn:oc:entity:experimenters:b232fe3b-d1fe-424a-b676-cd03f247c4bb:58288dfa4a8e0bb01cd

on no2.subscribe()

if no2.get() < 400 do
  limit.set(70)
else do
  limit.set(90)
end if
```

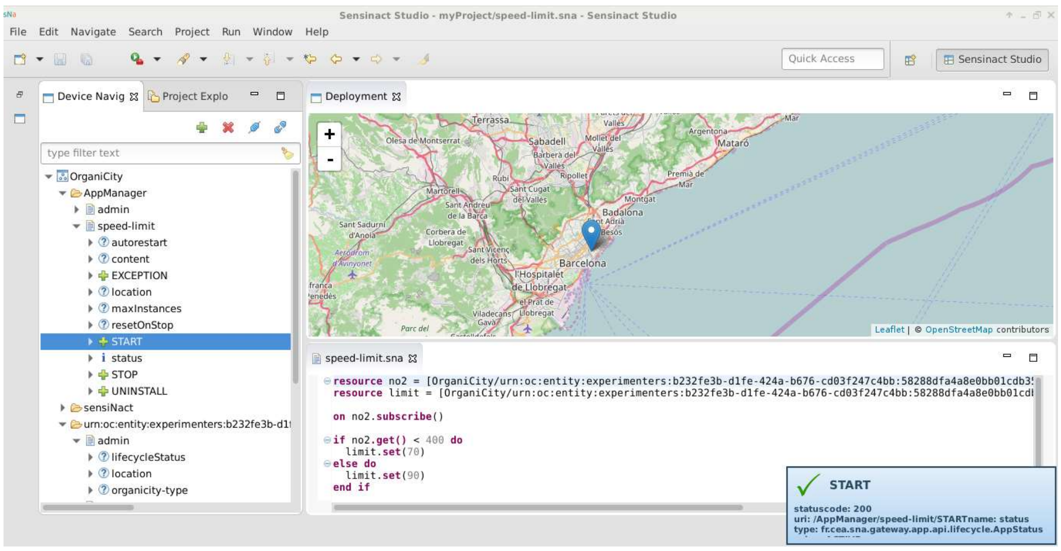
Figure 28: Application management resources.

First of all, a new service is created with the name of the sNa file (without the extension). In our example, it is speed-limit. This service representing the application always contains a standard set of resources (Table 21).

Resource	Type	Description
autorestart	property	In case of failure, decides if the application should be automatically started again
content	property	Script file content
EXCEPTION	action	Deprecated
location	property	GSP location which can be used if it makes sense
maxinstances	property	Number of parallel instances which should be started
resetOnStop	property	On Stop, decides if the generated resources by the application should be destroyed or kept
START	action	Starts the application
status	state variable	Current status of the application: START/STOP/...
STOP	action	Stops the application
UNINSTALL	action	Removes the application

Table 21: Application management resources

To start the application, simply double click on the START resource. This will launch the start action, which will run the script.



```
resource no2 = [OrganiCity/urn:oc:entity:experimenters:b232fe3b-d1fe-424a-b676-cd03f247c4bb:58288dfa4a8e0bb01cdb3!
resource limit = [OrganiCity/urn:oc:entity:experimenters:b232fe3b-d1fe-424a-b676-cd03f247c4bb:58288dfa4a8e0bb01cdl

on no2.subscribe()

if no2.get() < 400 do
  limit.set(70)
else do
  limit.set(90)
end if
```

```
START
statuscode: 200
uri: /AppManager/speed-limit/STARTname: status
type: fr.cea.sna.gateway.app.api.lifecycle.AppStatus
```

Figure 29: Application start-up.

Figure 29 shows that the application is up and running on the server. The studio can be used to check if the application has the expected behaviour, by querying the resources. The studio can also be shut down, since the applications are executed on the gateway.

Reusable components

SensiNact is easily portable to any hardware platform and supports a large number of protocols that allows its easy integration and reuse in various IoT environments.

SensiNact Studio Web

Presenting the information that is absorbed by the gateway is as important as organizing and aligning the concepts on the backend of gateway itself.

Thus in order to provide a simple interface to watch over the sensors integrated in the Sensinact Gateway, each instance of SensiNact offers a mobile-compatible interface that enables the end user to read the information of the gateway and execute standard device calls in the platform.

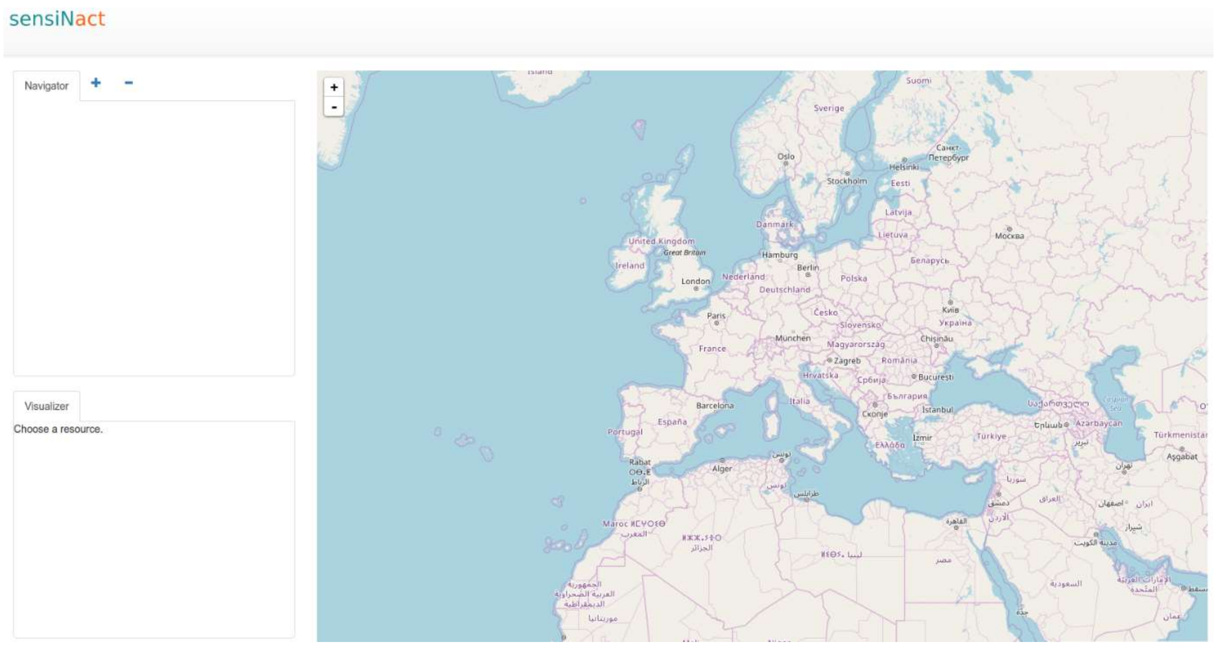
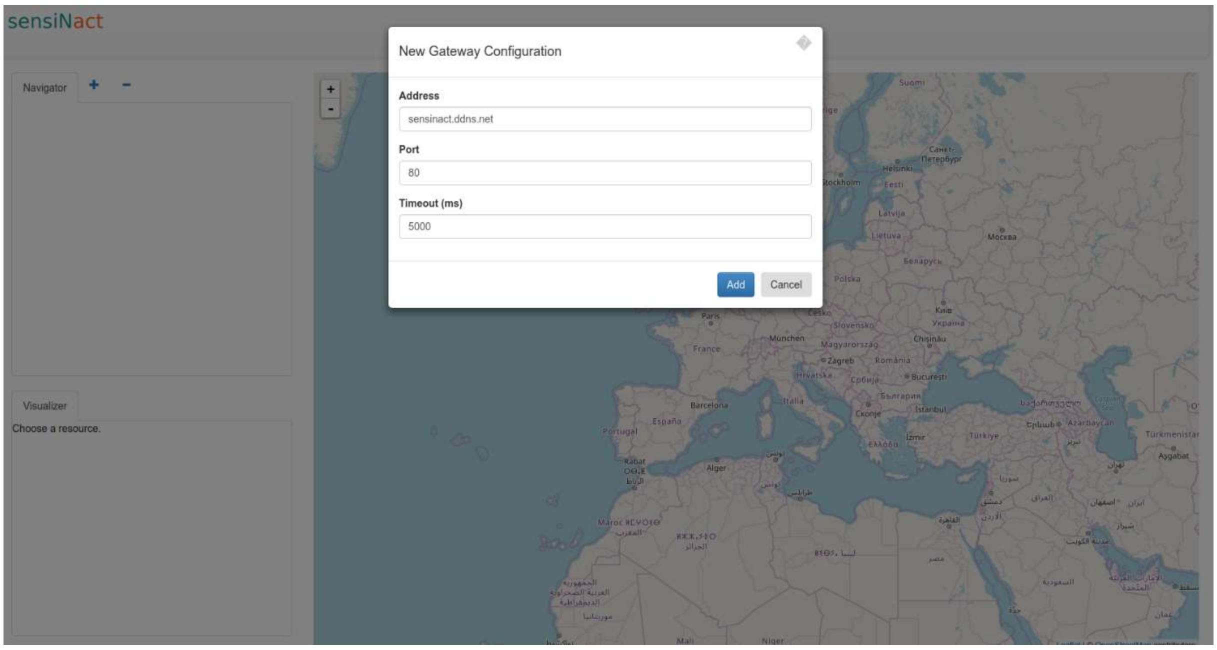


Figure 30 StudioWeb initial screen

The initial screen Figure 30 is composed of 3 areas: *Navigator, Map* and *Visualizer*, they are situated on upper-left, right and bottom-left respectively. Navigator is where the available gateway will be display along with the entire resource hierarchy, meaning displaying the gateway above all the sensors (known as providers), the sensors with all the services available and each service with their respective resource, all display in a tree in which the top most element is the selected gateway.

Within the *Navigator* area it is possible either add a new gateway using the "+" sign, or disconnect (and remove) the gateway using the "-" sign, noticing that this function only make sense if you are connected to one. The connect options can be seen in Figure 31.

This is where the client will input the information about the address of the gateway he/she wishes to be connected to and the actual port.



New Gateway Configuration

Address: sensinact.ddns.net

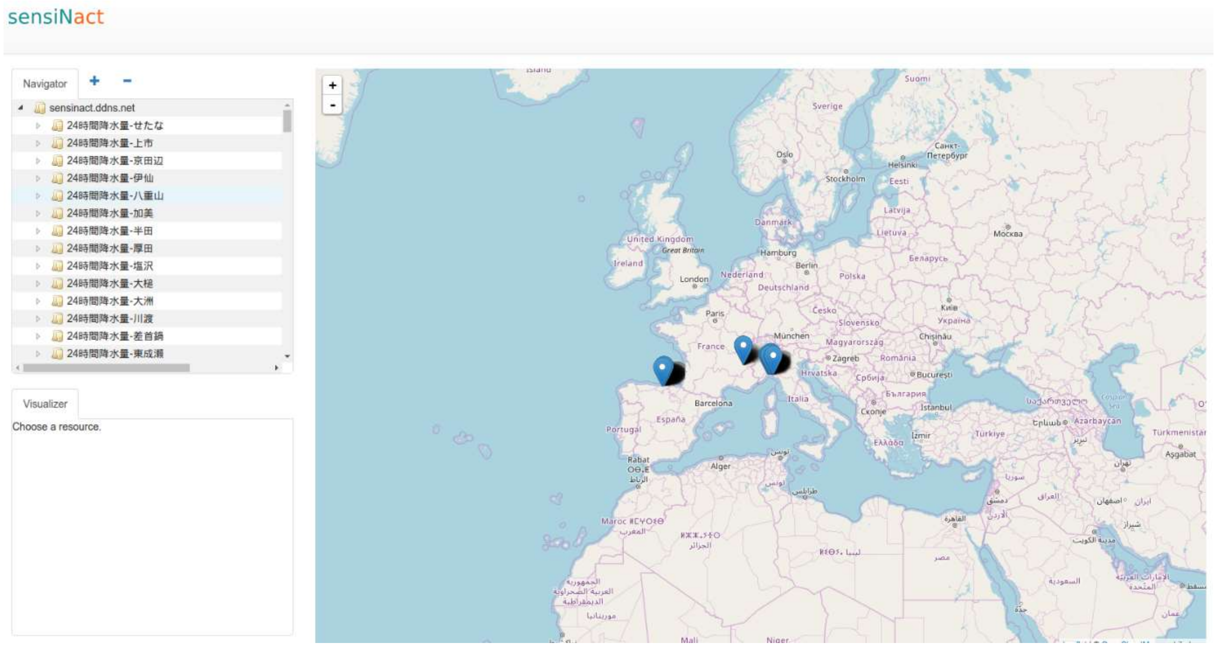
Port: 80

Timeout (ms): 5000

Choose a resource.

Figure 31 StudioWeb connect

After a successful connection, you will be able to see the gateway along with the tree of devices/services/resource attached to the gateway Figure 32.

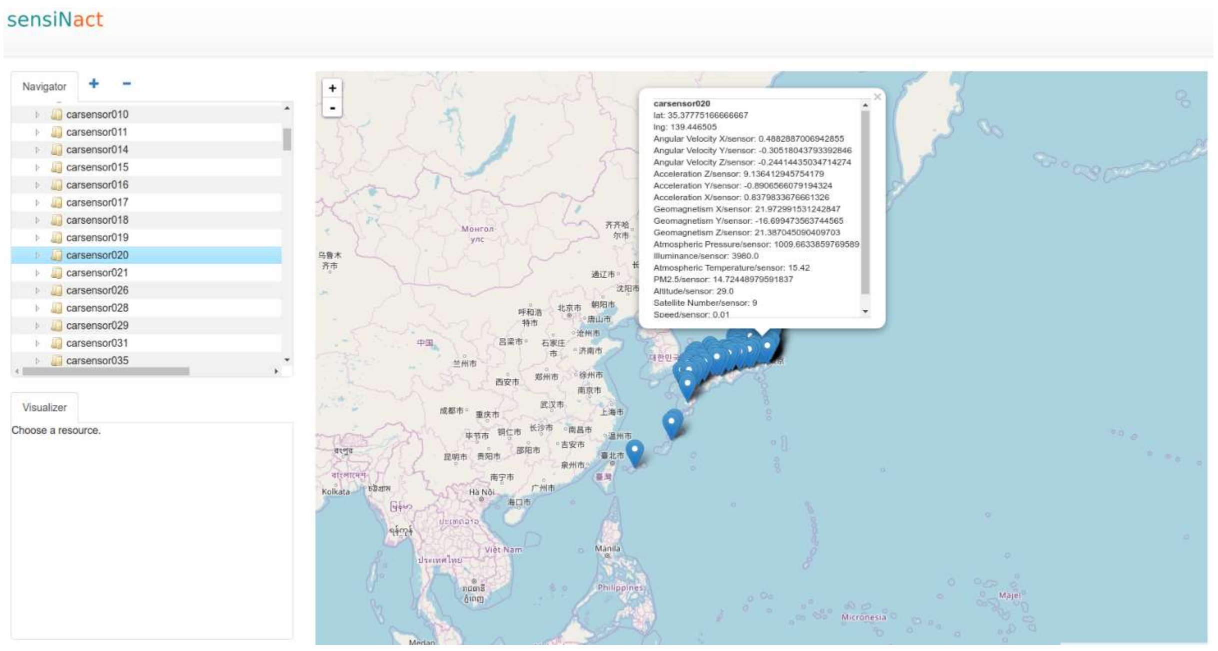


sensinact.ddns.net
- 24時間降水量-せたな
- 24時間降水量-上市
- 24時間降水量-京田辺
- 24時間降水量-伊仙
- 24時間降水量-八重山
- 24時間降水量-加美
- 24時間降水量-半田
- 24時間降水量-厚田
- 24時間降水量-塩沢
- 24時間降水量-大樹
- 24時間降水量-大洲
- 24時間降水量-川渡
- 24時間降水量-差首鍋
- 24時間降水量-東成瀬

Choose a resource.

Figure 32 StudioWeb gateway content

All sensors, known as well as providers, are display on the upper-left part of the interface, if a specific sensor is selected, the Map area will lead to the geographic position of that sensor (if that information is available) and all the information of services and resources contained in that sensor will be display directly on the Map area, see Figure 33.

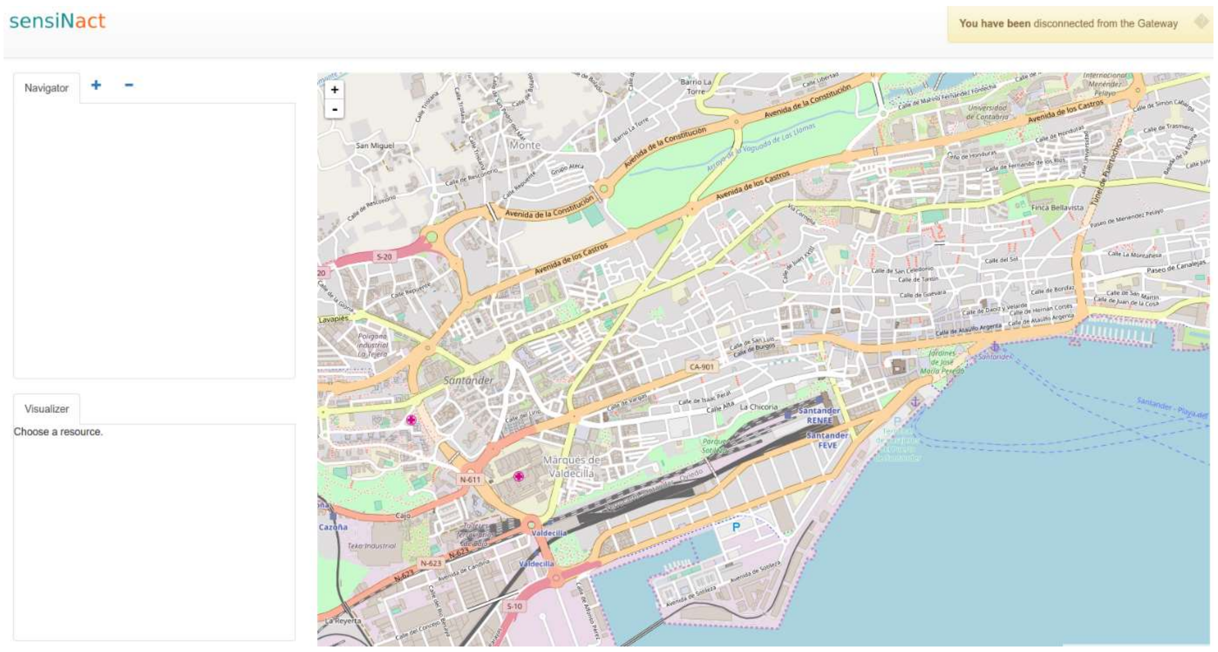


carsensor020
lat: 35.37775166666667
lng: 139.446505
Angular Velocity X/sensor: 0.4882887006942855
Angular Velocity Y/sensor: -0.30518043793392846
Angular Velocity Z/sensor: -0.24414435034714274
Acceleration Z/sensor: 9.136412945754179
Acceleration Y/sensor: -0.8906566079194324
Acceleration X/sensor: 0.8379833676661326
Geomagnetism X/sensor: 21.972991531242847
Geomagnetism Y/sensor: -16.699473563744565
Geomagnetism Z/sensor: 21.387045090409703
Atmospheric Pressure/sensor: 1009.6633859769589
Illuminance/sensor: 3980.0
Atmospheric Temperature/sensor: 15.42
PM2.5/sensor: 14.72448979591837
Altitude/sensor: 29.0
Satellite Number/sensor: 9
Speed/sensor: 0.01

Figure 33 StudioWeb: Sensor data

Once the connection is established with an active gateway, the StudioWeb is receiving updates notifications from the gateway for the new data, which may include new devices attached to the platform, disconnected devices or updated sensor data notifications. According to the capability of the gateway (processing capability) you may be disconnected in order to preserve the resource consumption on the back-end.

After been disconnect voluntarily from the gateway a message will be display on the upper-right part of the screen, see Figure 34



You have been disconnected from the Gateway

Figure 34 StudioWeb gateway disconnection

3.3.1.6 Integration of IoT framework

General description

In this section of the document the approach to integrate Sensinact Platform and InterIoT will be described underlining the interception point on each system and the advantage on using them. Although before describing the integration approach, it is important to understand how the two system work apart; this will allow to establish the best connection point and the information exchanged between the two systems.

The integration of sensiNact to the Inter-IoT Framework will be done at the middleware layer. sensiNact is currently in use as IoT middleware platform in various collaborative projects such as OrganiCity, FESTIVAL, BigClouT, Wise-IoT, ACTIVAGE and IoF2020, in which applications in various domains have been (and will be) developed and deployed in close to real life environments in domains such as smart city, smart home, smart farming, smart living and smart ski resort. Integration. With integration of sensiNact with the Inter-IoT framework, we aim at bringing more devices, platforms, thus more data sources to enrich the "catalogue" of Inter-IoT supported platforms. This will allow, not only to Inter-IoT validating its interoperability methodology and tools but it will also allow sensiNact being compatible with other platforms supported by Inter-IoT.

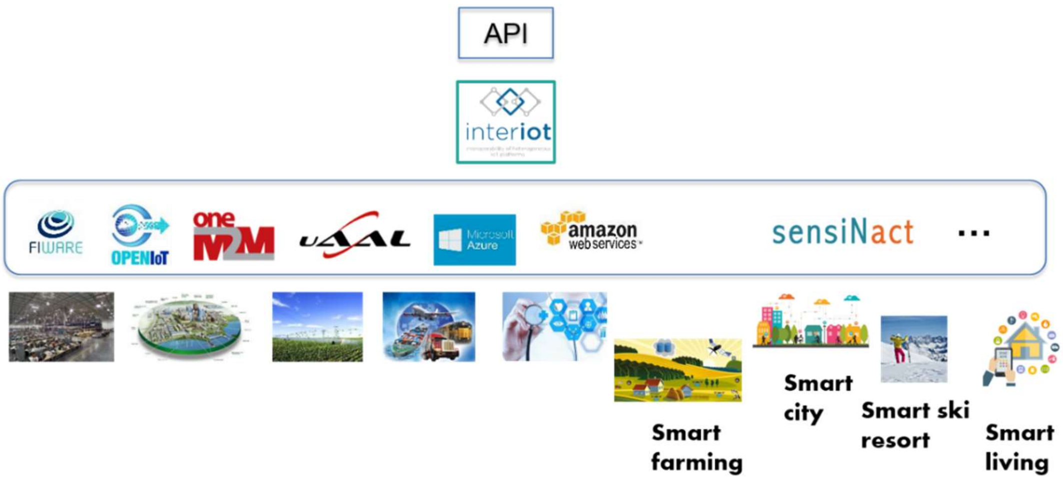


Figure 35 Integrating sensiNact and Inter-IoT catalog

3.3.1.7 Deliverables and version overview

The following table contains a deliverable list which need to be signed of before FAT testing commences.

ID	Description	Check
Documents		
1	Validation and Test reports of the IoT system components	
2	Validation and Test reports of the Pilot system components	
3	FAT applicable for the platform targeted	
Hardware		
	Not applicable	
Tools		

7	Wireshark	
8	Java SDK 1.7	
9	Telnet	
10	CURL	

Table 22: Deliverable checklist

The following table shows the software components and version of which the system release version 1.5 of sensiNact

ID	Description	Version	Check
IoT Physical Gateway			
1	Protocol Controller		
IoT Virtual Gateway			
2	MW Controller		
3	API Request Manager		
4	Platform Request Manager		
5	REST API Interface		
6	Sensinact Gateway	v1.5	
7	Sensinact Studio	v1.5	
8	Sensinact Studio Web	v1.5	
Universal container			
7	UniversAAL REST API	v3.2.1	

Table 23: Component version overview

3.3.1.8 Requirements, scenarios and use cases

The following table provides an overview of the relation between the requirements and the test(s) that validate their implementation.

ID	Description	Covered by
R01	Extensibility (feature evolution)	TS 0
R04	Support of common IoT communication protocols	TS 0
R07	Real time support	TS 0, TS 0
R10	Remote device control	TS 0
R11	System security	TS 0
R12	System privacy	TS 0
R18	Service discoverability	TS 0
R02	API REST	TS 0
R04	Easy-to-use user interface	TS 0
R05	Application response time	TS 0
R06	Communication with message size efficient protocols	TS 0
R13	Open Source	TS 0, TS 0

R15	Usability	TS 0
R16	Documentation	TS 0, TS 0
R02	Extensibility of the use cases	TS 0
R03	Use of standards	TS 0, TS 0, TS 0
R07	Multiple interface options	TS 0, TS 0
R12	Gateway access API	TS 0
R07	Manage a sensor or actuator	TS 0, TS 0

Table 24: Requirements vs test mapping

3.3.1.9 Test environment

Introduction

To test the functionality of the SensiNact in combination with the IoT framework a representative test system is needed. The test system needs to approach the "real world" as much as possible. The pilot setups must be recreated and proven. This chapter will describe this environment and the used hardware, software, tools and platforms.

Test environment

This stage is the last point to assure the compliance to Sensinact IoT platform with respect to the requirements established in INTER-IoT requirements. Those requirements were selected in Section 3.3.1.8 and will be used as reference to what setup to use in order to verify the functionality in the target platform.

Test setups, tools, hooks and probes

This paragraph describes the test setups, tools, hooks and probes used in the by this document defined tests.

A test setup defines a setup used for a test, and describe the used system parts, interface etc.

Tools define the used COTS tooling that will be used during testing e.g. Logic analyzers, packet sniffer like e.g. Wireshark.

Hooks are used to inject data or control parts of the system used to get the system to handle a needed scenario. This can be hidden, debug or service interfaces which contain the needed functionality.

Probes are used to gather logging from parts of the system under test to provide system feedback and to prove that the system is handling the scenario as it should.

The following paragraphs will define the used setups, tools, hooks and probes which will be referred to by the test descriptions.

TS_01 Encrypted data transfer

In order to preserve user traffic information, the platform is able to encrypt data been transferred through the network to avoid user information from been hijack or simply accessed by a third-party actor, in either case the user can be compromised if this accessed is granted.

One way to protect the user information is to encrypt the communication channel, there are several techniques in the marked to doing so, one of the most common way of doing it is to user HTTPS protocol. HTTPS or Hyper-Text Transfer Protocol Secure is a secure protocol used to transfer information across HTTP.

This protocol make use either of SSL (or TLS) in order to encrypt the content to be transmitted. The technical details on how this encryption happen is out of the scope of this section, in this document it will be explained in how to configure this mechanism in the platform in order to protect the user data transmission by assuring that no network relay become a safety breach.

In order to configure the support for this encryption mechanism couple steps are necessary, initially it will be necessary to have a repository of trusted certificates available in the computer instance running the Sensinact gateway, this repository is called *keystore*.

In most cases all computer instance running as application server already have one keystore that should be re-used to store the credential of the sensinact application to provide the encryption capability.

In order to setup this test we will need:

- Install Sensinact platform
- Select modules pertinent for the test
- Generate encryption key
- Setup platform to take into consideration the encryption key
- Start sensinact

Along with Java Standard Development Kit, there exists an application called *keytool* that can be used to generate a security key.

In this test we will generate one *keystore* using *keytool* command.

TT_01 Wireshark

Wireshark is a packet analyzer capable of sniffing the packets on network layer and display their content to the user. This tool allows the user to inspect the package information from the network layer.

The tool already provides several modes of displaying the package/frame information providing already a human-readable version of the package circulating on the network.

The minimum version to be used for this FAT is the version 1.10.6.

TT_02 CURL

CURL is a command line tool to design to easy up the HTTP calls execution. It allows the user to perform HTTP calls and allows to customize the calls in the most convenient way to the user. This tool will be used during the test to perform the calls to the gateway platform on the north-bound.

The minimum version that can be used in this FAT is the version 7.35.0

TT_03 Keytool

Keytool is a command line application included in the Java SDK that manages key generation and certificates for the JVM.

The *keytool* version adopted for this FAT is the version provided with Java SDK 1.7.

TS_02 Device materialization from MQTT protocol

MQTT is a well-known messaging communication protocol, this standard is actually managed by OASIS group. This broker has become a reference and have been used by large communities like Eclipse.

TT_04 CURL

Refer to previous section.

TT_05 Mosquitto Client

Mosquitto is a server and client from a well-known message protocol called MQTT. The client and server can be used separately.

Mosquitto Client is capable of connecting to a MQTT broker and subscribe to a topic. The notification is received directly on the command console or can be redirected to a file.

TT_06 Mosquitto Server

Provided with the *mosquitto* package, the mosquito server is one of the reference implementation for the MQTT message protocol.

The *mosquitto* server is used directly by the mqtt southbound module of SensiNact gateway.

TS_03 Documentation

The documentation of the project should be present in accessible servers held by a third party entity that creates or host open source project or technological initiatives.

TP_07 Eclipse website

The latest documentations at architectural level and end user tutorials can be found on the Eclipse foundation website (below), Eclipse foundation which is one of the largest entities to host open source projects[1].

IoT Open Platforms hosts the conceptual documentation of Sensinact project, this documentation can be found in the web site[2].

Technical tutorials and other tool guides can also be found in the following web[3].

TS_04 Source Code

SensiNact is an open source project incubated by Eclipse foundation, as such, sensinact source code is available to anyone. Three roles exist: Lead, Committers and Contributors.

The Lead will give a broad view of the direction and collaboration transversally with other IoT projects or technologies. Committers are actively pushing new functionalities, corrections and evaluating the work of Contributors to be integrated in the main source repository.

Anyone can contribution can be integrated in the main source repository as long as it is approved by at least two Committers.

TP_08 Source from Eclipse Gerrit website

[1] https://projects.eclipse.org/proposals/eclipse-sensinact

[2] http://open-platforms.eu/library/sensinact-aka-butler-smart-gateway/

[3] https://wiki.eclipse.org/SensiNact

The source code can be found in Eclipse Gerrit website[4].

TS_05 RESTful API

In order to allow SeniNact users/administrators to access remotely the sensors and their respective services, the sensinact gateway offers a RESTful API that allow to use HTTP protocol to invoke actions and request for information either related to the gateway itself or to resources.

TP_09 Swagger API

Swagger API is an easy-to-use interface for testing RESTful API, it provides a user interface to access the API documentation and at the same time an interface to test the API calls.

By default, the sensiNact gateway embeds a Swagger interface. In order to access this interface, you have to:

- Have sensinact available on your machine
- Activate the module named “rest”
- Point a browser to *http://localhost/swagger-api/index.html*

TS_06 Websocket communication channel

Websocket is a full-duplex communication protocol standardizes by IETF in RFC 6455 used in browser to allow back-end services to push message from server to clients. This protocol is located at Application layer on the OSI model.

In order to provide a near real-time notification, SensiNact platform uses Websocket protocol, the support is integrated via northbound, and allows user to receive notification from sensinact gateway about data updates or any other notification will request from the client based on a subscription pattern.

TP_10 Connect to SensiNact websocket channel

Sensinact websocket channel is available non-encrypted “ws://” or encrypted “wss://” fashion, the non-encrypted fashion is available without any additional configuration, although to use the encrypted channel the user must follow the procedure to configure *Encrypted data transfer* configuration in the section with the same title.

Even though the non-encrypted fashion is by default communication transmission used by default, since sensinact is an OSGi standard based platform all the functional modules are optional, thus, the websocket support is included in one of the functional modules of sensinact, and to activate you have to follow the steps:

- Have sensinact available on your machine
- Activate the module named “rest”
- You can reach the Websocket channel using the *ws://localhost/websocket/*
    - *Be aware that you will need to use some other tool (e.g. javascript) that include a websocket client in order to access sensinact websocket northbound*

TS_07 SensiNact Studio Web

In order to offer a basic access to sensinact gateway resources, sensinact comes with an embedded user interface named Studio Web that is present in the sensinact distribution.

[4] https://git.eclipse.org/c/sensinact/org.eclipse.sensinact.gateway.git/

This module is present but not activated by default by the platform.

- Have sensinact available on your machine
- Activate the module named “studio-web”

3.3.1.10 Test description

Test output log files… Folder “log”, prefix “sensinact-`Tx.y.z.log`”

Scenario

ID	T1.1
Test	Verify that the application uses only encryption channels to exchange with clients when fetching sensor data
Setup	TT 0, TT 0
Start	Activate module rest, simulated devices; start the platform, start Wireshark
Req.	R
Input	Retrieve the value of the current value of simulated button device
Output	Verify on Wireshark if any of the information sent can be seen by the sniffer.
Logs	sensinact-`T1.1.1.log`
Outcome	Pass / Fail

ID	T1.2
Test	Verify that the application uses only encryption channels to exchange with clients when activating an actuator
Setup	TT 0, TT 0
Start	Activate module rest, simulated devices; start the platform, start Wireshark
Req.	R
Input	Retrieve the value of the current value of simulated button device
Output	Verify on Wireshark if any of the information sent can be seen by the sniffer.
Logs	sensinact-`T1.2.1.log`
Outcome	Pass / Fail

ID	T1.3

Test	Verify that the application uses only encryption channels to exchange with clients when receiving a notification from the gateway
Setup	TT 0, TT 0, TT 0
Start	Activate module rest, simulated devices; start the platform, start Wireshark
Req.	R
Input	Retrieve the value of the current value of simulated button device
Output	Verify on Wireshark if any of the information sent can be seen by the sniffer.
Logs	sensinact-`T1.3.1.log`
Outcome	Pass / Fail

ID	**T2.1**
Test	Verify documentation availability on the website
Setup	TT 0
Start	Point the browser to URL indicated on the setup
Req.	R
Input	-
Output	Verify that the documentation is available
Logs	sensinact-`T2.1.1.log`
Outcome	Pass / Fail

ID	**T3.1**
Test	Verify that the code source is available on the website
Setup	TT 0
Start	Point the browser to URL indicated on the setup
Req.	R
Input	-
Output	Verify that the documentation is available
Logs	sensinact-`T3.1.1.log`
Outcome	Pass / Fail

ID	T4.1
Test	Verify that the REST API is available on the gateway
Setup	TT 0
Start	Point the browser to URL indicated on the setup
Req.	R
Input	-
Output	Verify that the documentation is available
Logs	sensinact-`T4.1.1.log`
Outcome	Pass / Fail

3.3.1.11 Test outcome overview

Test outcome

The following table will provide an overview of the test result of all the performed tests in this FAT.

Test	Description	Outcome
T1.1	Verify that the application uses only encryption channels to exchange with clients when fetching sensor data	Pass / Fail
T1.2	Verify that the application uses only encryption channels to exchange with clients when activating an actuator	Pass / Fail
T1.2	Verify that the application uses only encryption channels to exchange with clients when receiving a notification from the gateway	Pass / Fail
T2.1	Verify documentation availability on the website	Pass / Fail
T3.1	Verify that the code source is available on the website	Pass / Fail
T4.1	Verify that the REST API is available on the gateway	Pass / Fail
FAT Outcome		**Pass / Fail**

Table 25: Test outcome overview

3.3.2 Third Party: OM2M

The framework has a centralized structure with an OM2M Infrastructure Node (IN-CSE) at the top, which acts as the server able to store measurements. This Infrastructure Node, running in a standard computer, will also be the central entity of the oneM2M standard and the point of contact with other middleware services by means of the INTER-IoT – OneM2M Bridge. Then, a set of IoT devices will be deployed in the environment to acquire physical data and send it to the OM2M IN-CSE. The IoT population will consist of several Stickntrack GPS devices which are able to trace the location and detect the activity of the assets to which they are stuck.

Actually, these IoT devices send activity and location measurements to the Sensolus cloud by using the Sigfox network. However, those measurements are retrieved periodically from the cloud by an Interworking Proxy Entity (IPE) which can run in the OM2M Infrastructure Node. This is possible by means of the APIs provided by the company Stickntrack. The IPE is a Java application that provides a Graphical User Interface (GUI) with which an end user can interact to get data directly from the OM2M IN-CSE. Moreover, the user can visualize the position of all its assets through a Google Maps interface included in the GUI.

The framework is complemented with the Stickntrack Geobeacons which allow continuity of the location information inside buildings. Indeed, when the Stickntrack GPS moves in an environment shielded from the satellite connection, it loses its GPS signal. Since these devices cannot rely on the Assisted GPS (A-GPS) system, they need to be motionless for about 10-15 minutes with a clear view of the sky to recover the signal. Geobeacons will be an alternative to GPS for indoor positioning. The Stickntrack Geobeacon is a simple device that needs to be fixed indoors (e.g. in a warehouse). The device will start sending Bluetooth Low Energy signals that a Stickntrack GPS device can detect via its Bluetooth radio interface. The position of the Stickntrack Geobeacon needs to be specified in the Stickntrack cloud application by using the web interface and account provided by the company Stickntrack. We will provide a OM2M way to do it.

The functionalities of the IPE have already been tested with few devices in the field. The measurements are properly stored in the OM2M Infrastructure Node and can be easily accessed with the GUI by the user. A Stickntrack GPS device has been tested in combination with a Stickntrack Geobeacon device, further tests with several devices of the two types will be performed in the port.

Since the bridge has not been implemented yet, it is possible to retrieve the measurements only from the oneM2M common service layer (in this case OM2M IN-CSE). However, when the bridge will be completed, applications built under a different middleware service will have access to those measurements. At that time, several tests should be carried out to verify the good functioning of the developed bridge.

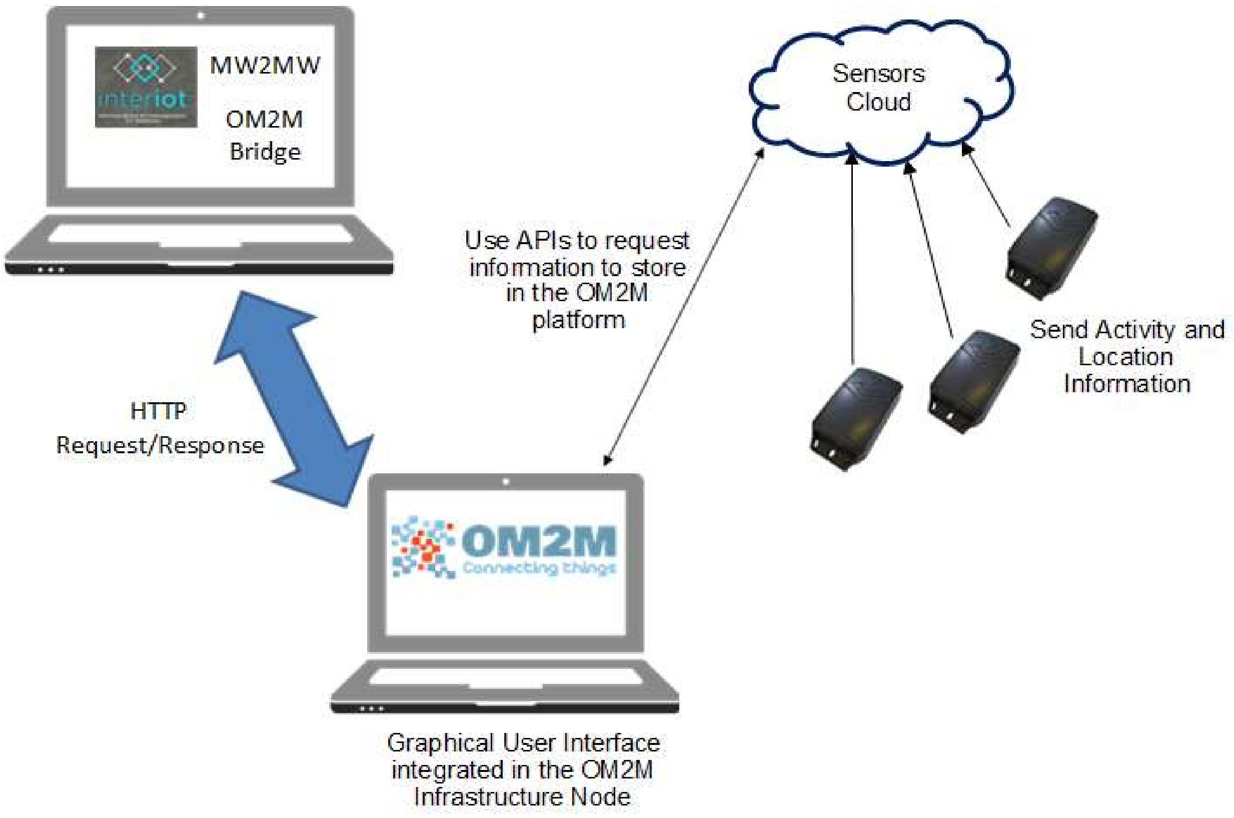

MW2MW
OM2M
Bridge
interiot
Sensors Cloud
Use APIs to request information to store in the OM2M platform
Send Activity and Location Information
HTTP Request/Response
OM2M
Connecting things
Graphical User Interface integrated in the OM2M Infrastructure Node

Figure 36: Test set-up

3.3.2.1 Integration of IoT framework

The Stickntrack GPS devices will be used both to trace the position of vehicles and equipment used in the port and to check the operational state (moving or at standstill). In this way, the workers in the field do not have to waste time to locate machines or vehicles and they are aware of the motion of equipment/vehicles in the port. By using the IPE, it is also possible to set different types of alerts such that a user can be informed whenever a vehicle has not been used for a long period of time or when it has been moved outside a predefined area.

The oneM2M service layer, which stores the measurements from the Stickntrack GPS devices, will be integrated into the pilot architecture by means of the INTER-IoT – oneM2M bridge. Then, a different service layer can send requests to the OM2M Infrastructure Node and retrieve the data. As soon as the bridge will be implemented, it will be possible to start testing the communication between different middleware.

Component	Tests
OM2M IN-CSE	The implementation of the OM2M Infrastructure Node has already been tested by its developers and is open source. On the other hand, we tested the Interworking Proxy Entity with 3 active Stickntrack GPS and during several hours. Everything is prepared to run further tests with a larger amount of IoT devices deployed in the port of Valencia

INTER-IoT – OM2M bridge	This is the most important component of the system, hence it will require several tests before to be used in a real life application
Stickntrack GPS	This device has already been tested by the company Stickntrack so no further tests are necessary. The correct functioning has been confirmed by our experiments
Stickntrack Geobeacon	This device has already been tested by the company Stickntrack so no further tests are necessary. The correct functioning has been confirmed by our experiments

Table 26: Test for each OM2M component.

3.3.2.2 Deliverables and version overview

The Table 27 contains a deliverable list which needs to be signed of before FAT testing commences.

ID	Description	Check
Documents		
1	Validation and Test reports of the IoT system components	
2	Validation and Test reports of the Pilot system components	
Hardware		
4	Standard computer running the OM2M IN-CSE	
5	Stickntrack GPS	
6	Stickntrack Geobeacon	
Tools		
7	Wireshark	
8	IPE Graphical User Interface	

Table 27: Deliverable checklist

The Table 28 shows the software components and version of which the system release version V1.0 consists of.

ID	Description	Version	Check
IoT Physical Gateway			
1	AN Controller	V1.0.3	
2	IN-CSE with IPE	V1.1.0	
IoT Virtual Gateway			
4	Fiware	V4.2.3	
5	oneM2M	V1.1.0	
Universaal container			
7	UniversAAL REST API	V3.2.1	

Table 28: Component version overview

3.3.2.3 Requirements, scenarios and use cases

The following table provides an overview of the relation between the requirements and the test(s) that validate their implementation.

ID	Description	Covered by
14	Platform independency	T2
15	Support of common IoT communication protocols	T1, T2
16	Inter-connection support	T1, T2
23	Device semantic definition	T1
26	Remote device control	T1, T2
27	System security	T1, T2
31	Tools / libraries to support design	T1, T2
42	Heterogeneous information representation	T1
52	API REST	T1
53	Location of sensor and measurement is included in semantic models	T1
70	Easy-to-use user interface	T1
95	Robustness, resilience and availability	T2
108	Open Source	T1, T2
111	Documentation	T1, T2
122	Extensibility of the use cases	T2
123	Use of standards	T1, T2
127	Availability of sensor data	T1
153	Cacheable Data	T1
154	Time stamped event recording	T1
159	Development support for systematic IoT platforms integration/interconnection	T2
168	Provide an alert system	T1
169	Methodology and tools to integrate a proprietary IoT platform	T1
178	Inter Platform Semantic Mediator provides data and semantic interoperability functionality	T2
179	Inter Platform Semantic Mediator supports platform communication	T2
180	Syntactic and semantics interoperability - Data format and semantics translation	T2
235	Support of semantic modelling in the middleware layer	T2
255	A common data model compatible with all platform-specific models is shared	T2
281	Publish data stream into a platform	T1

Table 29: Requirements vs test mapping

The following table provides an overview of the relation between the scenarios and the test(s) that validate their implementation.

ID	Scenario name	Covered by
2	IoT support for transport planning and execution	T1
29	Reliable control of robotic cranes and trucks in port terminals	T1
33	Heterogeneous Platforms Methodology-driven Integration	T2

Table 30: Scenario vs test mapping

3.3.2.4 Test environment

Introduction

To test the functionality of the OM2M component in combination with the IoT framework a representative test system is needed. The test system needs to approach the "real world" as much as possible. The pilot setups must be recreated and proven. This chapter will describe this environment and the used hardware, software, tools and platforms.

Test environment

This paragraph describes the test environment and the complete system setup used during this FAT.

3.3.2.5 Test setups, tools, hooks and probes

This paragraph describes the test setups, tools, hooks and probes used in the by this document defined tests.

A test setup defines a setup used for a test, and describe the used system parts, interface etc.

Tools define the used COTS tooling that will be used during testing e.g. Logic analyzers, packet sniffer like e.g. Wireshark etc.

Hooks are used to inject data or control parts of the system used to get the system to handle a needed scenario. This can be hidden, debug or service interfaces which contain the needed functionality.

Probes are used to gather logging from parts of the system under test to provide system feedback and to prove that the system is handling the scenario as it should.

The following paragraphs will define the used setups, tools, hooks and probes which will be referred to by the test descriptions.

TS_01 Test setup 1

The software that will be tested in the lab consists of two parts: the IPE running in the OM2M Infrastructure Node and the OM2M bridge. Regarding the IPE, we will run this application for some hours and check if it is able to manage the data provided by the IoT devices, in particular the Sigfox tracers that will control the movement/placement of trucks in the port. The same application should also provide user-friendly graphical interface so that the user is able to monitor all the data. As concerned the OM2M bridge, some tests will be performed to check data retrieval from a different middleware platform.

TT_01 Test tool 1

We will use an API developing environment (Postman) in order to check the correctness of the resources retrieved from the Sensolus cloud and created in the OM2M server. Postman is a desktop application which can be used to send HTTP requests to a specific URL. So we can insert the URL of the OM2M server to retrieve and check the data.

TH_01 Test hook 1

We will use the Interworking Proxy Entity (IPE) to inject the data in the OM2M IN. The IPE can also be used to control some functionalities of the IoT devices.

TH_01 Test probe 1

The API developing environment Postman can be also used as test probe to verify if the IPE retrieves all the resources stored in the Sensolus cloud according to the APIs.

Test description

Test output log files… Folder "`Tx_Output`", prefix "`Tx.y.1_`"

The most important test consists in the verification of the correct functioning of the OM2M bridge. In particular, we will try to send requests from the native INTER-IoT middleware to the bridge which has to convert them in a understandable format for the OM2M middleware. Obviously, the OM2M bridge should be able to perform the opposite conversion. Therefore, other tests will involve the conversion of the response from a OM2M primitive response to a clear response for the INTER-IoT middleware.

The most common information flows of OneM2M: creation of Application Entities (AEs), containers, content instances, subscription and other kind of resources. These resources can later be read, updated or deleted by users registered in the Inter-IoT platform. The messages from the Inter-IoT MW2MW layer are translated by the bridge into HTTP request and responses using 'xml' format and sent to the OM2M Infrastructure Node (IN-CSE), as it is specified that only the IN can receive requests originating from outside the Service Provider domain.

3.3.2.6 Test description

3.3.2.7 Scenario 1

Use case 1

T1 OM2M IPE functioning / performances

ID	T1
Test	Data retrieval from the Sensolus cloud and provision of GUI
Type	System and application testing
Setup	Deployment of Stickntrack devices Enabling IPE in OM2M IN
Start	Registration of IoT devices as AE in the OM2M IN
Req.	[15], [16], [23], [26], [27], [31], [42], [52], [53], [70], [108], [111], [123], [127], [153], [154], [168], [169], [281]
Input	Setting of a timer for the periodic retrieval of sensor data Enable Graphical interface for interaction between end user and stored data
Output	Check the creation of the resource in IN-CSE Check the good functioning of the GUI Check the integration of the Google Maps interface for positioning
Logs	Log of the IN-CSE

Outcome	Pass / Fail

T2 OM2M bridge testing

ID	T2
Test	Data collection from Inter-IoT
Type	Network communication
Setup	Connection between computers running MW2MW and OM2M
Start	Creation request from MW2MW
Req.	[14], [15], [16], [26], [27], [31], [95], [108], [111], [122], [123], [159], [178], [179], [180], [235], [255]
Input	Enable Graphical interface for interaction between end user and stored data Register through MW2MW REST API
Output	Check the HTTP messages exchanged between MW2MW and OM2M Check the creation of the resource in IN-CSE
Logs	Log of the IN-CSE
Outcome	Pass / Fail

3.3.2.8 Test outcome overview

Test outcome

The following table will provide an overview of the test result of all the performed tests in this FAT.

Test	Description	Outcome
1	Platform independency	Pass / Fail
2	Support of common IoT communication protocols	Pass / Fail
3	Inter-connection support	Pass / Fail
4	Device semantic definition	Pass / Fail
5	Remote device control	Pass / Fail
6	System security	Pass / Fail
7	Tools / libraries to support design	Pass / Fail
8	Heterogeneous information representation	Pass / Fail
9	API REST	Pass / Fail
10	Location of sensor and measurement is included in semantic models	Pass / Fail
11	Easy-to-use user interface	Pass / Fail
12	Robustness, resilience and availability	Pass / Fail
13	Open Source	Pass / Fail
14	Documentation	Pass / Fail
15	Extensibility of the use cases	Pass / Fail
16	Use of standards	Pass / Fail
17	Availability of sensor data	Pass / Fail
18	Cacheable Data	Pass / Fail
19	Time stamped event recording	Pass / Fail
20	Development support for systematic IoT platforms integration/interconnection	Pass / Fail
21	Provide an alert system	Pass / Fail
22	Methodology and tools to integrate a proprietary IoT platform	Pass / Fail
23	Inter Platform Semantic Mediator provides data and semantic interoperability functionality	Pass / Fail
24	Inter Platform Semantic Mediator supports platform communication	Pass / Fail

25	Syntactic and semantics interoperability - Data format and semantics translation	Pass / Fail
26	Support of semantic modelling in the middleware layer	Pass / Fail
27	A common data model compatible with all platform-specific models is shared	Pass / Fail
28	Publish data stream into a platform	Pass / Fail
29	IoT support for transport planning and execution	Pass / Fail
30	Reliable control of robotic cranes and trucks in port terminals	Pass / Fail
31	Heterogeneous Platforms Methodology-driven Integration	Pass / Fail
FAT Outcome		**Pass / Fail**

Table 31: Test outcome overview

3.3.2.9 Integration ethics and security

Introduction

During the Ethical Review it was requested that ethics had to be included within the project scope. This section addresses recommendations and proposed solutions to the ethical and security points of the integration.

Inter-OM2M

For the moment, our research team is not collecting sensitive data related to people such that we are not under the law of the protection of the privacy of individuals.

Nevertheless, we will work on several security features to add to the system. The security of the data collected in the VUB OM2M server is an essential aspect for the privacy of the end customer. Activity and location information should be kept hidden from any external entity which tries to get access to them illegally. To secure the data, we will include some security features to both access the data (using access policies) and the communication through the OM2M Bridge. In addition, we will check the possibility to provide integrity of the data whenever a legitimate user requires access to this data using the OM2M Bridge.

3.3.3 Third Party: INTER-HARE

The INTER-HARE project is intended to design a new LPWAN technology flexible enough to transparently encompass both LPWAN devices and multiple so-called *low-power local area networks* (LPLANs) while ensuring overall system's reliability. A cluster-tree network, is created, where the LPWAN acts not only as data collector, but also as backhaul network for several LPLANs, as shown in Figure 37.

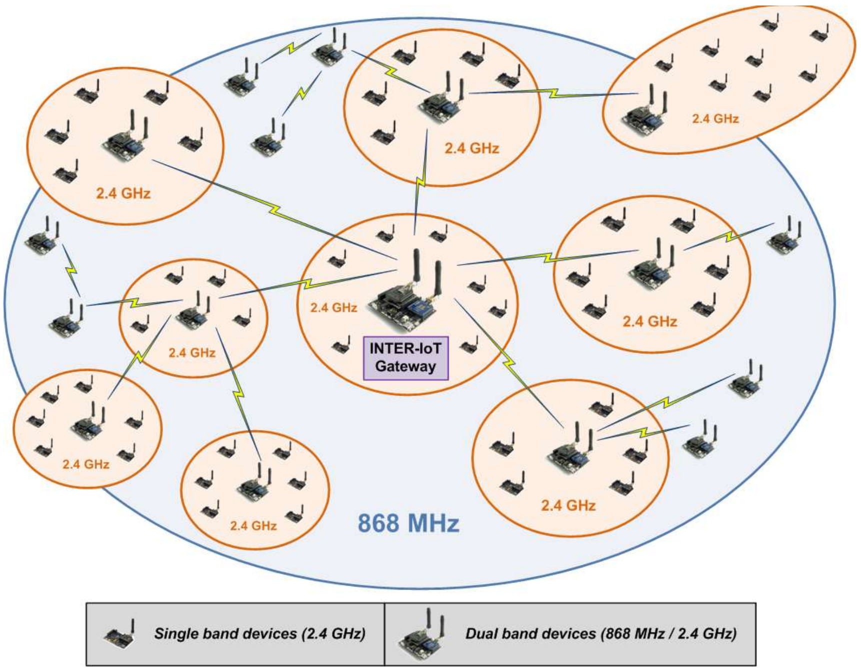

2.4 GHz
INTER-IoT Gateway
868 MHz
Single band devices (2.4 GHz)
Dual band devices (868 MHz / 2.4 GHz)

Figure 37: INTER-HARE network environment

Communication within the LPWAN is based on the HARE protocol stack, ensuring transmission reliability and low energy consumption by adopting uplink multi-hop communication, self-organization, and resilience. Under these premises, LPWAN boundaries are extended beyond typical 868 MHz coverage range and easily integrate devices coming from adjacent/overlapping 2.4 GHz LPLANs. Use of separated frequency bands in overlapping networks results in an overall reduction of interferences. Lastly, thanks to the hierarchic system proposed, scalability is enforced by a management based on subnetworking techniques.

3.3.3.1 General considerations

The INTER-HARE platform is conceived as an innovative evolution of HARE protocol stack and, according to the classification of multiprotocol/multiband systems, it can be considered as a concurrent multiprotocol. In INTER-HARE, the INTER-IoT gateway (GW) and the cluster-heads (CH) of each LPLAN share the same protocol stack operating at 868 MHz and, at the same time, these CHs and the corresponding data acquisition devices (DADs) also share the same protocol stack, in this case operating at 2.4 GHz.

The INTER-HARE transport network conceives end devices as elements controlled by the GW by means of beacons, so that these are first received by CHs and then immediately retransmitted at a different frequency band in order to be listened by DADs. This centralized approach allows DADs to remain asleep the majority of the time, so that their single concern is to be awake enough in advance to listen to the next beacon. Network synchronization is thus achieved and allows the GW to ask for specific data and/or distribute configuration changes easily.

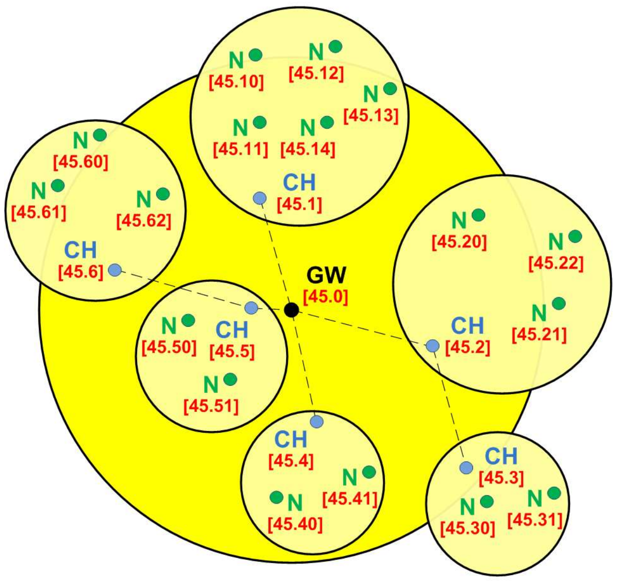


Figure 38: Example of INTER-HARE transport network.

The GW is considered to be appropriately placed close to a power source or an energy harvesting solution. Therefore, it may always stay in an active state and is provided with the ability to directly communicate (i.e., via single-hop communications) with any CH of the network through unicast and/or broadcast messages as well as to redirect gathered data to other networks or the Internet.

Conversely, CHs can take advantage of their neighbors to create multi-hop paths over which data is transmitted to the GW by means of lower transmission power levels. Depending on their position within these paths, CHs of the LPWAN are ideally organized into rings, as shown in Figure 38. The number of hops to reach the GW determines the ring number (i.e., CHs from ring 2 need two hops to reach the GW).

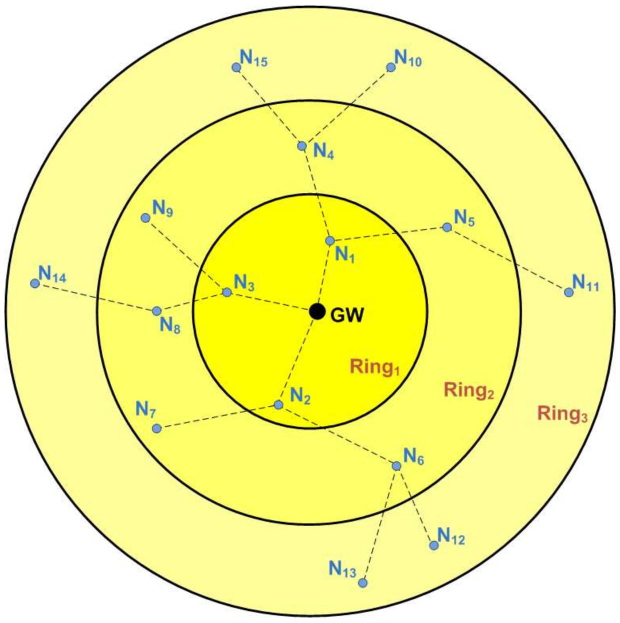


Figure 39: Ring structure of the LPWAN.

Each uplink data transmission phase (consisting of one or more transmission windows) begins with a beacon signal from the GW. Transmission windows are in turn virtually split into as many TDMA slots as network rings, so that CHs are only active during their own slot (for transmitting data) and the previous one belonging to their children5 (for receiving data).

The first slot is allocated to the highest ring and the rest are scheduled consecutively. Data received by CHs is aggregated to that generated by themselves (i.e., data previously received from STAs of their LPLAN), and finally sent to the corresponding parent at the minimum power level which ensures reliable communications. This process is repeated as many times as rings the network has.

The correct reception of data transmissions at the GW is acknowledged with a broadcast message, so that CHs are not only aware of their own end-to-end reliability, but also of those CHs in the same path to the GW. These acknowledgment beacons, together with the information obtained from their adjacent nodes, allow CHs to decide whether they should remain awake to perform retransmissions of lost network packets.

Network association (also started by a beacon) remains stable until a change in the topology is detected or the mechanism is reset by the GW. Nevertheless, the agreed transmission power between adjacent nodes in the association phase is constantly monitored and adjusted in a decentralized way in order to reduce the energy consumption.

Hardware platform

The hardware used for the data acquisition devices, the cluster-heads and the part of the INTER-IoT gateway corresponding to the transport network contain a USB interface that may be easily connected to a computer for debugging or configuration purposes. These devices, also known as *motes*, endow sensors with processing and communication abilities, so that they do not only take environmental data, but also send the information to the base station. Typical components of these devices are:

- A CPU
- Flash memory
- Separate SW program memory
- An optional sensor board with several sensors: light, humidity, pressure, etc.
- Radio module to communicate with other motes
- ADC: Analog-to-Digital converter
- Batteries

For this project, the Zolertia RE-Mote (Revision B) device has been chosen due to its ability to easily communicate with other devices by using any of its two available frequency bands: 868 MHz and 2.4 GHz. A complete description of the main features of the Zolertia RE-Mote can be found in ***Annex A: Zolertia RE-Mote***. For more information, we address the reader to check its datasheet.

[5] *Children* refers to all CHs of an adjacent higher ring from which an CH receives packets. Similarly, parent refers to that CH from an adjacent lower ring to which a CH transmits its own packets (after aggregating the ones from its children) in its way to the GW.

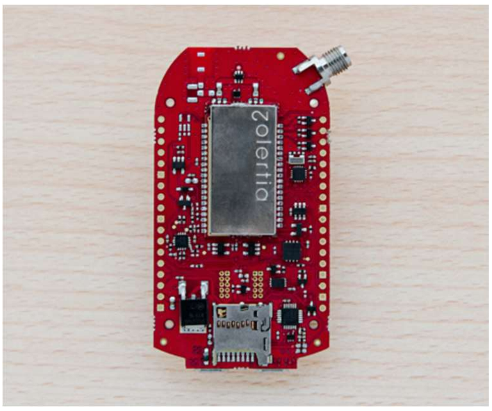


Figure 40: Zolertia RE-Mote.

Some other market-available devices with dual-band operation have been considered, such as Libelium Waspmote[6], Laird RM1XX[7], and WEPTECH WEP-6LoWPAN-IoT-GW[8]. However, none of them offer provide the same functionalities as the Zolertia RE-Mote in terms of flexibility, market availability, multi-threading operation and technical support.

Software platform

Among the vast quantity of current OS for the IoT, only the four most known have been considered for the current project: Contiki, TinyOS, FreeRTOS, and RIOT. As for the number of publications (in 2010), the percentage of articles related to each operating system included in the main scientific and engineering online databases (IEEE Xplore, ACM Digital Library and Science Direct) are the following: 81% TinyOS, 9% Contiki.

The main features of the Operating Systems have been summarized in Table 32, including their minimum requirements in terms of RAM and ROM, usage for a basic application, support for programming languages, multi-threading, MCUs without Memory Management Unit (MMU), modularity, real-time behavior, and availability of a simulator.

OS	Min. RAM	Min. ROM	C Support	C++ Support	Multi-threading	MCU w/o MMU	Modula-rity	Real-Time	Simulator
Contiki	< 2 kB	< 30 kB	Yes	No	Partial	Yes	Partial	Partial	Yes (Cooja)
TinyOS	< 1 kB	< 4 kB	No	No	Partial	Yes	No	No	Partial (TOSSIM)
FreeRTOS	< 1 kB	~ 10 kB	Yes	No	No	Yes	Yes	Yes	Partial
RIOT	~ 1.5 kB	~ 5 kB	Yes	Yes	Yes	Yes	Yes	Yes	No

Table 32: Key characteristics of Contiki, TinyOS, FreeRTOS, and RIOT

Although it has the best scores in the analyzed parameters, RIOT suffers from a lack of availability of a simulator. This makes its election unsuitable for the INTER-HARE platform, as

[6] Libelium Waspmote - http://www.libelium.com/products/waspmote/

[7] RM1XX Modules with LoRa and BLE - https://www.lairdtech.com/products/rm1xx-lora-modules

[8] Saker 6LoWPAN IoT Gateway - https://www.weptech.de/en/6lowpan/gateway-saker.html

the complexity of the system and the great quantity of nodes per network requires preliminary and comprehensive simulations to complete the design of the different protocols.

In this sense, the most complete simulator is Cooja, from Contiki OS. Actually, Cooja is the single emulator of the list, so that programming code used for simulations is the same that eventually loaded in devices, resulting in more accurate simulations and faster deployment times.

As for the programming model, Contiki is again the best option thanks to its combination of protothreads and events. In this way, it can execute easily multiple processes or threads concurrently, which in turn can be interrupted by pre-programmed events by I/O interruptions, for instance. Contiki, together with FreeRTOS and RIOT is written in C language programming. TinyOS, on its behalf, uses its particular C dialect (nesC), which supposes a steeper learning curve for programmers.

All considered Operating Systems have plenty of supported hardware platforms. However, the great amount of publications and deep knowledge of the UPF researching group in some of them (TelosB, MicaZ, Arduino-based and Zolertia Z1) has favored the election of an Operating System compatible with these platforms.

Due to the critical issue of the power management, the ability of controlling the power consumption of the mote has been another factor taken into account. In this sense, the most important techniques are intended to save power in nodes and components, by forcing them to go into a low-power mode when not interacting with the rest of the network. Another important point for designing purposes is the availability of techniques offered to estimate energy consumption of sensor nodes, such as the *powertrace* application of Contiki.

The supporting most active communities are the TinyOS development group, with more than 10 new releases in a decade, support for 12 different platforms and an annual TinyOS technology exchange developer meeting, and the Contiki group, with seven releases and a development team composed of people from prestigious companies and research institutions.

OS	Latest Release	Date	Source model	License
Contiki	3.0	26 August 2015	Open source	BSD
TinyOS	2.1.2	20 August 2012	Open source	BSD
FreeRTOS	8.2.3	16 October 2015	Source available	Modified GPL
RIOT	2016.04	22 April 2016	Open source	LGPLv2

Table 33: Latest releases of the considered Operating Systems

Lastly, another factor which was considered before choosing the best platform was the degree of software content being up-to-date. As it can be seen in Table 33, the oldest current version is the one from TinyOS, which dates from 2012. The rest of considered WSNs have rather up-to-date functional versions.

After analyzing all these factors, we have opted for the open source operating system Contiki OS[9] that gives developers and researchers flexibility to provide low-capable devices with novel

[9] Contiki Operating System main webpage: http://www.contiki-os.org/

communication mechanisms. Besides, its powerful emulator Cooja gives us a great deal of flexibility to design and test our custom protocols intended to improve the INTER-HARE transport network performance as well as to minimize its power consumption.

Physical layer

The physical layer of the wireless communications within the INTER-HARE transport network is based on the technology provided by the Texas Instruments CC2538 microcontroller + transceiver, and the Texas Instruments CC1200 transceiver embedded in the Zolertia RE-Mote devices, compatible with the IEEE 802.15.4 standard.

TI CC2538 Microcontroller

The CC2538 is the ideal wireless microcontroller System-On-Chip (SoC) for high-performance ZigBee applications. The device combines a powerful ARM Cortex-M3-based MCU system with up to 32KB on-chip RAM and up to 512KB on-chip flash with a robust IEEE 802.15.4 radio. This enables the device to handle complex network stacks with security, demanding applications, and over-the-air download.

Thirty-two GPIOs and serial peripherals enable simple connections to the rest of the board. The powerful hardware security accelerators enable quick and efficient authentication and encryption while leaving the CPU free to handle application tasks. The multiple low-power modes with retention enable quick startup from sleep and minimum energy spent to perform periodic tasks. For a smooth development, the CC2538 includes a powerful debugging system and a comprehensive driver library. To reduce the application flash footprint, CC2538 ROM includes a utility function library and a serial boot loader.

TI CC2538 Transceiver

The Texas Instruments CC2538 radio operates in the frequency band of 2394-2507 MHz for compliance with IEEE 802.15.4 standard. The CC2538 device family provides a highly integrated low-power IEEE 802.15.4-compliant radio transceiver. The radio subsystem provides an interface between the MCU and the radio which makes it possible to issue commands, reads status, and automatizes and sequence radio events. The radio also includes a packet-filtering and address-recognition module.

The current porting of the TI CC2538 RF transceiver in Contiki OS only allows the whole platform to transmit at **250 kbps** when using the 2.4 GHz frequency band, which enables -97 dBm of receiving sensitivity. The programmable output power of the TI CC2538 RF transceiver has a range from more than 30 dB (from -24 dBm to 7 dBm), with their corresponding current consumption depicted in Table 34. It is worth noting here than only the current consumption values for the power outputs of 0 dBm and 7 dBm are provided by the TI CC2538 datasheet.

Selectable TX power output	Current consumption
7 dBm	34 mA
5 dBm	*N/A*
3 dBm	*N/A*
1 dBm	*N/A*
0 dBm	24 mA
-1 dBm	*N/A*
-3 dBm	*N/A*

-5 dBm	*N/A*
-7 dBm	*N/A*
-9 dBm	*N/A*
-11 dBm	*N/A*
-13 dBm	*N/A*
-15 dBm	*N/A*
-24 dBm	*N/A*

Table 34: Current consumption value of the transmitting operational mode depending on the selected output power

TI CC1200 Transceiver

Texas Instruments CC1200 is a fully-integrated single-chip radio transceiver designed for high-performance at very low-power and low-voltage operation in cost-effective wireless systems. All filters are integrated, removing the need for costly external SAW and IF filters. The device is mainly intended for the ISM (Industrial, Scientific and Medical) and SRD (Short Range Device) frequency bands at 164-190 MHz, 410-475 MHz, and 820-950 MHz.

The CC1200 provides extensive hardware support for packet handling, data buffering, burst transmissions, clear channel assessment, link quality indication, and Wake-On-Radio. The CC1200 main operating parameters can be controlled via an SPI interface. In a typical system, the CC1200 will be used together with a microcontroller and a few external passive components.

As default the Zolertia RE-Mote uses the IEEE 802.15.4g mandatory mode for the 868 MHz band, configured for 2-GFSK modulation, 50 kbps data rate and with 33 channels available. The current porting of the TI CC1200 RF transceiver in Contiki OS only allows the whole platform to transmit at 50 kbps when using the 868 MHz frequency band, which enables -109 dBm of receiving sensitivity.

The TI CC1200 library of Contiki allows up to 31 different output power values, ranging from -16 dBm to 14 dBm, with steps of 1 dB. It provides the platform with a high flexibility to adapt the different multi-hop links to the most appropriate output power depending in different factors such as the distance between stations, the remaining battery or the channel conditions.

Due to the disparity of current consumption values among different information sources, any calculation from this document will use the measures obtained from Texas Instruments official tests. In these tests is described how to combine the CC2538 and the CC1200 devices in the same design. The most interesting result is that from Figure 40Figure 41 where it is plotted the real current consumption of the whole system depending on the selected output power.

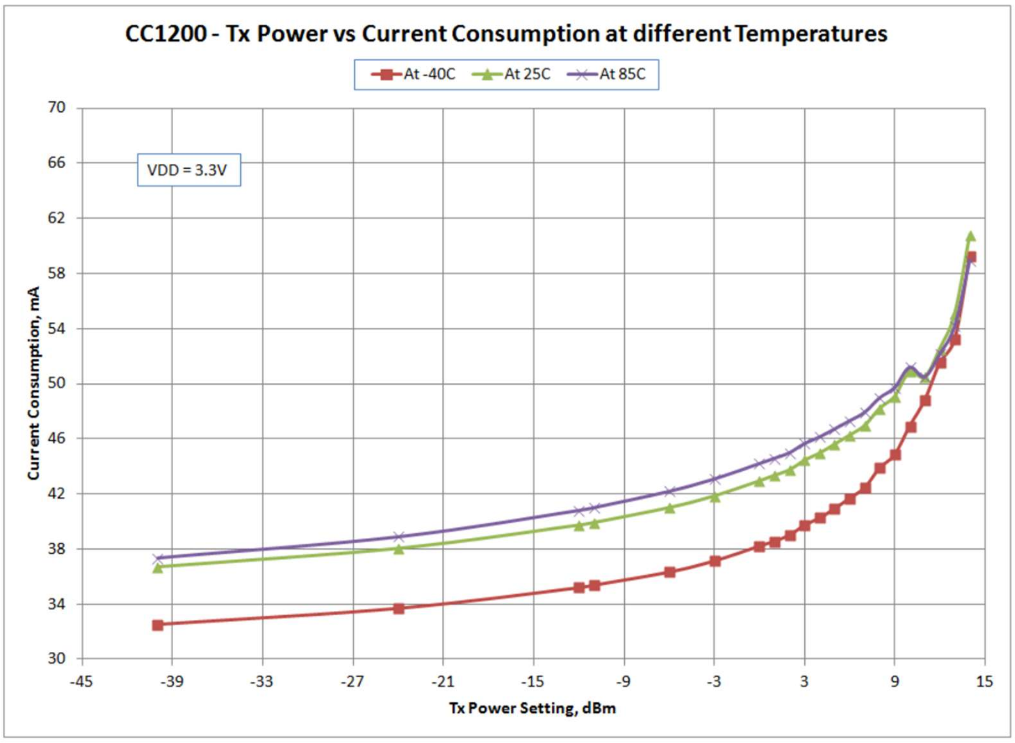


Figure 41: CC1200 – TX Current Consumption vs TX Power at Different Temperatures

From Figure 41 where all parameters are measured on the CC2538+CC1200 Combo EM reference design at an Antenna connector (T_C = 25°C, VDD = 3.3 V, f_c = 915 MHz if nothing else is stated.), the following table has been filled with real current consumption values of the TI CC1200 transceiver operating at 25°C:

Transmission power setting	Current consumption	Transmission power setting	Current consumption
-16 dBm	39 mA	0 dBm	43 mA
-15 dBm	39.2 mA	1 dBm	43.5 mA
-14 dBm	39.4 mA	2 dBm	44 mA
-13 dBm	39.6 mA	3 dBm	44.5 mA
-12 dBm	39.8 mA	4 dBm	45 mA
-11 dBm	40 mA	5 dBm	45.5 mA
-10 dBm	40.2 mA	6 dBm	46 mA
-9 dBm	40.4 mA	7 dBm	47.5 mA
-8 dBm	40.6 mA	8 dBm	48.5 mA
-7 dBm	40.8 mA	9 dBm	49 mA
-6 dBm	41 mA	10 dBm	51 mA
-5 dBm	41.3 mA	11 dBm	50.5 mA
-4 dBm	41.6 mA	12 dBm	52 mA
-3 dBm	42 mA	13 dBm	55 mA
-2 dBm	42.3 mA	14 dBm	61 mA
-1 dBm	42.6 mA		

Table 35: Current consumption value of the transmitting operational mode depending on the selected output power

Range coverage

To ensure the communication link between the different elements of the transport network is responsibility of the chosen RF transceiver. One advantage of using lower frequencies (868 MHz even more than 2.4 GHz) is that signals have better penetration, meaning they pass through objects such as walls with less attenuation. This effect results in larger range coverage.

The wireless radio channel poses a severe challenge as a medium for reliable high-speed communication. It is not only susceptible to noise, interference, and other channel impediments, but these impediments change over time in unpredictable ways due to user movement. Three mutually independent, multiplicative propagation phenomena can usually be distinguished: large-scale path loss, shadowing and multipath fading:

- **Large-scale path loss:** The 'large-scale' effects of path losses cause the received power to vary gradually due to signal attenuation determined by the geometry of the path profile in its entirety. This is in contrast to the local propagation mechanisms, which are determined by terrain features in the immediate vicinity of the antennas.
- **Shadowing:** Shadowing is a 'medium-scale' effect caused by obstacles between the transmitter and receiver that absorb power.
- **Multipath fading:** Multipath propagation leads to rapid fluctuations of the phase and amplitude of the signal due to the constructive and destructive addition of multipath signal components.

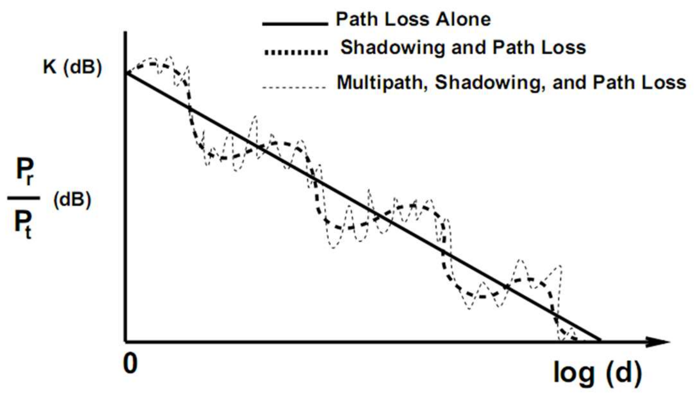


Figure 42: Path Loss, Shadowing and Multipath versus Distance [1]

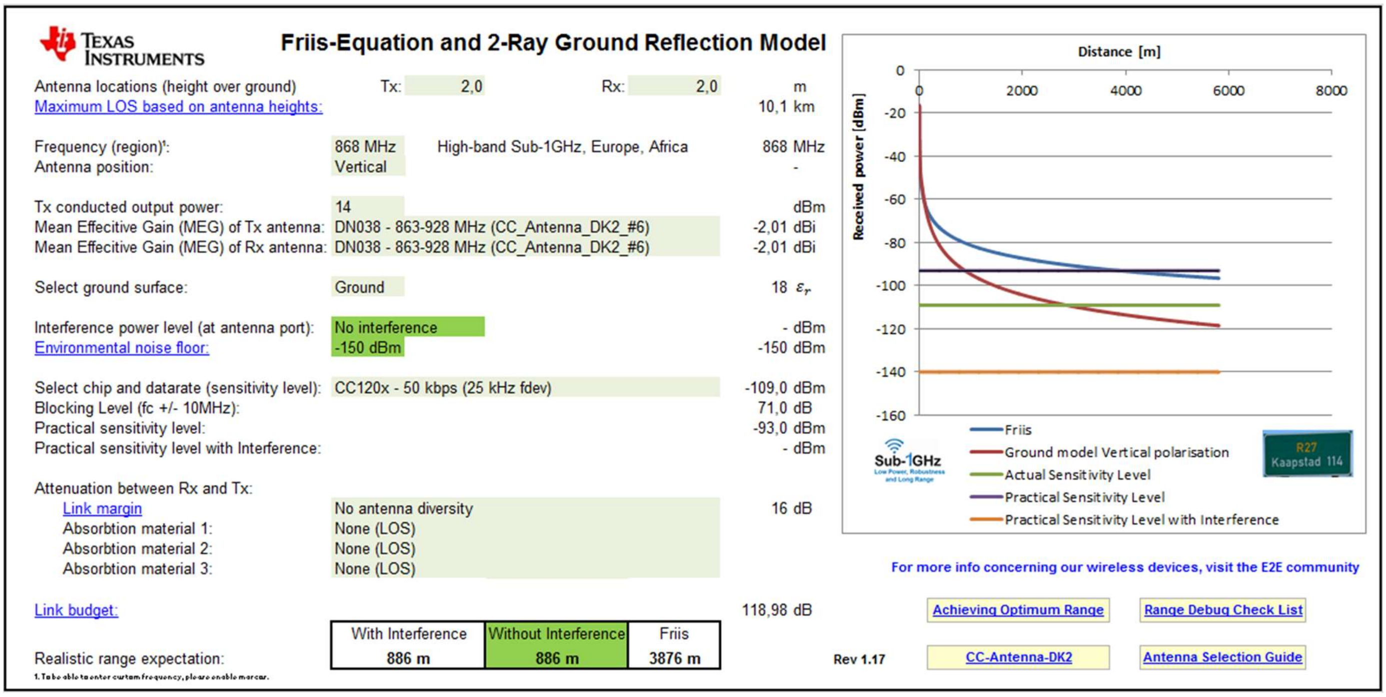
The Texas Instrument range coverage calculator provides a theoretical calculation of the range coverage of different TI RF transceivers in function of different parameters, such as the frequency, the TX and RX antenna locations, the data rate, or even the presence of different materials in the signal path. It is based on the Friis Equation[10] and can also characterize the shadowing effects of up to 3 materials placed between the transmitter and the receiver.

This tool has been used to prove the feasibility of using the TI CC1200 and the TI CC2538 transceivers in the INTER-HARE platform. From all the output power values of transceivers, only the ones which can be selected in the corresponding Contiki OS porting of these devices have been used in the aforementioned tool, with the resulting values compiled in Table 36 and Table 37. Some captures of the tool are also shown in Figure 43 and Figure 44.

From results it can be noticed the difference of range coverage between both transceivers; while TI CC1200 is far above 100 m. in almost any configuration, TI CC2538 can hardly

[10] Friis transmission equation (definition from Wikipedia) - https://en.wikipedia.org/wiki/Friis_transmission_equation

achieve more than 150 m. When comparing the most favorable configuration from both devices, packets can be transmitted up to 886 m. in the TI CC1200 transceiver, whereas the TI CC2538 one only has 175 m. of range coverage. Lastly, two important design variables should be taken into account in future testbeds and/or real experimentations: the selected TX power output of the transceiver as well as the antenna locations.


Texas Instruments

Friis-Equation and 2-Ray Ground Reflection Model

Antenna locations (height over ground)	Tx: 2,0	Rx: 2,0	m
Maximum LOS based on antenna heights:			10,1 km
Frequency (region)¹:	868 MHz	High-band Sub-1GHz, Europe, Africa	868 MHz
Antenna position:	Vertical		-
Tx conducted output power:	14		dBm
Mean Effective Gain (MEG) of Tx antenna:	DN038 - 863-928 MHz (CC_Antenna_DK2_#6)		-2,01 dBi
Mean Effective Gain (MEG) of Rx antenna:	DN038 - 863-928 MHz (CC_Antenna_DK2_#6)		-2,01 dBi
Select ground surface:	Ground		18 ε_r
Interference power level (at antenna port):	No interference		- dBm
Environmental noise floor:	-150 dBm		-150 dBm
Select chip and datarate (sensitivity level):	CC120x - 50 kbps (25 kHz fdev)		-109,0 dBm
Blocking Level (fc +/- 10MHz):			71,0 dB
Practical sensitivity level:			-93,0 dBm
Practical sensitivity level with Interference:			- dBm
Attenuation between Rx and Tx:			
Link margin	No antenna diversity		16 dB
Absorbtion material 1:	None (LOS)		
Absorbtion material 2:	None (LOS)		
Absorbtion material 3:	None (LOS)		
Link budget:			118,98 dB

Realistic range expectation:	With Interference	Without Interference	Friis
	886 m	886 m	3876 m

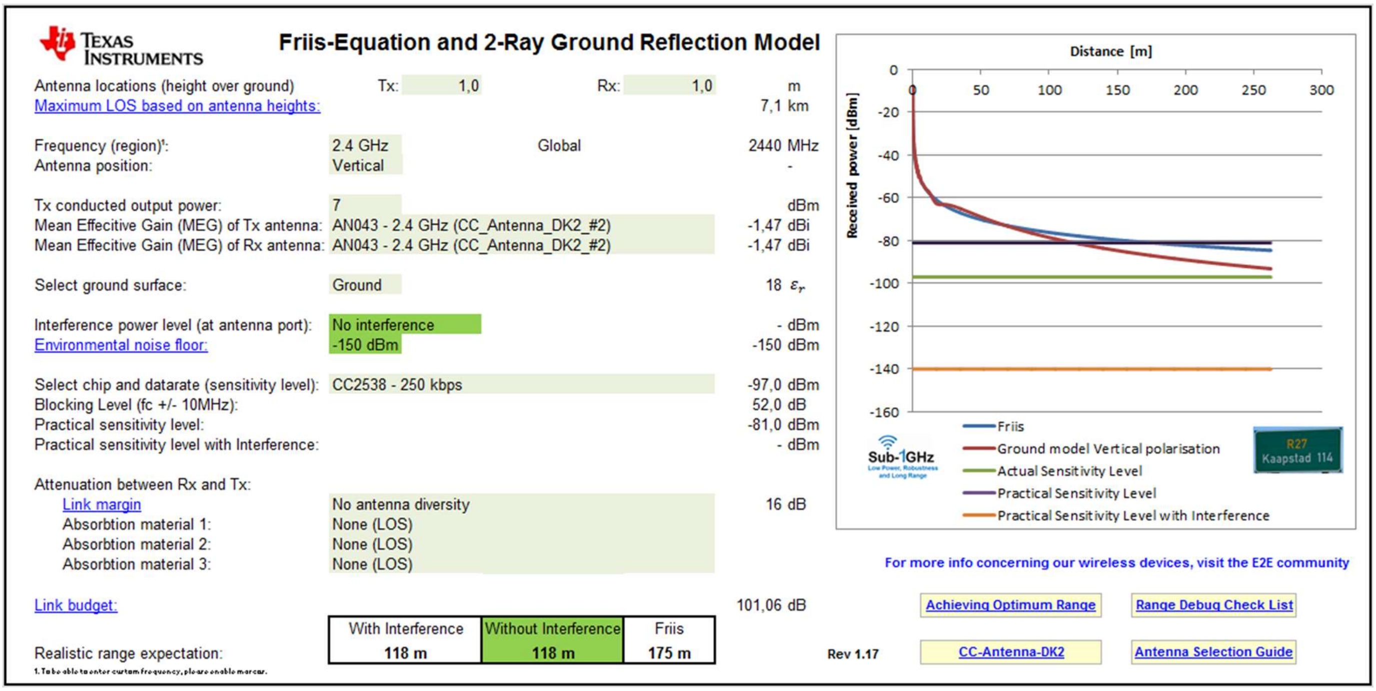
1. To be able to enter custom frequency, please enable macros.

For more info concerning our wireless devices, visit the E2E community

Achieving Optimum Range | Range Debug Check List | CC-Antenna-DK2 | Antenna Selection Guide

Rev 1.17

Figure 43: Range coverage for the TI CC1200 transceiver


Texas Instruments

Friis-Equation and 2-Ray Ground Reflection Model

Antenna locations (height over ground)	Tx: 1,0	Rx: 1,0	m
Maximum LOS based on antenna heights:			7,1 km
Frequency (region)¹:	2.4 GHz	Global	2440 MHz
Antenna position:	Vertical		-
Tx conducted output power:	7		dBm
Mean Effective Gain (MEG) of Tx antenna:	AN043 - 2.4 GHz (CC_Antenna_DK2_#2)		-1,47 dBi
Mean Effective Gain (MEG) of Rx antenna:	AN043 - 2.4 GHz (CC_Antenna_DK2_#2)		-1,47 dBi
Select ground surface:	Ground		18 ε_r
Interference power level (at antenna port):	No interference		- dBm
Environmental noise floor:	-150 dBm		-150 dBm
Select chip and datarate (sensitivity level):	CC2538 - 250 kbps		-97,0 dBm
Blocking Level (fc +/- 10MHz):			52,0 dB
Practical sensitivity level:			-81,0 dBm
Practical sensitivity level with Interference:			- dBm
Attenuation between Rx and Tx:			
Link margin	No antenna diversity		16 dB
Absorbtion material 1:	None (LOS)		
Absorbtion material 2:	None (LOS)		
Absorbtion material 3:	None (LOS)		
Link budget:			101,06 dB

Realistic range expectation:	With Interference	Without Interference	Friis
	118 m	118 m	175 m

1. To be able to enter custom frequency, please enable macros.

For more info concerning our wireless devices, visit the E2E community

Achieving Optimum Range | Range Debug Check List | CC-Antenna-DK2 | Antenna Selection Guide

Rev 1.17

Figure 44: Range coverage for the TI CC2538 transceiver

Selectable TX power output	Current consumption	Antenna locations (height over ground)	
		TX: 1 m. and RX: 1 m.	TX: 2 m. and RX: 2 m.
14 dBm	61 mA	652 m.	886 m.
13 dBm	55 mA	605 m.	825 m.
12 dBm	52 mA	561 m.	768 m.
11 dBm	50.5 mA	520 m.	716 m.
10 dBm	51 mA	482 m.	667 m.
9 dBm	49 mA	448 m.	622 m.
8 dBm	48.5 mA	415 m.	580 m.
7 dBm	47.5 mA	385 m.	541 m.
6 dBm	46 mA	357 m.	504 m.
5 dBm	45.5 mA	331 m.	471 m.
4 dBm	45 mA	308 m.	439 m.
3 dBm	44.5 mA	285 m.	410 m.
2 dBm	44 mA	265 m.	383 m.
1 dBm	43.5 mA	246 m.	358 m.
0 dBm	43 mA	228 m.	334 m.
-1 dBm	42.6 mA	212 m.	312 m.
-2 dBm	42.3 mA	197 m.	291 m.
-3 dBm	42 mA	183 m.	272 m.
-4 dBm	41.6 mA	170 m.	254 m.
-5 dBm	41.3 mA	158 m.	238 m.
-6 dBm	41 mA	146 m.	222 m.
-7 dBm	40.8 mA	136 m.	207 m.
-8 dBm	40.6 mA	126 m.	194 m.
-9 dBm	40.4 mA	118 m.	181 m.
-10 dBm	40.2 mA	109 m.	169 m.
-11 dBm	40 mA	102 m.	158 m.
-12 dBm	39.8 mA	94 m.	147 m.
-13 dBm	39.6 mA	88 m.	137 m.
-14 dBm	39.4 mA	82 m.	128 m.
-15 dBm	39.2 mA	76 m.	119 m.
-16 dBm	39 mA	70 m.	111 m.

Table 36: Range coverage computation for the TI CC1200 transceiver

Selectable TX power output	Power consumption	Antenna locations (height over ground)	
		TX: 1 m. and RX: 1 m.	TX: 2 m. and RX: 2 m.
7 dBm	34 mA	118 m.	175 m.
5 dBm	*N/A*	104 m.	139 m.
3 dBm	*N/A*	91 m.	111 m.

1 dBm	*N/A*	79 m.	88 m.
0 dBm	24 mA	74 m.	59 m.
-1 dBm	*N/A*	69 m.	57 m.
-3 dBm	*N/A*	55 m.	54 m.
-5 dBm	*N/A*	44 m.	44 m.
-7 dBm	*N/A*	35 m.	31 m.
-9 dBm	*N/A*	28 m.	28 m.
-11 dBm	*N/A*	19 m.	21 m.
-13 dBm	*N/A*	15 m.	18 m.
-15 dBm	*N/A*	14 m.	14 m.
-24 dBm	*N/A*	5 m.	5 m.

Table 37: Range coverage computation for the TI CC2538 transceiver

3.3.3.2 Link layer

Link layer of the *transport network* in the INTER-HARE platform is split into the MAC sublayer and the radio duty cycling (RDC) sublayer (see Table 38).

The MAC sublayer consists of two levels:

1. A time division multiple access (TDMA) scheme to divide the time in slots, and
2. A carrier sense multiple access with collision avoidance (CSMA/CA) technique performed by the group of STAs allocated into each slot (see Figure 45).

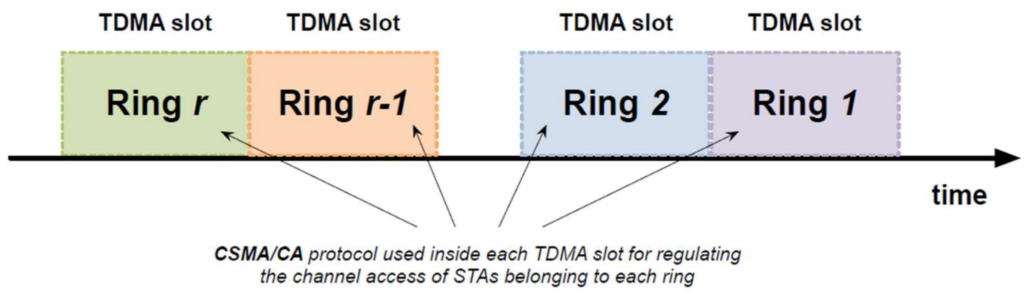

TDMA slot
TDMA slot
TDMA slot
TDMA slot
Ring *r*
Ring *r-1*
Ring 2
Ring *1*
time
CSMA/CA *protocol used inside each TDMA slot for regulating the channel access of STAs belonging to each ring*

Figure 45: MAC sublayer consisting of TDMA slots and CSMA/CA technique inside them

The number and duration of the TDMA slots, and the STAs allocated to each one are managed by the gateway, thus providing the platform with the necessary flexibility to encompass heterogeneous scenarios.

The RDC sublayer tries to keep the radio module turned off while providing enough rendezvous points for two nodes to be able to communicate with each other.

Frequency band	**MAC sublayer**		**RDC sublayer**
	Scheduling-based protocol	**Contention-based protocol**	
LPWAN 868 MHz	TDMA (with 1 slot per ring)	CSMA/CA	X-MAC
LPLAN 2.4 GHz	None	CSMA/CA	X-MAC / ContikiMAC

Table 38: Link layers used in the INTER-HARE platform

3.3.3.3 MAC sublayer

Scheduling-based protocol

The designed beaconing system has a double function: synchronizing the network devices and scheduling the different actions to be performed in a time-division multiplexing scheme. Beacons are first transmitted in the LPWAN by the INTER-IoT gateway at 868 MHz and immediately repeated by each one of the cluster-heads in their own LPLAN at 2.4 GHz.

Two types of beacons are used in the INTER-HARE platform: primary and secondary beacons. Both beacons include a timestamp, the time until the next beacon, and the next action to be taken by the network. Primary beacons are intended to be used for periodic data transmission and association purposes. Conversely, secondary beacons are only used for requests coming from the gateway and/or alarms generated by STAs. As shown in Figure 46, primary beacons can also perform downlink requests and handle uplink alarms.

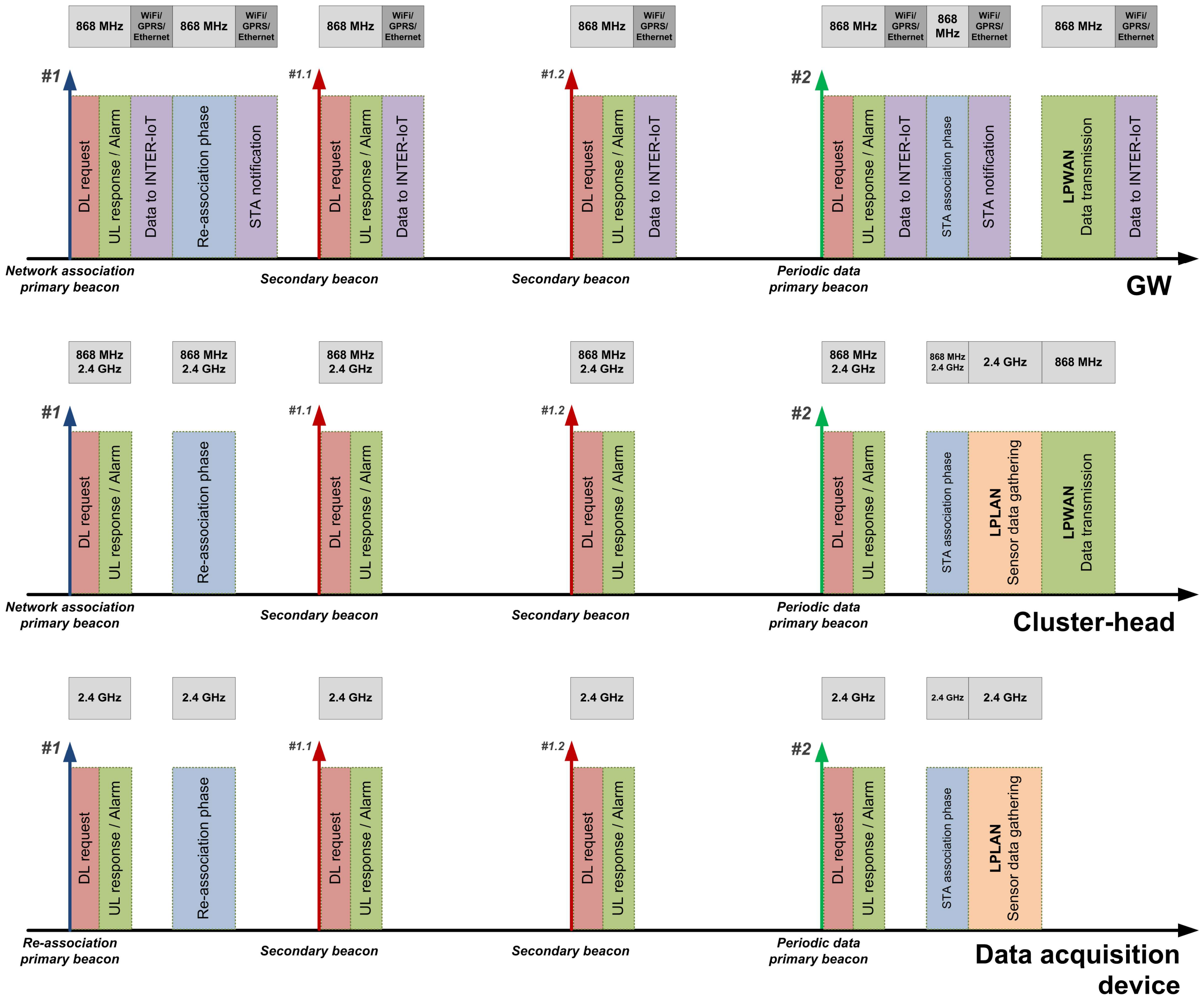


Figure 46: Beaconing system in the INTER-HARE platform

Time between two consecutive primary beacons and two consecutive secondary beacons is defined as T_p and T_s, respectively. Where $T_p = (k_s + 1) \cdot T_s$, being k_s the number of secondary beacons transmitted after every primary beacon. As it can be seen in Figure 47, there will be always a certain delay in the INTER-IoT physical gateway between the reception of a request order coming from the INTER-IoT system and its handling in the next beacon. This time period is called *scheduling delay*.

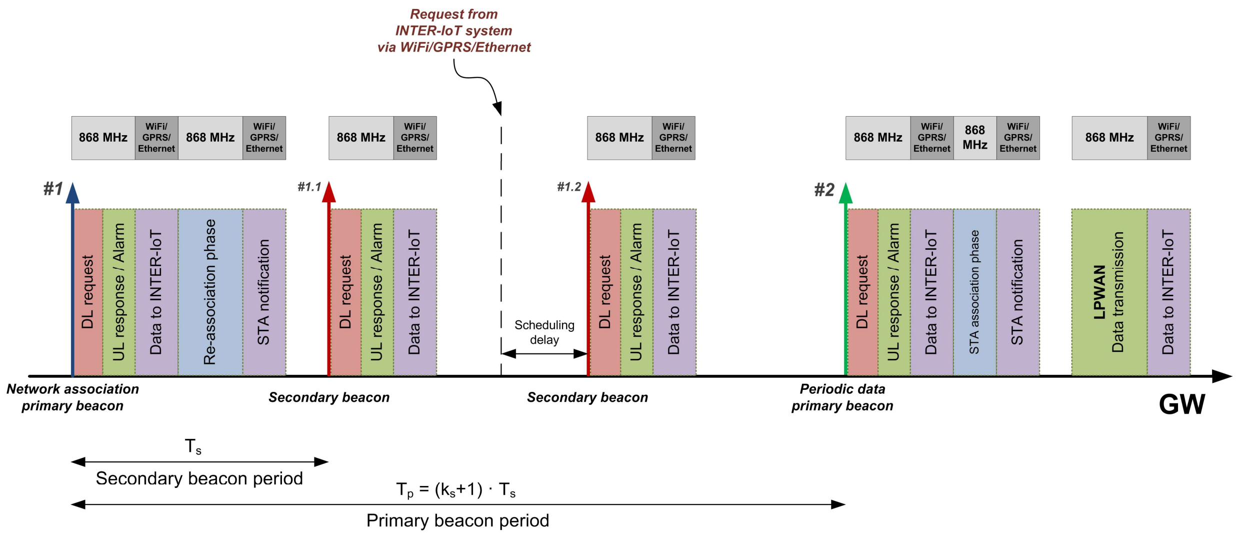


Figure 47: Primary and secondary beacon periods

Wakeup patterns

A wakeup pattern is a set of instructions generated by the GW which define the wakeup plan of its associated STAs over time periods. With the goal of minimizing the time STAs remain active (and, consequently, their energy consumption), two different wakeup patterns controlled by the GW are proposed according to the network's traffic flow.

The periodic wakeup pattern is suitable for listening to unicast and broadcast downlink communications from the GW, as it makes all STAs wake up at the same time. On the other hand, uplink communications follow a staggered wakeup pattern, which allocates different active periods to nodes belonging to adjacent rings with partial overlapping (as shown in Figure 48). Apart from reducing time STAs are awake during uplink communications; this method facilitates the implementation of data aggregation mechanisms.

Even though STAs have predetermined active periods, they can go to sleep even earlier in the transmitting (TX) time period if their parent has acknowledged all their data, or in the receiving (RX) time period after having received all data from their children.

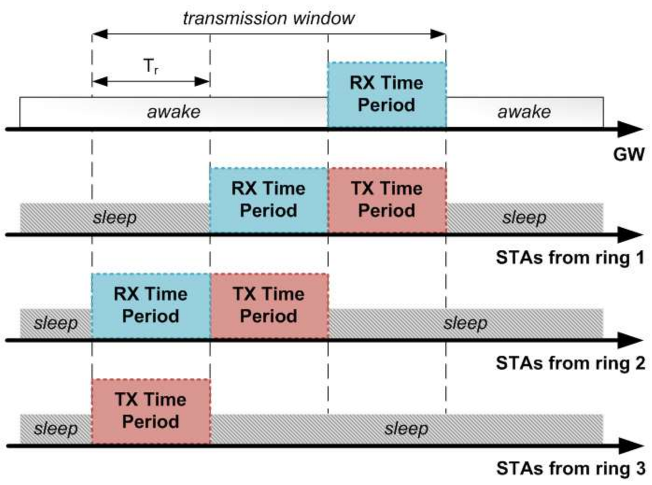


Figure 48: Example of a staggered wakeup pattern in a 3-ring LPWAN performing uplink communications.

Contention-based protocol

Carrier-sense multiple access with collision avoidance (CSMA/CA) is the default contention-based protocol within the MAC sublayer both for the LPWAN and for the different LPLANs11. The MAC sublayer receives incoming packets from the RDC sublayer and uses the RDC sublayer to transmit packets. If the RDC sublayer or the physical layer detects a radio collision, the MAC sublayer may retransmit the packet at a later point in time. The CSMA/CA mechanism retransmits packets if a collision is detected. It is also able to keep record of the number of retransmissions, collisions, deferrals, etc.

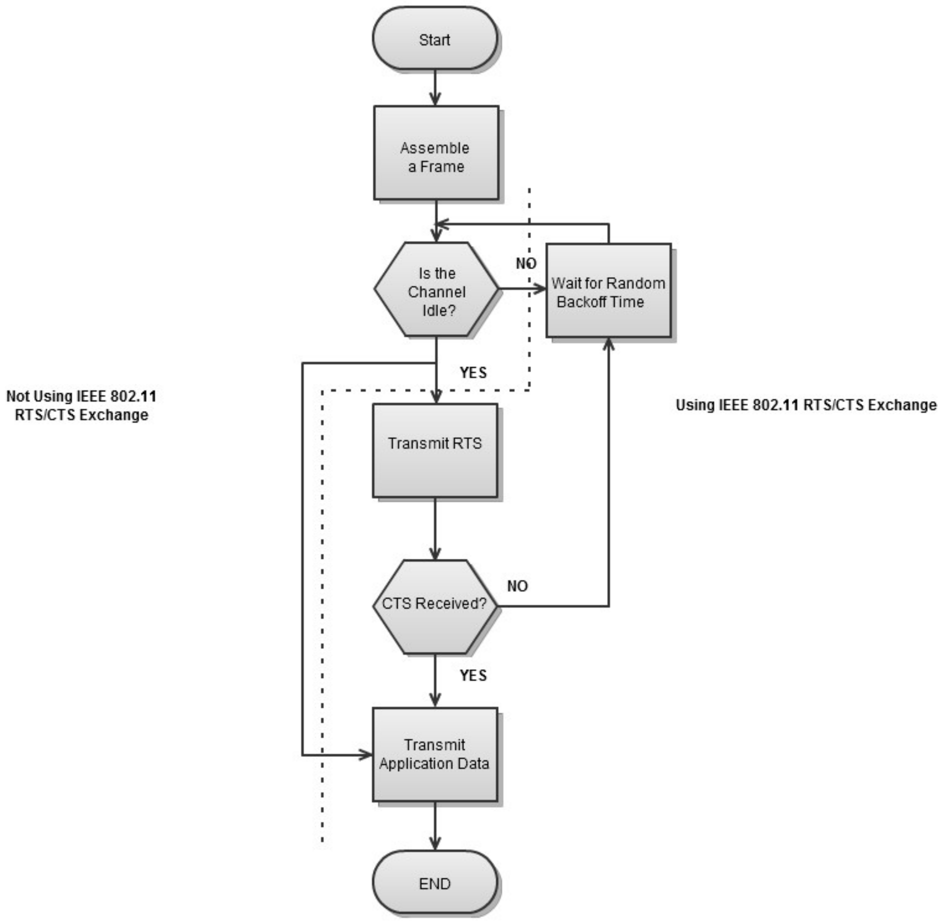

Start
Assemble a Frame
Is the Channel Idle?
NO
Wait for Random Backoff Time
YES
Not Using IEEE 802.11 RTS/CTS Exchange
Using IEEE 802.11 RTS/CTS Exchange
Transmit RTS
CTS Received?
NO
YES
Transmit Application Data
END

Figure 49: Simplified algorithm of CSMA/CA

Clear Channel Assessment (CCA) is a mechanism used to determine if a wireless channel is currently free. In wireless MAC protocols like the one implemented in the INTER-HARE platform, CCA is used to implement CSMA/CA: each node first listens to the medium to detect ongoing transmissions, and transmits its packet only if the channel is free, thus reducing the chance of collisions.

CCA is typically implemented by comparing the Received Signal Strength (RSS) obtained from the radio against a threshold. The channel is assumed to be clear if the RSS does not exceed the given threshold. As false negatives result in collisions and false positives cause increased latency, the choice of the threshold is critical.

[11] It is worth noting here that the RTS/CTS Exchange mechanism has not been considered in the current implementation of the CSMA/CA multiple access method.

3.3.3.4 RDC sublayer

Radio Duty Cycling (RDC) layer takes care of the sleep period of nodes. This is one of the most important elements of the link layer because it is the one responsible for deciding exactly when the packets will be transmitted and it is responsible for making sure that the node is awake when packets are to be received. Two possible options are considered for this task: X-MAC and ContikiMAC.

X-MAC

The Contiki OS implementation of X-MAC is the RDC sublayer chosen both for the LPWAN and for the different LPLANs. X-MAC is a power-saving MAC protocol in which senders use a sequence of short preambles (strobes) to wake up receivers (see Figure 50). Nodes turn off the radio for most of the time to reduce idle listening. They wake up shortly at regular intervals to listen for strobes. When a receiving node wakes up and receives a strobe destined to it, it replies with an acknowledgment indicating that it is awake.

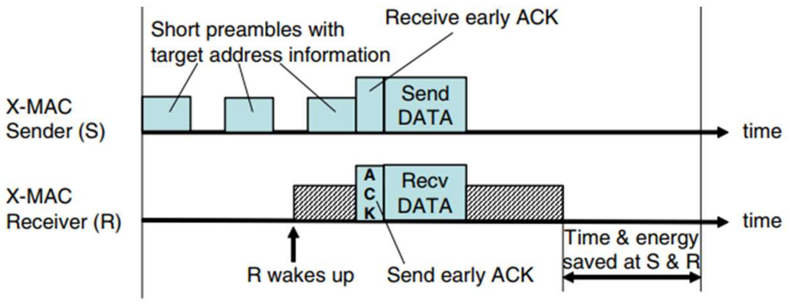


Figure 50: X-MAC's short preamble approach

ContikiMAC

ContikiMAC is considered as an alternative for transmitting packets in the LPLANs of the INTER-HARE project. ContikiMAC is a radio duty cycling protocol that uses periodical wake-ups to listen for packet transmissions from neighbors. If a packet transmission is detected during a wake-up, the receiver is kept on to be able to receive the packet. When the packet is successfully received, the receiver sends a link layer acknowledgment.

To transmit a packet, a sender repeatedly sends its packet until it receives a link layer acknowledgment from the receiver. Packets that are sent in broadcasts do not result in link-layer acknowledgments. Instead, the sender repeatedly sends the packet during the full wake-up interval to ensure that all neighbors have received it.

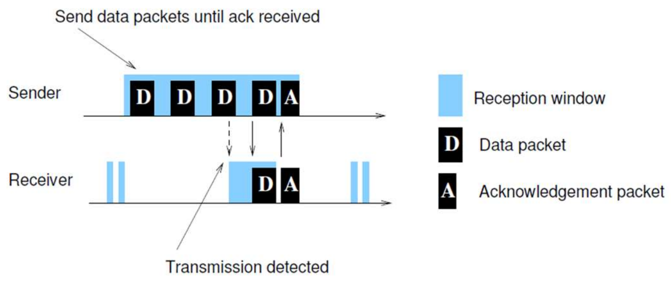


Figure 51: Unicast transmission in ContikiMAC

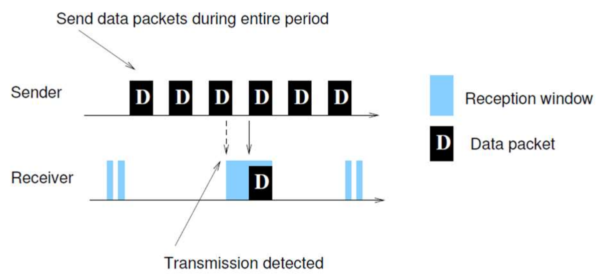


Figure 52: Broadcast transmission in ContikiMAC

3.3.3.5 Data transmission

Data transmission in the INTER-HARE platform can follow 3 different data delivery models, as described in: *query-driven model*, *event-driven model* and *continuous model.*

- **Query-driven model:** The sink initiates the communication by sending a request to one or some nodes to perform management, reconfiguration or probing tasks. Data flow is two-way: requests from the sink are downlink (DL) and unicast while responses are uplink (UL) and follow a multi-hop route.
- **Event-driven model:** In this case, the activation of an alarm in a sensor node generates an uplink (UL) data flow, which follows a multi-hop route.
- **Continuous model:** It is based on the continuous transmission of sensing data by nodes to a sink at a predefined rate. Therefore, the data flow is uplink (UL) and follows a multi-hop route.

Lastly, the data delivery model in which the three aforementioned systems coexist is called *hybrid*. In our case, INTER-HARE follows a hybrid data delivery model, as it implements all the possible options. The specific implementation of these data delivery models in the INTER-HARE platform is detailed in the following subsections.

Query-driven model

As shown in Figure 46, after any beacon emitted by the GW, there are always two reserved slots: one for allocating the requests of the GW, and another one for responses from data acquisition devices. As the transport network is formed by two tiers, all requests sent by the GW to a specific STA must go through its corresponding CH, which will act as a relay at a different frequency band. Similarly, once the targeted STA receives the request, it must wait for the response slot and emit its message through the intermediate CH.

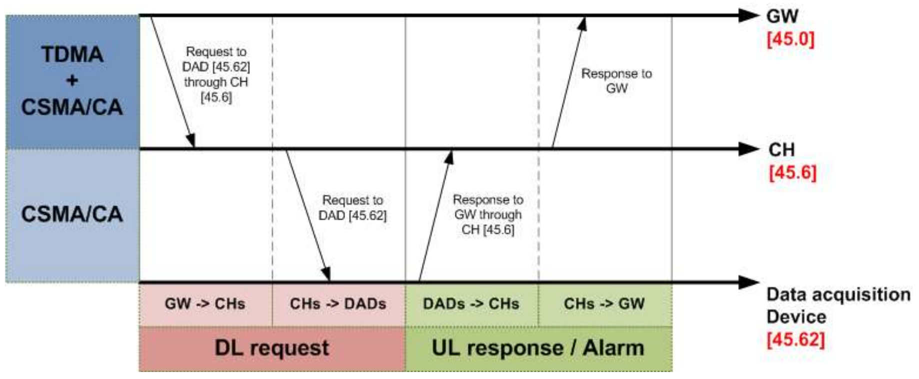


Figure 53: Diagram of the query-driven data delivery model operation

Queries requested by the GW can be for a single DAD or for a group of DADs (in our case, all STAs belonging to the same LPLAN). The first type of requests will be transmitted via *unicast* messages, while the second type will use a *multicast* message. In fact, this multicast message will be first transmitted from the GW to the corresponding CH by using a unicast message, and then, the CH will transmit a broadcast message to all the DADs of its LPLAN.

It is worth noting here that the query-driven model may also be used to modify some configuration parameters of the DADs or even to distribute some new firmware elements. Depending on the kind of update, these reconfiguration elements distributed by the GW may not require a response from DADs, or could only ask for a confirmation ACK message.

Event-driven model

As in the query-driven data delivery model, the event-driven model also uses the predetermined slots located after each beacon emitted by the GW. In fact, only the second slot is used, as a way to transmit the alarms generated since the last beacon.

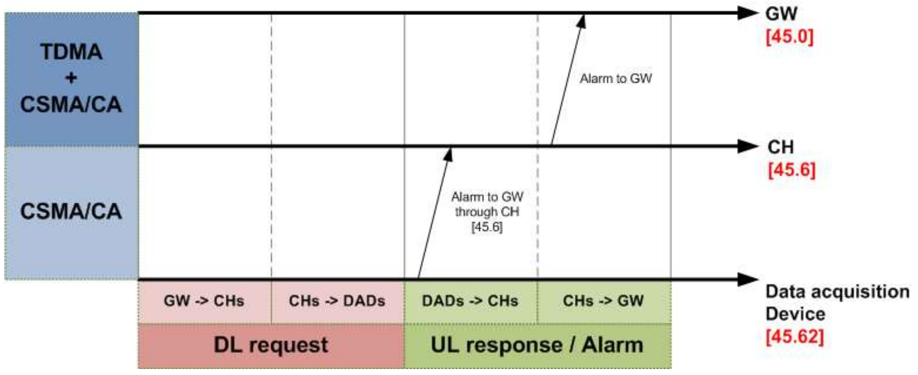


Figure 54: Diagram of the event-driven data delivery model operation

Each CH gathers all possible alarms from the LPLAN under its control, aggregates them and transmits them over the LPWAN until reaching the GW.

Continuous model

Each data transmission phase in the continuous data delivery model begins with a data primary beacon from the GW (see Figure 46). After the slots reserved to possible GW requests, alarms generated by STA or new STA associations; two specific slots are devoted to this action: one slot for LPLAN sensor data gathering and one slot for LPWAN data transmission (see Figure 55).

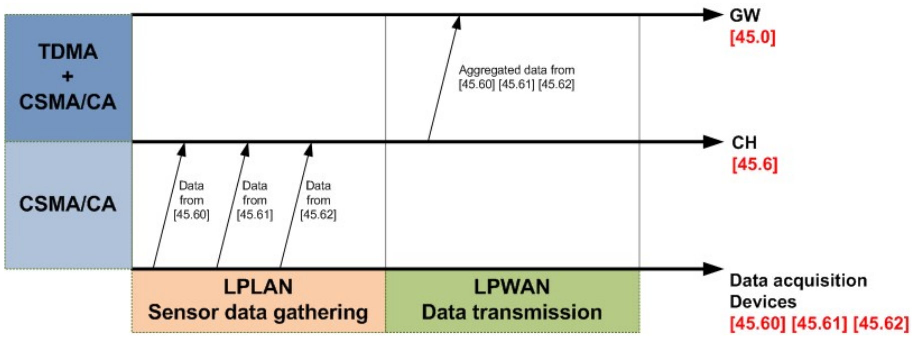

TDMA + CSMA/CA
CSMA/CA
Aggregated data from [45.60] [45.61] [45.62]
Data from [45.60]
Data from [45.61]
Data from [45.62]
LPLAN Sensor data gathering
LPWAN Data transmission
GW [45.0]
CH [45.6]
Data acquisition Devices [45.60] [45.61] [45.62]

Figure 55: Diagram of the continuous data delivery model operation

In the LPLAN sensor data gathering stage, all STAs send a message to their corresponding CH by means of the CSMA/CA contention-based protocol. After this stage is completed, the

LPWAN data transmission stage starts. In this case, each uplink data transmission phase is virtually split into as many TDMA slots as network rings, so that CHs are only active during their own slot (for transmitting data) and the previous one belonging to their children (for receiving data).

The first slot is allocated to the highest ring and the rest are scheduled consecutively. Data received by CHs is aggregated to that generated by themselves (i.e., data previously received from STAs of their LPLAN), and finally sent to the corresponding parent at the minimum power level which ensures reliable communications. This process is repeated as many times as rings the network has.

The correct reception of data transmissions at the GW is acknowledged with a broadcast message, so that CHs are not only aware of their own end-to-end reliability, but also of those CHs in the same path to the GW. These acknowledgment beacons, together with the information obtained from their adjacent nodes, allow CHs to decide whether they should remain awake to perform retransmissions of lost network packets.

QoS Management

The growing use of wireless sensor networks (WSN) brings networks consisting of hundreds or thousands of devices equipped with heterogeneous sensors performing different tasks depending on the running application. In such networks, a single access point (AP) or gateway (GW) includes/signals different procedures and time periods to manage the channel access of all its associated stations, which probably require different quality of service (QoS) levels. The scarce available bandwidth in the sub-1GHz band impedes the use of fully dedicated channels to determined types of application, as they would increase interference among communication devices.

Therefore, the GW must not only manage a large number of accesses from many devices, but also ensure the required QoS of each application running over the network. Apart from dealing with the well-known issues inherited from most general wireless networks, the very particular characteristics of WSNs add some challenges when trying to achieve a certain level of QoS; namely, resource constraints, unbalanced mixture traffic, data redundancy, network dynamism, energy efficiency or scalability, among others.

To manage the heterogeneous QoS requirements of varied applications in the INTER-HARE project, we will follow the scheme presented in, with the three general data delivery models defined in Section 3.3.3.5. Once defined the different data delivery models, it is necessary to quantify its performance by setting some QoS metrics. Commonly specified QoS metrics in WSNs are collective: latency, packet loss, bandwidth, and information throughput. More concretely, the metrics provisioned by the MAC layer, which is the focus of our work, are throughput, average packet, delay, and transmission reliability.

Different levels of priority will be therefore allocated in the INTER-HARE project depending on the running application (and its inherent data delivery model) of a STA / group of STAs. Under this approach, the system will prioritize event-driven communications ahead of query-driven ones. Continuous data transmissions will be then pushed into the background, as they are the less critical ones.

Aggregation and segmentation

In common IT applications, data aggregation refers to the process of compiling information from databases with intent to prepare combined datasets for data processing. In WSNs, data aggregation techniques use different node parameters to select and store data attributes in an aggregated format for further evaluation and usage. In multi-hop WSNs, data aggregation is performed in a distributed way, so that all nodes are responsible for performing these techniques over the received data.

The staggered wakeup pattern fits here perfectly with the approach of data aggregation in WSN. Thus CHs attach their own data to that received from their children and all the information is jointly sent to the next hop (i.e., parent). If the total amount of data aggregated by a CH exceeds the maximum payload supported by the hardware, it is split into segments[12] sent consecutively.

A selective ACK mechanism has been developed, so that before the end of the allocated time slot, the receiver explicitly lists which segments in a stream coming from the same child are acknowledged. Upper layers are therefore responsible for making the sender retransmit only the missing segments in successive transmission windows.

Power regulation mechanism (PRM)

The selection of the minimum suitable transmission power level for CH outgoing packets is managed through a mechanism based on the received signal strength indicator (RSSI). For this purpose, a safety margin for reliable communications is defined by $RSSI_{min}$ and $RSSI_{max}$. If a node is transmitting data packets (or ACKs) to its parent (or child) at a power level making the received RSSI higher than $RSSI_{max}$, it will be asked to decrease it for the next transmission. Similarly, if the received RSSI is lower than $RSSI_{min}$, it will be asked to increase it.

PRM requests are included in an RSSI control field of data packet and ACK headers. Possible values of this field are: *increase*, *keep*, and *decrease*. Once computed the requests from parent and children, the CH determines whether and how to regulate its own power level depending on the following considerations:

- If one or more CHs ask for a higher value, increase the power level.
- If all CH ask for a reduction, decrease the power level.
- Otherwise, keep the current power level.

In addition, if a CH needs to retransmit a packet to its parent, it will also increase the power level in each new transmission window. Regarding the association process, whenever a CH listens to a discovery request, it will answer at maximum power. The CH selected as parent will keep the maximum power level at the beginning and regulate it following the previously described procedure. Instead, those CHs not selected as parents will set their power back to the level they had before answering to the discovery request.

3.3.3.6 Network layer

Network communications follow a centralized scheme, where the GW adopts the main role and assumes the responsibility of managing network associations, delivering network addresses, and periodically notifying the start of new routing processes.

CHs adopt a subordinate role waiting for orders coming from the GW. In the routing process, they organize themselves in paths autonomously, but all subsequent data transmissions are addressed to the GW, directly or through other CHs. Conversely, the GW can make use of its greater transmission power to periodically send broadcast messages to all network CHs, or send unicast messages to selected CHs.

[12] The amount of data aggregated by an STA (from itself and from its children) is called packet. If this packet is split into different parts, each one of these parts is called segment. In case both terms can be indistinctly used, the current article will use packet.

In turn, STAs also adopt a subordinate role with regard to their corresponding CH, so that any network operation must be executed by using first these intermediate nodes.

Addressing system

The addressing system is managed by the GW, which allocates a unique network address to each node during the association process. Nodes will maintain the same network address as long as they do not leave the network. A dynamic record matching the MAC and the network address of all CHs and STAs is stored in the GW. The size of the network address is configurable and its value determines the addressing range.

In all the preliminary tests and in the INTER-HARE pilot, the addressing system will follow the Rime format, consisting of two 8-bit numbers. Similarly to IP addressing, the use of netmasks leads to flexible subnetting configurations with up to $(2^{16} - 2 = 65534)$ STAs. In our particular case, the first 8-bit number identifies the network prefix shared by all devices, and the second one the host part, whose value for GWs is 0, for cluster-heads is a number from 1 to 9, and for STAs is a number from 10 to 99.

The way for distinguishing one LPWAN from another is having a look at the first byte of any address. In addition, and for simplicity, the address of a data acquisition device is determined by the address of its corresponding cluster-head.

For example, if the address of a cluster-head is 45.5, addresses of the data acquisition devices from its LPLAN can only take values from 45.50 to 45.59. The range of the whole addressing system can be then determined as shown in Table 39. An example of address allocation for a typical INTER-HARE is shown in Figure 56, where GW represents a gateway, CH a cluster-head and N a data acquisition device.

Element	Frequency band	Quantity per LPWAN	Total amount	Example
Gateways	868 MHz	1	256	45.0
Cluster-heads	868 MHz & 2.4 GHz	9	9 · 256 = 2.304	45.5
Data acquisition devices	2.4 GHz	90	90 · 256 = 23.040	45.58

Table 39: Main features of the INTER-HARE addressing system

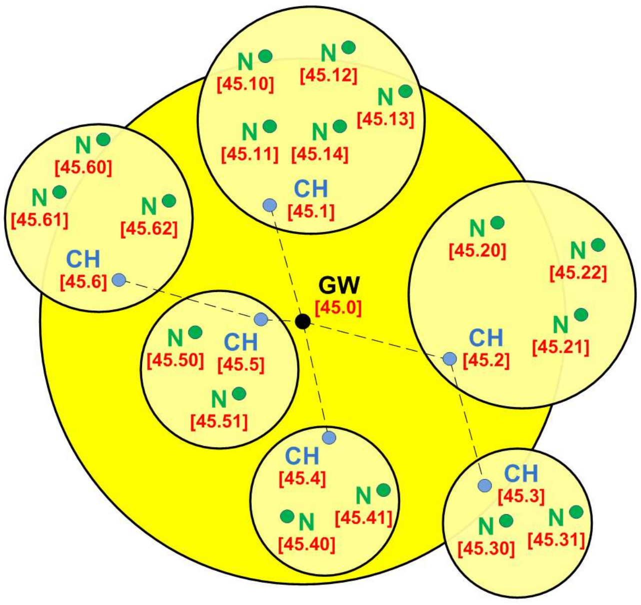


Figure 56: Example of address allocation for a typical INTER-HARE deployment

3.3.3.7 Network association

To cope with multiple association requests in a short period of time, the system is able to admit new CHs and STAs through two different mechanisms: an active, global, scheduled one, called *network association mechanism*; and a passive, singular one, called *STA association mechanism*.

Network association mechanism

The *network association mechanism* (also known as *re-association mechanism*) allows a large amount of CHs to associate to the network in a short period of time. Once the GW is activated, or after a pre-determined number of primary beacons (N_{pr}), the GW broadcasts a network association primary beacon.

Depending on the RSSI value received in the network association primary beacon, CHs determine their association turn (generally, the greater the RSSI received, the earlier association turn is selected). Then, a random number is selected by the CH, determining its association slot within the association turn. The number of association turns (a_t), association slots per turn (a_s), and the length of a slot (T_a) are parameters set by the GW and included in every network association primary beacon.

CHs then follow with a discovery message sent via broadcast, which is responded by the GW and all the already associated CHs, provided they are within the coverage range. The process of selecting the best path to reach the GW is detailed in Section 3.3.3.9: Routing.

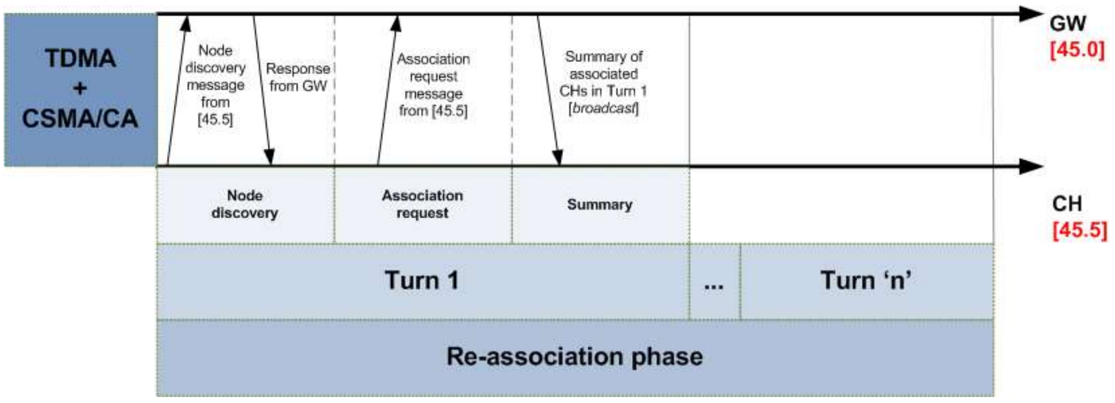


Figure 57: Diagram of the network association mechanism operation

Once the routing mechanism is completed, the GW notifies the joining of new CHs by means of a summary broadcast message sent immediately after every association turn.

STA association mechanism

The *STA association mechanism* provides a solution to those specific CHs that (i) have not found a path to the GW during the network association mechanism, (ii) have been powered on between two consecutive network association primary beacons, or (iii) have simply suffered routing problems in their path to the GW. It is also the way for STAs to enter in the network.

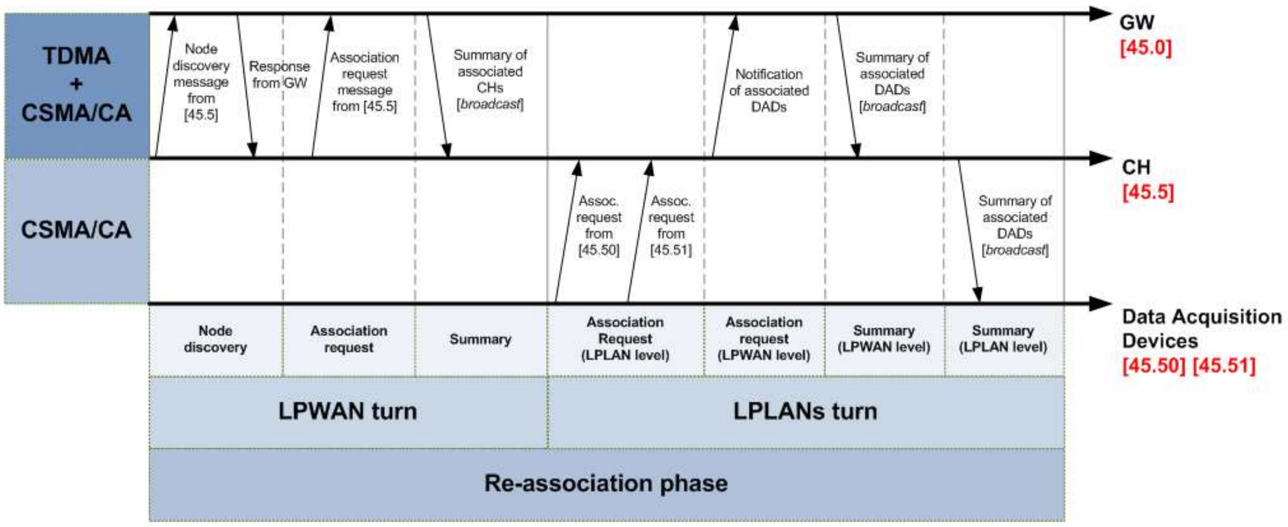

TDMA + CSMA/CA
CSMA/CA
Node discovery message from [45.5]
Response from GW
Association request message from [45.5]
Summary of associated CHs [broadcast]
Notification of associated DADs
Summary of associated DADs [broadcast]
Assoc. request from [45.50]
Assoc. request from [45.51]
Summary of associated DADs [broadcast]
GW [45.0]
CH [45.5]
Data Acquisition Devices [45.50] [45.51]
Node discovery
Association request
Summary
Association Request (LPLAN level)
Association request (LPWAN level)
Summary (LPWAN level)
Summary (LPLAN level)
LPWAN turn
LPLANs turn
Re-association phase

Figure 58: Diagram of the STA association operation

This mechanism follows the same pattern as the network association one, with the single exception that there is only one association turn located immediately after each data primary beacon to be used by non-associated CHs and, then, another one for non-associated STAs.

3.3.3.8 Network disassociation

Inactive or erratic CHs are removed from the network and the GW's routing table to create, if necessary, new routing paths that ensure correct packet reception from remaining network CHs. Disassociations can be controlled by the GW through the disassociation mechanism or by the CHs themselves through the self-disassociation mechanism.

Disassociation mechanism

The GW removes a CH from the network if not receiving any data packet during a pre-determined number of consecutive primary beacons (N_{pd}). A roster with the latest disassociated CHs is included in every primary beacon. This information is not only useful for malfunctioning CHs, which can make immediate use of the CH association mechanism, but also for their parents, as they can check the current state of their children. Hence, if all its children became disassociated, a parent would go to sleep during the RX time period allocated to its ring.

Self-disassociation mechanism

The goal of this mechanism is to avoid repetitive association requests and other energy consuming procedures that could make CHs run out of battery when no connection with the GW is possible. All CHs have a timer that is activated after being switched on or when receiving a primary beacon. From that moment on, if a CH does not receive any other beacon during a predetermined period (T_d), it turns itself off. Thus the CH is considered dead and it will need to be reactivated by manual procedures.

3.3.3.9 Routing

Routing will be only used in the LPWAN, as all LPLAN connections will follow a single-hop approach between the data acquisition devices and the corresponding cluster-head. In the specific case of the LPWAN, the INTER-HARE platform uses its own distance vector routing protocol inspired by RPL to build a destination oriented directed acyclic graph (DODAG).

This protocol is only executed as part of the *Network association mechanism* and the *STA association mechanism*, so that there is no continuous routing packet exchange among neighbors. Thus, according to the responses to the discovery message coming from other nodes, each STA determines the best candidate to become its parent in its path to the GW; i.e., the node with the minimum S value from:

$$S = a_1 \cdot \left(P_{TX_{max}} - RSSI_{TX}\right) + a_2 \cdot \left(P_{TX_{max}} - RSSI_{RX}\right) + a_3 \cdot r + a_4 \cdot c$$

Where $P_{TX_{max}}$ is the maximum transmission power of the transceiver (in dBm), $RSSI_{TX}$ is the RSSI received at the candidate (in dBm), $RSSI_{RX}$ is the RSSI received at the STA itself (in dBm), r is the ring to which the candidate belongs, and c is the current number of candidate's children. The a weights are attached to every *primary beacon*, and can be tuned by the GW according to environmental requirements.

Once computed the best parent, the STA sends it a specific request. This request will be forwarded by the parent through its own path until reaching the GW, which will send a packet via broadcast confirming the association and providing the STA with its new address. This way, both the newly associated STA and its parent are informed of the establishment of the new path.

When the association process is finished, the STA exactly knows the next hop its messages must follow to reach the GW. As long as the STA is associated to the network, it uses the same routing path, which is only recomputed after an internal or external (i.e., from its parent) failure. Indeed, no new routing process is initiated unless it is part of a new *network association mechanism*.

3.3.3.10 Transport layer

Reliable end-to-end communications from the cluster-heads to the GW, where retransmissions are only executed when needed and by the minimum number of involved cluster-heads (and relays), are achieved in the INTER-HARE platform by using the mechanisms described in the current section.

It is worth noting here that data acquisition devices from 2.4 GHz LPLANs will remain out from this transport layer, so that their packets will be individually acknowledged by their corresponding cluster-head in the previously established single-hop 2.4 GHz link. If any data acquisition device did not send its data packet to the cluster-head due to an error or a connection failure, the cluster-head would inform the INTER-IoT gateway of this circumstance by attaching a message similar to *'no data'* in the field corresponding to the erratic station.

3.3.3.11 End-to-end ACK

According to the staggered wakeup pattern described in a future section, STAs from ring 1 are the last ones to access to the channel and transmit their information. Once compared the data sources with the expected uplink traffic the GW emits a broadcast message called *end-to-end ACK* (e2e ACK) with a list of acknowledged STAs. Apart from being simple, quick and simultaneously listened by all network elements, e2e ACKs allow STAs to evaluate the state of their path to the GW and act consequently.

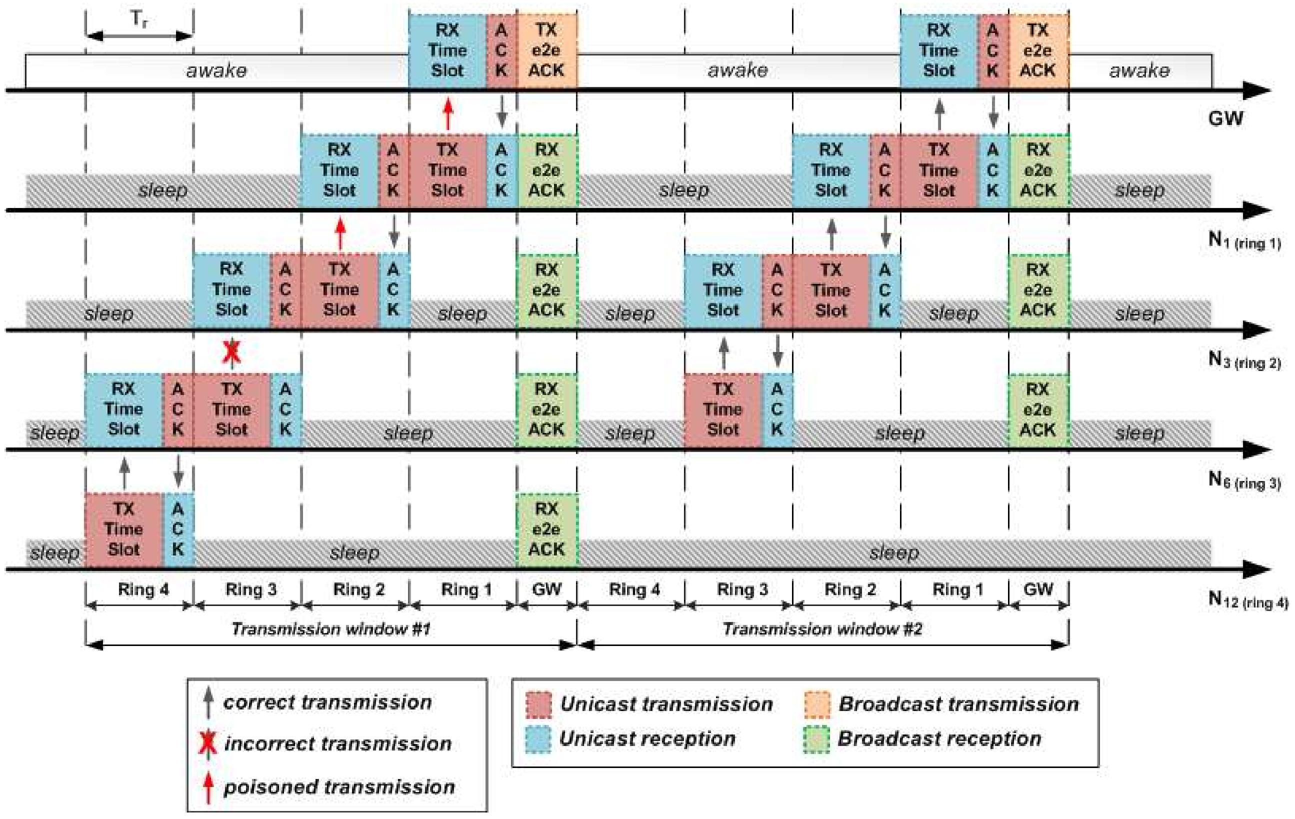


Figure 59: e2e ACK operation

Figure 59 shows an example of the e2e ACK operation at the end of every LPWAN transmission window. Note the communication problems in the first transmission window between nodes N_6 and N_3 and how they are solved in the second transmission window with the collaboration of just the affected nodes.

The time difference between the transmission of a data packet and the proper reception of the e2e ACK determines the end-to-end communication delay of an STA (D_{e2e}). This value can be theoretically computed by using the following equation:

$$D_{e2e} = T_r + (r-1)\cdot T_r + (i-1)\cdot R\cdot T_r = (r+(i-1)\cdot R)\cdot T_r,$$

where T_r is the duration of a ring slot, r is the ring to which the STA belongs, i is the index of the transmission window in which the STA receives the e2e ACK (with $i \in [1, w]$), and R is the total number of network rings.

3.3.3.12 Poisoning mechanism

The poisoning mechanism identifies which specific nodes experience communication problems in their path to the GW, so that they can perform subsequent retransmissions. Nodes having problems with their children transmit packets with the poison flag activated. An STA is considered poisoned if, before transmitting an outgoing data packet, one of the following conditions is satisfied:

1. The STA is part of a poisoned path; i.e., it has received one or more packets with the poison flag activated during the current transmission window.
2. The STA has not received any data packet from one or more of its children.
3. The STA has not received all the expected segments from one or more of its children.

In Figure 60, node N3 activates its poison flag after not receiving data from its child N6. In its way to the GW, a data packet from N3 poisons its next hop: N1. Therefore, nodes N6, N3, and N1 form a poisoned path, as shown in Figure 61. Together with the GW, these nodes (colored in red) stay awake during the second transmission window. The rest of nodes (colored in green) go to sleep as they are not involved in the new transmission process.

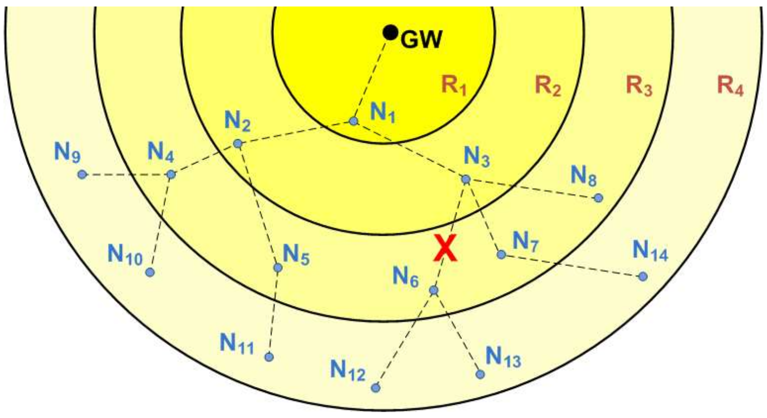


Figure 60: Network topology of the multi-hop LPWAN from Figure 59

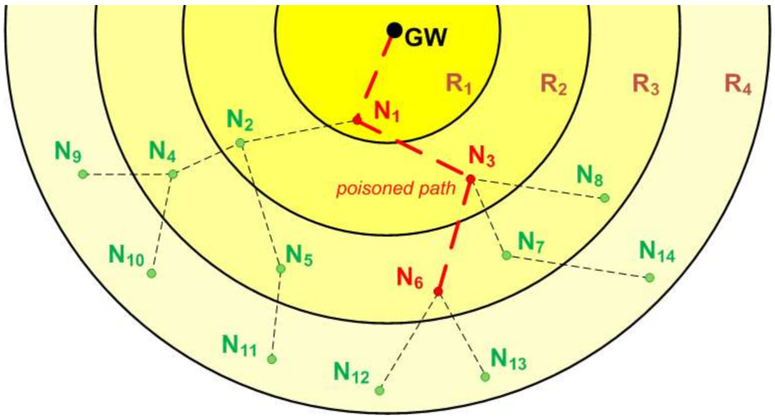


Figure 61: State of the network from Figure 60 after the corresponding e2e ACK

3.3.3.13 Transmission windows

A number of transmission windows (w) with their corresponding e2e ACK are included in each uplink data transmission phase to ensure correct packet reception. Within these windows, not all STAs remain awake, but only the ones directly involved in the transmission process. Before the start of a new transmission window, STAs evaluate whether they shall stay awake or go to sleep.

This decision takes into account if the STA has been previously poisoned by one of its children as well as several other conditions according to the decision flowchart from Figure 62. Whenever an STA decides to go to sleep, it will remain in this state until the next *primary beacon*.

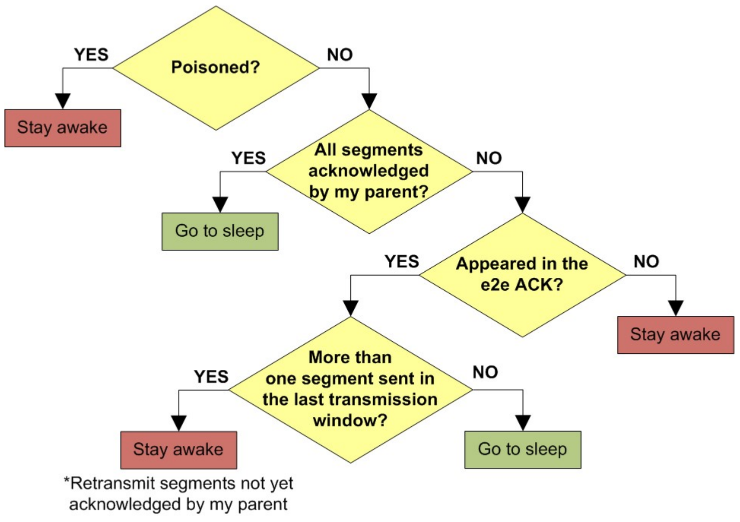

Poisoned?
YES
NO
Stay awake
All segments acknowledged by my parent?
YES
NO
Go to sleep
Appeared in the e2e ACK?
YES
NO
Stay awake
More than one segment sent in the last transmission window?
YES
NO
Stay awake
Go to sleep
*Retransmit segments not yet acknowledged by my parent

Figure 62: STA's decision flowchart to stay awake or go to sleep before the start of a new transmission window

3.3.3.14 Distributed caching

Due to the structure of multi-hop networks, lost packets cause expensive retransmissions along every hop of the path between the sender and the receiver. To alleviate this problem, a distributed caching system is used in the INTER-HARE *transport network*, so that parents acknowledge the correct reception of packets from children and cache their data until it is properly received in the GW.

As it can be seen in Figure 61, nodes N_{12} and N_{13} can go to sleep after the first transmission window, because their data packets have been acknowledged by node N_6, which will cache them in memory together with its own data to be sent in the next transmission window.

3.3.3.15 Frame size

The INTER-HARE application frame size is determined by the 127 bytes of the maximum transmission unit (MTU) in the IEEE 802.15.4 standard, which is used as the PHY and MAC layer of the current development. In addition, the free space available for the application is also reduced as consequence of the headers from both the RDC (in our case, X-MAC) and the network layers (in our case, Rime).

TYPE OF COMMUNICATION	IEEE 802.15.4 MTU	Size of RDC layer header	Size of network layer header	Available space for the application layer
Broadcast	127 bytes	2 bytes	16 bytes	109 bytes
Unicast	127 bytes	2 bytes	32 bytes	93 bytes

Table 40: Available space for the application layer

As it will be described in Subsection 3.3.3.18 with the description of the different application packets, data and statistics packets consist of 10 and 20 bytes of information, respectively. Having into account the available space for the application layer, in case an intermediate STA aggregates data from other stations, it could include up to 8 different data payloads and up to 4 different statistics payloads.

Packet headers

All packets in the INTER-HARE platform include a header of 2 bytes (16 bits) for allowing the receiver determining the type of packet. Depending on the packet, other fields may be encoded as shown in the list of headers summarized in the table below.

Id	Packet	Bit index						
		Packet ID 1-4	**5**	**6**	**7**	**8-10**	**11-13**	**14-16**
0	**Void beacon**	0000	*(Void)*					
1	**Data packet**	0001	RSSI control		Agg.	Total segments	Current segment	*(Void)*
2	**Data poissoned**	0010	RSSI control		Agg.	Total segments	Current segment	*(Void)*
3	**ACK**	0011	RSSI control		*(Void)*			
4	**Data beacon**	0100	Kill flag	*(Void)*				
5	**e2e ACK**	0101	Routing aux			*(Void)*		
6	**Node discovery**	0110	Mode	*(Void)*				
7	**Association**	0111	Mode		Version	*(Void)*		
8	**Re-Association beacon**	1000	Mode		*(Void)*			
9	**Statistics packet**	1001	RSSI control		Agg.	Total segments	Current segment	*(Void)*
10	**Statistics poissoned**	1010	RSSI control		Agg.	Total segments	Current segment	*(Void)*
11	**Statistics beacon**	1011	Kill flag	*(Void)*				
12	**Query-driven**	1100	Mode	*(Void)*				
13	**Event-driven**	1101	Mode	*(Void)*				

14	**Connection test**	1110	Mode	*(Void)*

Table 41: Packet headers encoding in INTER-HARE

The fields encoded in the headers are detailed below:

- **Packet id:** packet identifier.
- **RSSI control:** flag for acting according to the PRM.
- **Aggregation:** flag indicating if more than one segment will be sent in a transmission.
- **Total segment:** total number of segments in a transmission.
- **Current segment:** number of the segment being sent.
- **Kill flag:** flag for identifying if any of the STAs in the network has been killed (or de-associated).
- **Mode (for Discovery, Association, Re-association, Query-driven, Event-driven and Connection test):**
    - **Req:** node discovery / association request.
    - **Resp:** node discovery / association response (not for Event-driven).
    - **Ret:** (only for Assoc. and Re-Assoc) node request retransmitted to a lower ring.
- **Version:** software version (only included in the association request frame).

3.3.3.16 Management frames

There are three types of management packets: discovery and association. While the first one is related to the node discovery phase, the second is used for requesting and confirming association. The connection test is the first message transmitted by an STA once it has been switched on, and its main purpose is to check the availability of GWs in its own range coverage.

Type	**Transmission Byte index**													
	1-2	**3**	**4**	**5**	**6**	**7-8**	**9-10**	**11-12**	**13-14**	**15**	**16**	**17-18**	**19**	**20**
A	*(Void)*													
B	rssi_rx	ring	ch	*(Void)*										
C	mac address						*node_id*	*(Void)*						
D	timestamp			length	new	mac_children address				rime_parent		next_hop	*ring*	*ch*
E	mac_child address						*node_id_ch*	rime	*(Void)*					
F	*(Void)*													
G	*Data packet*							*(Void)*						
H	*Data packet*							*(Void)*						
I	*(Void)*													
J	t_response			*(Void)*										

Table 42: Field content of management frames

Type	Description	Sender	Transmission
A	**Discovery request**	STA	Broadcast
B	**Discovery response**	GW/ STA	Unicast
C	**Association request**	STA	Unicast
D	**Association response**	GW	Broadcast
E	**Association retransmission**	STA	Unicast
44F	**Query-driven request**	GW	Unicast
G	**Query-driven response**	STA	Unicast
H	**Event-driven request**	STA	Unicast
I	**Connection test request**	STA	Broadcast
J	**Connection test response**	GW	Unicast

Table 43: Type descriptions of management frames

- **Discovery:**
  - **Request:** the node discovery request is just composed with the 2-byte header. No additional fields are included in the packet.
  - **Response:** the node discovery response includes the RSSI received at the potential parent (*rssi_rx*), and its ring (*ring*) and number of children (*ch*).
- **Association:**
  - **Request:** the association request packet includes the MAC address (*mac*) and node ID (*node_id*) of the STA willing to be associated.
  - **Retransmission:** association request retransmissions include the information contained in the request forwarded and an additional field with the RIME address of the intermediate hop (*rime*).
  - **Response:** the association response is sent via broadcast by the GW and summarizes the information regarding the new STAs associated to the network. That is, the fields included are the timestamp for synchronization (*timestamp*), the packet length (*length*) for identifying the number of association response packets to be sent, the number of new associated STAs (*new*), and other fields containing the information of the new routing connections.

    Specifically, such fields are the MAC address of the new STAs associated (*mac_ch*), the RIME address of the assigned parent (*rime_parent*), the allocated next hop (*next_hop*), the ring of the associated STA (*ring*), and the number of children (*ch*).
- **Query-driven:**
  - **Request:** the query-driven request is just composed with the 2 byte header. No additional fields are included in the packet.
  - **Response:** the response consists of a traditional data packet.
- **Event-driven:**
  - **Request:** the request consists of a traditional data packet.
- **Connection test:**
  - **Request:** the connection test request is just composed with the 2 byte header. No additional fields are included in the packet.

- **Response:** the connection test response includes the time until the next beacon is transmitted (*t_response*).

3.3.3.17 Beacons

There are four different beacon types that can be emitted by the GW in a broadcast transmission: re-association beacon, void beacon, data beacon, and statistics beacon. As seen in orange color in its Subsection each one of them has its own packet header. As for the content of the beacon itself, they can be grouped into re-association beacons and regular beacons (this group contains the aforementioned void beacon, data beacon, and statistics beacon).

BEACONS	Byte index										
	1-4	5-8	9-12	13	14	15	16	17	18	19-24	25-30
Re-association beacon	timestamp	t_euthanasia	*t_reassoc_wake*	*association_info*							
Regular beacon	timestamp	t_duty	*t_action*	t_unit	t_tx	t_assoc	R	W	*(void)*	*association_info*	

Table 44: Field content of beacons

	Byte index										
	1-2	3	4	5	6	7	8	9	10	11	12
association_info	max_rssi	num_ranges	*gap_r1*	t_range	w_1	w_2	w_3	w_4	*max_ch*	*num_slots*	*t_slot*

Table 45: Content of the 'association_info' field

- **Re-association beacon:**
  - **timestamp:** Field for synchronization purposes
  - **t_euthanasia:** Time before STAs get disconnected by not listening any beacon
  - **t_reassoc_wake:** Time comprised between the beginning of a re-association beacon and its immediately following beacon
  - **association_info:** Block of fields containing information corresponding to the association phase of the system. It consists of the following parameters:
    - **max_rssi:** Maximum RSSI allowed in the re-association process
    - **num ranges:** Number of complete association ranges
    - **gap_r1:** RSSI gap of the first association range
    - **t_range:** Time length of an association range
    - **w_i:** Weights to compute the best response to the association proces
    - **w_1:** Weight of the received RSSI
    - **w_2:** Weight of the RSSI received at the candidate node
    - **w_3:** Weight of the ring of the candidate
    - **w_4:** Weight of the number of children of the candidate
    - **max ch:** Maximum number of children per STA set in the network
    - **num slots:** Number of node discovery slots
    - **t_slot:** Time of each node discovery slot
- **Regular beacon (void, data, or statistic beacon):**
  - **timestamp:** Field for synchronization purposes

- **t_duty:** Time to next duty cycle
- **t_action:** Time to the next primary beacon
- **t_unit:** Time unit in which the mentioned variables are represented (usually given in seconds)
- **t_tx:** Time an STA can be in TX state
- **t_assoc:** Time slot for association
- **R:** Number of rings in the network
- **W:** Number of transmission windows
- **-:** Reserved space (currently not used)
- **association_info:** Block of fields containing information corresponding to the association phase of the system. It consists of the following parameters:
  - **max_rssi:** Maximum RSSI allowed in the re-association process
  - **num ranges:** Number of complete association ranges
  - **gap_r1:** RSSI gap of the first association range
  - **t_range:** Time length of an association range
  - **w_i:** Weights to compute the best response to the association proces
  - **w_1:** Weight of the received RSSI
  - **w_2:** Weight of the RSSI received at the candidate node
  - **w_3:** Weight of the ring of the candidate
  - **w_4:** Weight of the number of children of the candidate
  - **max ch:** Maximum number of children per STA set in the network
  - **num slots:** Number of node discovery slots
  - **t_slot:** Time of each node discovery slot

3.3.3.18 Application packets

The INTER-HARE application relies on different packets exchanged between the different elements of the network. Four different types of packets have been defined: data, statistics, link ACK and end-to-end ACK packets.

Data packets

Data packets are intended to compile the information collected by sensors. The designed structure for this kind of packet is shown in Table 46 and contains the following fields:

APPLICATION PACKETS	Byte index									
	1	2	3	4	5	6	7	8	9	10
Data packet	Rime		Seq	Temp		Hum		Light	Bat	*(Void)*

Table 46: Field content of a data packet

- **Rime**: Network address of the STA (although the network address of the sender is always included in the IEEE 802.15.4 header, it is indispensable to attach this information in every data packet to avoid misunderstandings about the data source when performing data aggregation).
- **Seq:** Data packet sequence number.
- **Temp:** Temperature measured by the sensor (in °C) with up to three decimals accuracy. For instance, 25.453 °C. Note that the final accuracy of the measure will depend on the selected temperature sensor.

- **Hum:** Humidity measured by the sensor (in %) with up to three decimals accuracy. For instance, 58.947 %. Note that the final accuracy of the measure will depend on the selected humidity sensor.
- **Light:** Luminance measured by the sensor in % without decimals (for instance, 72 %) (optional).
- **Bat:** Remaining battery capacity of the Zolertia Re-Mote in % without decimals (for instance, 94%).
- ***(Void):*** Field reserved, if necessary, for future purposes.

Statistics packets

The statistics packets have been designed to provide the system administrator with information about the network performance as well as the operational state of STAs. They are periodically sent by STAs following the same paths established for the transmission of data packets. Their structure can be reviewed at Table 47 and contain the following fields:

APPLICATION PACKETS	Byte index									
	1-2	3	4	5-6	7-8	9-10	11-12	13-14	15-18	19-20
Statistics packet	Rime	Seq	*(Void)*	pkts_sent	pkts_acked	acks_sent	rtt_link	rtt_e2e	*time_states*	*power_levels*

Table 47: Field content of a statistics packet

- **Rime:** Network address of the STA (although the network address of the sender is always included in the IEEE 802.15.4 header, it is indispensable to attach this information in every data packet to avoid misunderstandings about the data source when performing data aggregation).
- **Seq:** Statistics packet sequence number.
- ***(Void):*** Field reserved, if necessary, for future purposes.
- **pkts_sent:** Number of data packets sent by the STA.
- **pkts_acked:** Number of data packets properly acknowledged by the STA's parent.
- **acks_sent:** Number of link ACK's packets sent to the STA's children.
- **rtt_link:** Average Round Trip Time (RTT) at link level expressed in ms.
- **rtt_e2e:** Average Round Trip Time (RTT) at end-to-end level expressed in s.
- **time_states:** Percentage of time spent by the STA in each of the possible transceiver states.
- **power_levels:** Maximum and minimum power level used by the STA during the period from the last statistics packets sent.

With all this information gathered from STAs as well as other internal processes, the GW is able to compute all the following performance metrics:

#	Variable	Metric	Type [13]	Update period
1	t_O	Observation time	Accumulated	Cycle
2	$n_{A,c}$	Number of associated STAs cycle c	Current	Cycle

[13] There are two types of metrics: **accumulated** metrics are those which are obtained by gathering the network accumulated information from the beginning of the execution, and **current** metrics are those which are obtained processing the partial information gathered during a cycle.

3	D	Number of duty cycles	Accumulated	Cycle
4	PDR	Packet delivery ratio	Accumulated	Cycle
5	CSR	Cycle stability ratio	Accumulated	Cycle
6	N_A	Total number of associations	Accumulated	Cycle
7	N_K	Number of kills	Accumulated	Cycle
8	$\bar{N}_R$	Mean number of rings	Accumulated	Cycle
9	$\bar{d}_A$	Mean association delay	Accumulated	Cycle
10	$N_{A,c}$	Number of associations in cycle c	Current	Cycle
11	N_c	Number of rings in cycle c	Current	Cycle
12	$\bar{N}_{ACK,c}$	Mean number of ACKs sent per STA in cycle c	Current	Statistics
13	$\bar{t}_{k,d}$	Mean share of time in state k in cycle c	Current	Statistics
14	$\bar{\rho}_d$	Mean number of transmissions per packet	Accumulated	Statistics
15	RTT_{link}	RTT at link level	Current	Statistics
16	RTT_{e2e}	RTT at end-to-end level	Current	Statistics
17	$\bar{\mathrm{t}}_{\mathrm{k,d}}$	Percentage of time in each state	Accumulated	Statistics
18	$\mathrm{P}_{\max}$ /$\mathrm{P}_{\min}$	Maximum and minimum employed power level	Current	Statistics

Table 48: Network performance metrics generated by the INTER-HARE gateway

- **Observation time and number of duty cycles**
  Time and number of duty cycles that had taken place during the experiment. We consider that time starts when the GW button is pressed (first beacon sent). It is important to note that before pressing the GW button, all the STAs in the network should be already turned on (in RX/listening state).
- **Associations and disassociations**
  - $\mathrm{n}_{\mathrm{A,c}}$: Number of associated STAs in the last cycle c. That is, the number of stations included in the routing table in the last cycle.
  - N_{A}: Total number of associations performed during the observation time.
  - $\mathrm{n}_{\mathrm{K,c}}$: Number of disassociated STAs in the last cycle c. That is, the number of stations removed from the routing table in the last cycle.
  - N_{K}: Total number of disassociations performed during the observation time.
- **PDR and CSR**
  - The Packet Delivery Ratio (PDR) metric measures the network efficiency in terms of data reception. It is the ratio of the number of expected payloads received and expected.

$$PDR = \frac{Num.of\ expected\ payloads\ received}{Num.of\ expected\ payloads}$$

  - The Cycle Stability Ratio (CSR) metric measures the network stability in terms of disassociated STAs. It is the ratio of the number of duty cycles without disassociations and the total number of duty cycles.

$$CSR = \frac{Num.\,of\ duty\ cycles\ with\ no\ disassociated\ STAs}{Num.\,of\ duty\ cycles}$$

- **Rings**
  - $\mathrm{n_{R,c}}$: Number of rings in the last duty cycle c. The ring is determined by the maximum number of hops of a branch in the network.
  - $\mathrm{\overline{N}_{R,c}}$: Mean number of rings per cycle (dynamic average) in duty cycle c. The formula uses the last cycle ring mean, which is modified in every cycle with the new mean.

$$\overline{N}_{R,c} = \frac{\overline{N}_{(c-1)} * (c-1) + n_{R,c}}{c}$$

- **Association delay**
  Mean association time that takes an STA to get associated (i.e. included in the GW routing table) from the last re-association beacon spread.

$$\mathrm{\bar{d}_A} = \frac{\sum_{\mathrm{s \in S}} (\mathrm{d_{A,s}} - \mathrm{t_{reassoc}})}{|\mathrm{S}|}$$

  Where $\mathrm{t_{assoc,s}}$ is the timestamp when the STA s is included in the routing table, $\mathrm{t_{reassoc}}$ is the timestamp when the re-association beacon is sent and S is the set of nodes associated.

- **Mean number of transmissions per data packet**
  Mean number of transmissions per packet14 for the stations associated in cycle d.

$$\bar{\rho}_\mathrm{d} = \frac{\sum_{\mathrm{s \in S}} \left(\frac{\mathrm{N_{acked,s}}}{\mathrm{N_{sent,s}}}\right)}{|\mathrm{S}|}$$

  Where $\mathrm{N_{acked,s}}$ is the number of packets of station s that have been acknowledged (both as a consequence of a parent ACK or en e2e ACK), and $\mathrm{N_{sent,s}}$ is the number of packets that stations s has sent.

- **Number of ACKs sent per STA**
  Mean number of ACKs sent per STA in the last cycle d. Each STA is expected to send 1 ACK per child; however, some retransmissions could be needed due to packet losses.

$$\mathrm{\overline{N}_{ACK,c}} = \frac{\sum_{\mathrm{s \in S}} \mathrm{N_{ACK,s}}}{|\mathrm{S}|}$$

  Where $\mathrm{N_{ACK,s}}$ is the number of ACKs sent by STA s.

- **Time in each state**
  Mean share of time an STA is in each of the possible 4 states during the last cycle d: CPU, LPM, TX, and RX.

$$\mathrm{\bar{t}_{k,d}} = \frac{\sum_{\mathrm{s \in S}} \mathrm{t_{k,s}}}{|\mathrm{S}|}$$

  Where $\mathrm{t_{k,s}}$ is the share of time a STA s is in state k.

[14] In the INTER-HARE packet hierarchy, we would be counting the number of transmissions needed for transmitting successfully **a transmission** (set of segments).

- **Round-trip-times**
  There are two kinds of round-trip-times (RTTs) considered:
    - Link, $\mathrm{RTT}_{\mathrm{link}}$: from child's transmission to parent's ACK reception.
    - End-to-end, $\mathrm{RTT}_{\mathrm{e2e}}$: from STA's transmission to gateway's end-to-end ACK reception.

The transmission time is measured right before the *unicast_send* command for DATA packets (statistics packets' RTTs are not computed). There are some instructions expected to be done before actually transmitting the packet, but measuring in the exact moment when bytes are starting to be transmitted is not possible. The reception time is measured right after identifying the ACK packet by its sequence ID as an acknowledgment of the transmitted packet.

Both parameters are obtained averaging the RTT values of the data cycles between two consecutive statistics cycles. Hence, they only consider the transmission of data packets (RTTs with respect to the transmission of statistics packets are not considered).

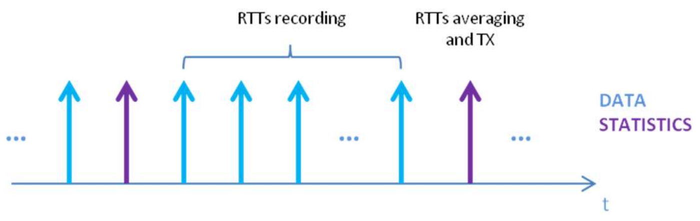


Figure 63: Mechanism of RTT averaging

In addition, if a transmitted packet is not acknowledged in the first transmission, the e2e RTT will be based on the first transmission time, no matter the number of retransmissions. Instead, the link RTT will consider the timestamp corresponding to the last transmission start.

Segmentation and link RTTs: If there is more than one packet to be transmitted, the RTT will be computed taking into account the first segment sent.

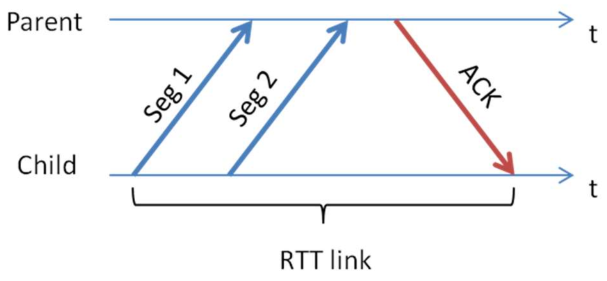


Figure 64: Computation of RTT link between a parent and a child when sending more than one data segment

In addition, two different kinds of histograms are periodically generated:

- **Number of cycles an STA is associated**
  This kind of histogram shows the number of cycles that an STA has been associated during a given observation time.

- **Number of STAs associated**
  This kind of histogram shows the number of cycles that an STA has been associated during a given observation time.

Link ACK packets

The link ACK packet informs about the proper reception of a data packet by a direct parent. Table 49 shows its internal structure, which includes the following fields:

APPLICATION PACKETS	Byte index	
	1	2
Link ACK packet	Segment	Seq

Table 49: Field content of a link ACK packet

- **Segment:** The sequence number of the data packet acknowledged.
- **Seq:** Link ACK sequence number.

End-to-end ACK packets (e2e ACK)

Similarly, to the link ACK packets, the purpose of the e2e ACK packets is to notify the proper reception of a data packet by the gateway. They only include a field:

APPLICATION PACKETS	Byte index
	1-4
e2e ACK packet	*ack_encoded*

Table 50: Field content of an e2e ACK packet

- **ack_encoded:** This field consists of a binary acknowledgement for each STA in the network. Such 4 bytes' value determines which STAs must retransmit their packets if they were not received by the GW.

Data Acquisition

Once a DAD receives a data primary beacon from its corresponding CH asking for environmental data or after an explicit request coming from the GW, the installed routines in the end device ask the corresponding sensor for this data. Every DAD of the INTER-HARE transport network contains a temperature and humidity sensor. The selected sensor for performing this task is the DHT22, whose main characteristics are detailed in ***Annex D: DHT22 temperature and humidity sensor***.

3.3.3.19 General considerations

The integration network is responsible for ensuring the communication between the physical gateway and the rest of the INTER-IoT system (or more specifically, with the *virtual* gateway). Communication within this network is bidirectional, as the INTER-IoT system allows both sending specific requests to deployed devices and receiving information from those nodes.

Specifically, communication between the physical and the virtual gateway can be conducted through a simple link using one of the following systems (WiFi, GPRS or Ethernet), so that no further modifications or new developments are considered at this communication level.

As for exchanging messages with the rest of the INTER-IoT system, the most two common protocols in the IoT will be considered: CoAP and MQTT.

- ***CoAP (Constrained Application Protocol)*** is a protocol intended for resource-constrained IoT devices that enables IoT devices to communicate with the Internet. CoAP is based on HTTP and the REST model where resources are retrieved from a server using URIs/URLs, the clients use the well-known methods of GET, PUT, POST, and DELETE to manipulate these resources. CoAP can be used via other mechanisms, such as SMS on mobile communication networks.

CoAP is designed to provide multicast support, low overhead, and simplicity. It is designed to work on microcontrollers with as low as 10 KB of RAM and 100 KB of storage space while also providing strong security.

- A different approach is provided by the MQTT protocol. ***MQTT (Message Queuing Telemetry Transport)*** is a lightweight protocol. MQTT is best suited for systems that rely on low bandwidth connections and require code with a small footprint. MQTT protocols uses the concept of publish-subscribe communications among nodes.

  The publish-subscribe schema requires the presence of an intermediate node called a message broker. Every source of data must publish the data element on the broker node indicating to which "topic" the data belongs. The nodes interested in receiving data on a specific topic must subscribe to that topic on the broker. The broker will then distribute the messages to interested clients based on the topic of a message.

Due to its operation based on well-known request and receive methods, CoAP will be the preferred protocol to be included in the INTER-IoT gateway of the INTER-HARE platform.

3.3.3.20 Architecture

The architecture of the integration network is closely related to that of the employed gateway. In the specific case of the INTER-HARE platform, and according to the options presented in deliverable ***D3.1. Methods for Interoperability and Integration***, the gateway element has been split into two parts: *"the physical part for the embedded device and the part that can be executed in a virtual container"*.

A clear definition of this architecture can be observed in Figure 65, where the two parts of the gateway (the physical and the virtual) are clearly defined. In the INTER-HARE platform, the elements performing each role are defined as follows:

- **Physical gateway:** A combination of a wireless frontend (responsible for the communication with the rest of the transport network) and a controller (responsible for the communication with the virtual gateway). Both elements are connected through a serial link.
- **Virtual gateway:** A virtual entity which can be executed in a remote location, based in the Docker[15] platform; i.e., a virtual container that provides an additional layer of abstraction and automation of operating-system-level virtualization on Windows and Linux.

[15] Docker main website - https://www.docker.com/

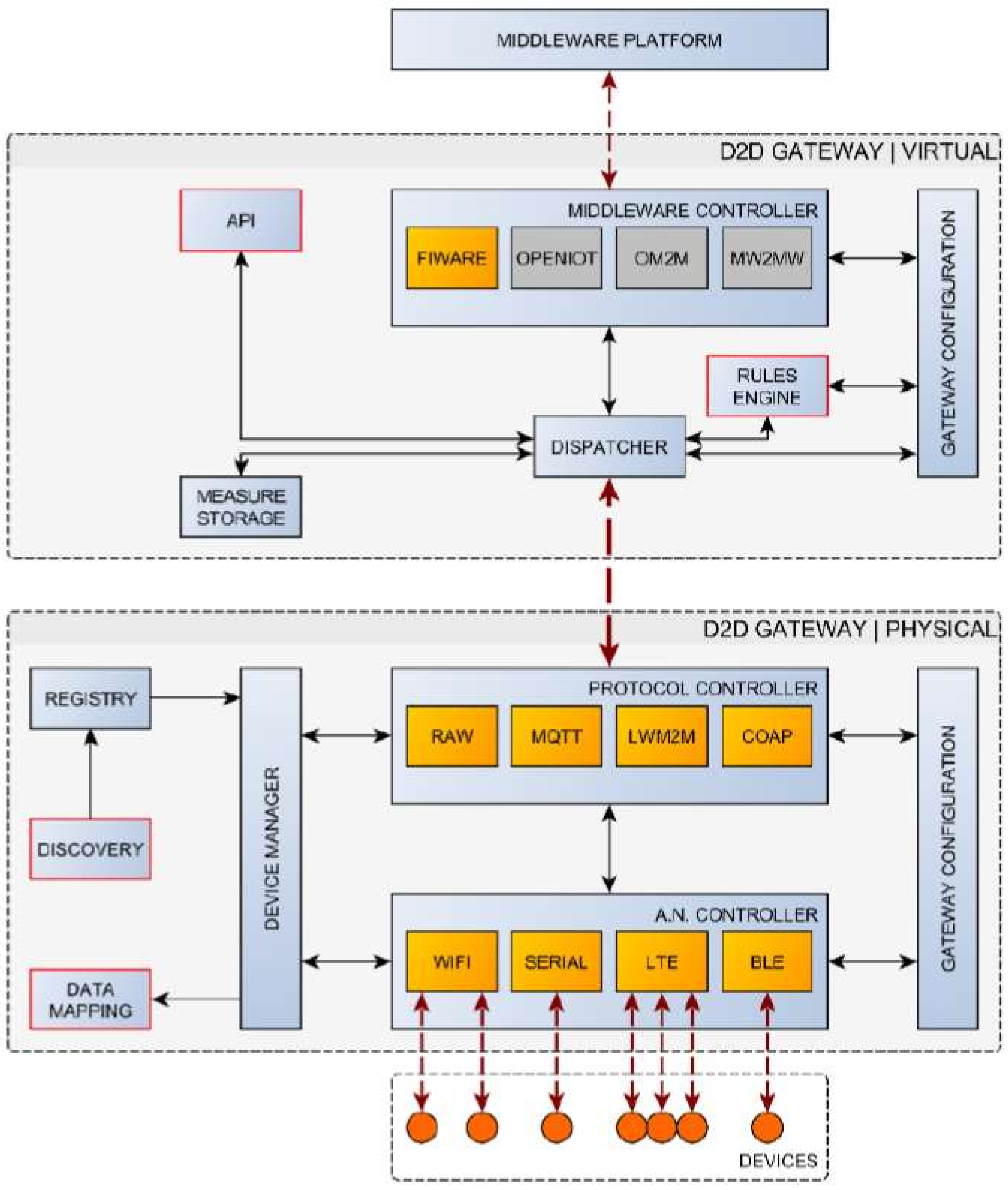


Figure 65: Gateway architecture split into two parts

3.3.3.21 Deliverables and version overview

The following table contains a deliverable list which needs to be signed before FAT testing commences.

ID	Description	Check
Documents		
1	Validation and Test reports of the IoT system components	
2	Validation and Test reports of the Pilot system components	
3	FAT document	
Hardware		
4	Zolertia RE-Mote	
5	Zolertia Orion Router	
6	BeagleBone Black	
7	DHT22 temperature and humidity sensor	

8	Aaronia Spectran HF-6065 spectrum analyzer	
Tools		
9	Wireshark 1.7.2 + Sensniff plug-in	
10	MCS Spectrum Analyzer 2.1.1a	
11	Java viewer tool 1.0	
12	Development and demonstration environments setup (in Microsoft Azure Cloud)	
13	INTER-FW portal	

Table 51: Deliverable checklist

The following table shows the software components and version of which the system release version 2.0 consists of.

ID	Description	Version	Check
IoT Data Acquisition Device			
1	INTER-HARE System - Data Acquistion Device Firmware	V1.0	
IoT Cluster Head			
2	INTER-HARE System - Cluster Head Firmware	V1.0	
IoT Relay			
3	INTER-HARE System - Relay Firmware	V1.0	
IoT Physical Gateway			
4	AN Controller	V1.0	
5	Protocol Controller	V1.0	
6	INTER-HARE System - Gateway Firmware	V1.0	
IoT Virtual Gateway			
7	Fiware	V4.2.3	
8	Virtual gateway docker	V0.2.0	
Universal container			
9	UniversAAL REST API	V3.2.1	

Table 52: Component version overview

3.3.3.22 Requirements, scenarios and use cases

The following table provides an overview of the relation between the requirements and the test(s) that validate their implementation.

ID	Description	Covered	Test code
Application			
239	Support Service choreography and Service Orchestration	**NO**	
240	Support Mash-up	**NO**	
241	Native support services	**NO**	
Architecture			
2	Scalability. Design	**YES**	T4.60.1, T4.60.2, T4.60.5, T4.60.6, T4.60.7, T4.63.1, T4.64.1, T4.62.1, T4.62.2, T4.62.3, T4.62.4, T4.62.5
6	Efficiency of the processing of information	**YES**	T4.60.7, T4.63.1, T4.64.1, T4.62.1, T4.62.2, T4.62.3, T4.62.4, T4.62.5, T4.46.1
9	Multi-level data processing support	**YES**	T4.60.7, T4.63.1, T4.64.1, T4.62.1, T4.62.2, T4.62.3, T4.62.4, T4.62.5, T4.46.1

Communications			
7	Support of opportunistic communications to avoid data loss	**YES**	T4.46.1
14	Platform independent	**YES**	T4.60.4, T4.60.5, T4.60.6, T4.60.7
15	Common IoT communication protocols must be supported.	**YES**	T4.60.4, T4.62.3
17	Dynamic network support	**Partially**	T4.60.7, T4.46.1, T4.61.1
18	Roaming across networks	**YES**	T4.60.3
39	Gateway capabilities	**YES**	T4.60.1, T4.60.4, T4.60.5, T4.60.6
45	Connectivity not based on HW identifiers	**YES**	T4.60.4, T4.60.5, T4.60.6, T4.60.7
78	Automatic and dynamic selection of communication protocol	**NO**	
80	Support multicast communication among devices	**YES**	T4.63.1, T4.64.1, T4.62.3, T4.19.1, T4.61.1
153	System cache in gateways and upper levels	**Partially**	T4.63.1, T4.64.1, T4.62.1, T4.62.2, T4.62.3, T4.62.4, T4.62.5, T4.46.1
227	Offloading	**NO**	
228	MPTCP support	**NO**	
229	SDN capabilities	**NO**	
230	6LoWPAN and RoLL protocol support	**NO**	
231	Network function virtualization	**NO**	
232	Fault tolerance	**YES**	T4.63.1, T4.64.1, T4.62.1, T4.62.2, T4.62.3, T4.62.4, T4.62.5, T4.46.1
233	Flow control and network information tracking	**YES**	T4.63.1, T4.64.1, T4.62.1, T4.62.2, T4.62.3, T4.62.4, T4.62.5, T4.46.1
Functionality			
11	Addressability and reachability	**Partially**	T4.60.6, T4.60.7, T4.63.1, T4.64.1, T4.62.1, T4.62.2, T4.62.3, T4.62.4, T4.62.5, T4.46.1, T4.19.1, T4.61.1
19	Mobility	**Partially**	T4.60.3, T4.61.1
20	Real time support	**YES**	T4.63.1, T4.64.1, T4.62.1, T4.62.2, T4.62.3, T4.62.4, T4.62.5, T4.46.1, T4.61.1
21	Real time output	**Partially**	T4.63.1, T4.64.1, T4.62.1, T4.62.2, T4.62.3, T4.62.4, T4.62.5, T4.46.1, T4.61.1
22	Unique identifier	**YES**	T4.60.6, T4.60.7, T4.46.1, T4.61.1
23	Device semantic definition	**YES**	T4.60.6, T4.60.7, T4.61.1
25	Remote programming of devices	**Partially**	T4.63.1, T4.64.1, T4.62.1, T4.62.3, T4.62.4, T4.62.5, T4.46.1, T4.19.1, T4.61.1
26	Remote device control	**Partially**	T4.63.1, T4.64.1, T4.62.1, T4.62.3, T4.62.4, T4.62.5, T4.46.1
43	IoT Services discovery	**YES**	T4.19.1, T4.61.1
89	Priority of routing and processing of critical messages upon low-priority sensor data	**YES**	T4.62.3, T4.62.4, T4.62.5, T4.46.1

179	IoT Platform Semantic Mediator supports platform to platform communication and communication between platforms and an external actor	**NO**	
183	IoT Platform Semantic Mediator does not store sensor data	**NO**	
API			
243	Gateway access API	**YES**	T4.19.1, T4.61.1
Interoperability			
4	Alignment with other IoT architectures, especially with AIOTI	**NO**	
13	Extensibility	**Partially**	T4.61.1
16	Inter-connection support	**YES**	T4.60.6, T4.60.7, T4.61.1
55	Independence of network layer	**Partially**	T4.19.1, T4.61.1
56	Secure synchronization	**YES**	T4.63.1, T4.64.1, T4.62.1, T4.62.2, T4.62.3, T4.62.4, T4.62.5, T4.46.1
86	API for proprietary systems interoperate with other systems	**NO**	
93	Standard protocol for the device communications	**Partially**	T4.62.3, T4.62.4, T4.62.5, T4.46.1
138	User device capability detection	**YES**	T4.60.4, T4.60.5, T4.60.6, T4.60.7, T4.19.1, T4.61.1
226	API for network services	**YES**	T4.19.1, T4.61.1
245	Legacy gateway integration	**NO**	
Legality			
29	Communication legislation and law	**YES**	T4.60.1, T4.60.2, T4.60.3
Middleware			
234	Provide connectors to middleware standards	**NO**	
236	Support of main Internet of Things platforms	**NO**	
237	API Middleware for interoperability between different platforms	**NO**	
238	Virtualization of common objects	**NO**	
Operational			
57	Device monitoring and self-awareness of the system	**YES**	T4.63.1, T4.64.1, T4.62.1, T4.62.2, T4.62.3, T4.62.4, T4.62.5, T4.46.1
73	Analyzing data from heterogeneous platforms	**NO**	
75	The interaction between IoT endpoints may follow the M2M communication concept	**YES**	T4.62.2, T4.62.3, T4.62.4, T4.62.5, T4.46.1
76	Interoperability between things from different administrative/management domains	**NO**	
96	Enable (automated or semi-automated) linking of relevant data sources	**NO**	
99	Mobility and crowd sensing	**NO**	
178	IoT Platform Semantic Mediator provides data and semantic interoperability functionality accessible with a set of interfaces	**NO**	
204	Support smart network resources allocation in heterogeneous wireless sensor networks	**YES**	T4.63.1, T4.64.1, T4.62.1, T4.62.2, T4.62.3, T4.62.4, T4.62.5, T4.46.1
205	Provide services to detect and predict devices' events in heterogeneous wireless networks	**YES**	T4.62.1, T4.62.3, T4.62.4, T4.62.5, T4.46.1
206	Support scalable devices using power saving communication protocols	**YES**	T4.63.1, T4.64.1, T4.62.1, T4.62.2, T4.62.3, T4.62.4, T4.62.5, T4.46.1

207	Shall support scalable network topologies	YES	T4.60.7, T4.62.3, T4.62.4, T4.62.5, T4.46.1
Performance			
72	Communication should be done using protocols that are efficient in terms of amount of exchanged information over message size	YES	T4.63.1, T4.64.1, T4.62.1, T4.62.2, T4.62.3, T4.62.4, T4.62.5, T4.46.1, T4.61.1
Security			
27	System security	YES	T4.63.1, T4.64.1, T4.62.1, T4.62.2, T4.62.3, T4.62.4, T4.62.5, T4.46.1
28	System privacy	YES	T4.63.1, T4.64.1, T4.62.1, T4.62.2, T4.62.3, T4.62.4, T4.62.5, T4.46.1
95	Robustness, resilience and availability	YES	T4.62.2, T4.62.3, T4.62.4, T4.62.5, T4.46.1
98	Data provenance	YES	T4.61.1
Semantics			
163	Design support for semantic interoperability	NO	
180	Semantic and syntactic interoperability	NO	
186	Design of required ontologies	NO	
224	Location semantic support for mobile smart objects	NO	
225	Special considerations in the semantic ontology to objects with low resources	NO	
235	Support of semantic modelling in the middleware layer	NO	
223	Semantic support for virtual smart objects	NO	
Virtualization			
242	Object/Device virtualization	YES	T4.60.7, T4.63.1, T4.64.1, T4.62.1, T4.62.2, T4.62.3, T4.62.4, T4.62.5, T4.46.1, T4.61.1
244	Gateway virtualization	YES	T4.60.4, T4.19.1, T4.61.1

Table 53: Requirements vs test mapping

The following table provides an overview of the relation between the scenarios and the test(s) that validate their implementation.

ID	Scenario name	Covered	Test code
1	Chronic disease prevention	NO	
2	IoT support for transport planning and execution	NO	
3	IoT Weighbridges	NO	
4	Monitoring reefer container	YES	All tests
5	Monitoring of containers carrying sensitive goods	NO	
6	Dynamic lighting in the port	NO	
7	SCADA port sensor system integration with IoT platforms	NO	
8	SEAMS integration with IoT platforms	NO	
9	Accident at the port area	NO	
10	Health monitoring system with passengers aboard a ferry	NO	
11	Primary prevention of cognitive decline	NO	
12	Health failure disease and mild Alzheimer disease	NO	

13	IoT interoperability for Vessel Arrivals	**NO**	
15	Surveillance systems for prevention programs	**NO**	
16	Elderly monitoring	**NO**	
17	Health monitoring system with passengers aboard a train	**NO**	
18	Containership is entering the harbour region	**NO**	
19	Transport on truck breaks down or is hijacked	**NO**	
20	Damage or problems to the container during shipment	**NO**	
21	Low risk of developing chronic diseases.	**NO**	
22	Increased risk of developing chronic diseases	**NO**	
23	High risk of developing chronic diseases	**NO**	
24	Very high risk of developing chronic diseases	**NO**	
25	Extremely high risk of developing chronic diseases	**NO**	
26	Alcohol / Drug testing for truck/ bus drivers	**NO**	
27	Vitamins intake analyser	**NO**	
28	Calories / nutrition mixer / cookware counter	**NO**	
29	Reliable control of robotic cranes and trucks in port terminals	**NO**	
30	IoT access control, traffic and operational assistance	**NO**	

Table 54: Scenario vs test mapping

At this point it is worth noting that the Monitoring Reefer Container scenario has been chosen due to the suitability of the INTER-HARE platform to gather periodic and event-based information from different locations through the wireless medium while consuming the minimum energy consumption in the employed devices.

3.3.3.23 Test environment

Introduction

To test the functionality of the INTER-HARE platform in combination with the IoT framework a representative test system is needed. The test system needs to approach the “real world” as much as possible. The pilot setups must be recreated and proven. This chapter will describe this environment and the used hardware, software, tools and platforms.

Test environment

This testbed is considered as the last stage of the design and implementation of the INTER-HARE platform and will be useful to evaluate the system’s performance in real conditions as well as to provide developers with enough information to fix and/or improve the detected failures / weaknesses. Therefore, it is possible to list the mail goals of this testbed (or test setup for integration) as:

1. To validate in a real scenario, the different hardware elements included in the INTER-HARE platform.
2. To validate the different communication protocols developed for the INTER-HARE platform and analyze their performance.
3. To validate (according to the INTER-IoT consortium) an end-to-end communication scheme with the rest of the INTER-IoT platform, thus ensuring interoperability between both environments.

Hardware components

A comprehensive list of all devices (and their consisting hardware components) considered for the INTER-HARE testbed is provided in Table 55.

Device	Subnetwork	Communication protocol	Operating frequency	Language / Software	Hardware
INTER-IoT gateway	Integration network	Connection from/to the INTER-IoT dispatcher	WiFi (2.4 GHz) / Ethernet (-)	Java over Linux OS	BeagleBoard Black + peripherals
	(1) Transport network	INTER-HARE	868 MHz	C over Contiki OS	Zolertia RE-Mote
Relay device	Transport network	INTER-HARE	868 MHz	C over Contiki OS	Zolertia RE-Mote
Cluster-head	Transport network	INTER-HARE	868 MHz + 2.4 GHz	C over Contiki OS	Zolertia Orion Router
Data Acquisition device	Acquisition network	HARE	2.4 GHz	C over Contiki OS	Zolertia RE-Mote + DHT 22 temperature and humidity sensor

(1) *Connected through the corresponding access network module*

Table 55: List of INTER-HARE testbed components

INTER-IoT gateway

The INTER-IoT gateway (specifically, its *physical* part) is considered the brain of the INTER-HARE platform and the single point of contact between the physical network and the rest of the INTER-IoT system. Due to this dual conception, it is easy to split its internal architecture into the conforming elements responsible for interacting with the network running the INTER-HARE platform (*transport network*) and the ones exchanging information with the rest of the INTER-IoT system (*integration network*).

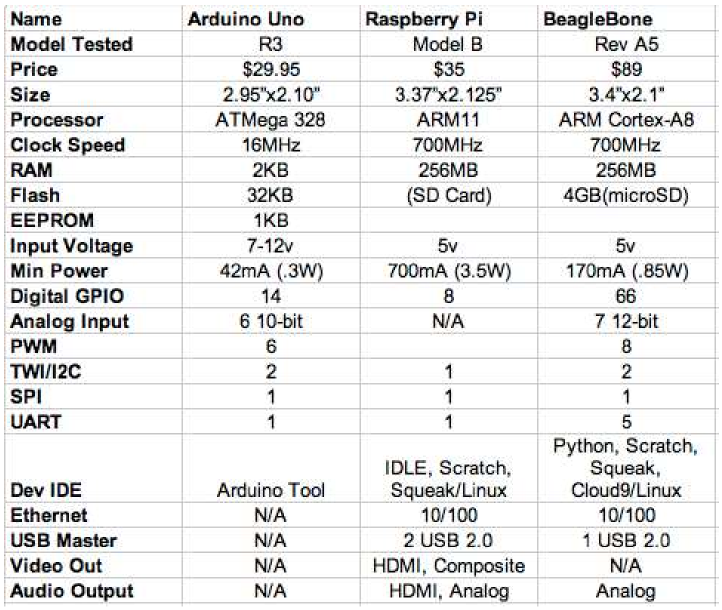
Integration network

The main element of the *integration network* part of the INTER-IoT gateway is a controller which will interact with the rest of the INTER-IoT system (or more specifically, with the *virtual* part of the INTER-IoT gateway).

Among the different controllers existing in the market, the preferred option at this stage of the project is the BeagleBone Black[16], which combines the advantages of using a small, powerful and lightweight ARM-based computer running a Linux-based OS with its facility to integrate and interact with other systems (see specifications in ***Annex C: BeagleBone Black***). However, other options like Raspberry Pi (see Table 56) could even be studied in the current project as alternative to the BeagleBone.

The controller should also include all the necessary communication interfaces to be able to communicate with both the rest of the INTER-IoT system (via a WiFi or Ethernet link) and with the *integration network* part of the INTER-IoT gateway (via a serial link).

[16] BeagleBone main website - http://beagleboard.org/bone



Name	Arduino Uno	Raspberry Pi	BeagleBone
Model Tested	R3	Model B	Rev A5
Price	$29.95	$35	$89
Size	2.95"x2.10"	3.37"x2.125"	3.4"x2.1"
Processor	ATMega 328	ARM11	ARM Cortex-A8
Clock Speed	16MHz	700MHz	700MHz
RAM	2KB	256MB	256MB
Flash	32KB	(SD Card)	4GB(microSD)
EEPROM	1KB		
Input Voltage	7-12v	5v	5v
Min Power	42mA (.3W)	700mA (3.5W)	170mA (.85W)
Digital GPIO	14	8	66
Analog Input	6 10-bit	N/A	7 12-bit
PWM	6		8
TWI/I2C	2	1	2
SPI	1	1	1
UART	1	1	5
Dev IDE	Arduino Tool	IDLE, Scratch, Squeak/Linux	Python, Scratch, Squeak, Cloud9/Linux
Ethernet	N/A	10/100	10/100
USB Master	N/A	2 USB 2.0	1 USB 2.0
Video Out	N/A	HDMI, Composite	N/A
Audio Output	N/A	HDMI, Analog	Analog

Table 56: Comparison table among Arduino UNO, Raspberry Pi and BeagleBone

Transport network

The main element of the *transport network* part of the INTER-IoT gateway is the Zolertia RE-Mote[17] (see specifications in ***Annex A: Zolertia RE-Mote***). This device will be responsible for controlling the two-tier cluster-tree network, gathering all the collected information, and transmitting this information via serial to the BeagleBone, which in turn acts as the main element of the *integration network*.

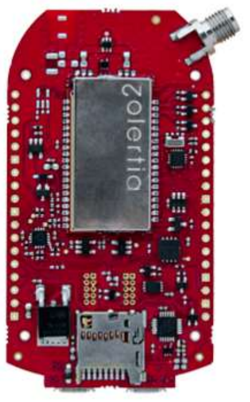


Figure 66: Zolertia RE-Mote

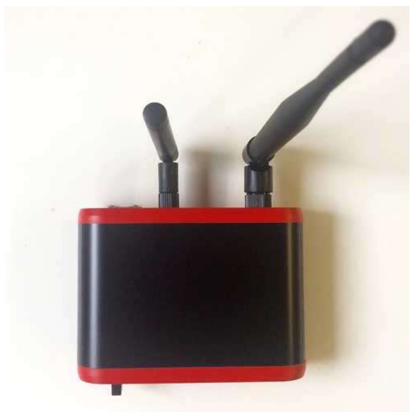


Figure 67: Zolertia Orion Router

Relay device

Only if it is necessary to extend the range coverage of the LPWAN network and retransmit the information from both the INTER-IoT gateway and the cluster-heads in the *transport network*, a relay device has been also considered in the INTER-HARE architecture. The Zolertia RE-Mote (whose specifications can be found in ***Annex A: Zolertia RE-Mote***) has been the selected technology to perform this task.

[17] Zolertia RE-Mote main website - https://github.com/Zolertia/Resources/wiki/RE-Mote

Cluster-head

Cluster-heads are entitled by the INTER-IoT gateway to manage their corresponding LPLAN in a hierarchic way. They are the only elements with a multiband radio module (working at 868 MHz and 2.4 GHz), so that they gathered the information transmitted by data acquisition devices of their own LPLAN at 2.4 GHz and retransmit it to the INTER-IoT gateway (or alternatively, to the closest relay device) at 868 MHz through the LPWAN.

In this case, the device selected to perform the role of cluster-head is the Zolertia Orion Router[18] (see specifications in ***Annex B: Zolertia Orion Router***), due to its ability to communicate with devices both from 868 MHz and 2.4 GHz band. In this case, the software application programmed in this device will make the radio module to periodically change of frequency band depending on the specific communication requirements of the system.

Optionally, and only if feasible in terms of energy consumption, cluster-heads will also contain a GPS module to inform INTER-IoT of their specific location. Communication between cluster-heads and their embedded GPS modules will be performed by using serial ports from both devices. Although the selection of the most appropriate GPS for the INTER-HARE platform will be studied during the design and implementation stage, wide-known modules from DIY environments (typically used in combination with Arduino or BeagleBone Black) will be preferably chosen. In this sense, the SparkFun GPS Logger Shield[19] is proposed as an example.

Data acquisition device

Data acquisition devices are those elements deployed directly on the area where one or more environmental variables must be monitored. In the current project, the selected data acquisition device is a Zolertia RE-Mote only working in its 2.4 GHz frequency band.

As for the additional sensor embedded in these devices, the selected one has been the DHT22 temperature and humidity sensor, whose main characteristics are included in ***Annex D: DHT22 temperature and humidity sensor***. Transmission of environmental data acquired by the DHT22 sensor is transmitted to the Zolertia RE-Mote by means of an analog connector.

Software components

Contiki 3.0 OS[20] is selected to validate the INTER-HARE platform, mainly due to its ability to easily execute multiple processes concurrently and its powerful COOJA network simulator. The INTER-HARE platform will be fully programmed in novel hardware-independent modules, one for each of the four possible network roles (INTER-IoT gateway, relay device, cluster-head device, and data acquisition device), adding all the required functionalities in order to ensure the proper operation of the *transport network*.

As for the *integration network*, the controller will be programmed by means of OSGi Java-based bundles. A list of interoperability tasks is detailed in the following lines:

1. Development of controller components according to the INTER-LAYER architecture:
    a. INTER-HARE access network module.
    b. Protocol modules (at least one from the following ones: CoAP, MQTT).
2. Communication routines between the protocol controller of the INTER-IoT physical gateway and the dispatcher.

[18] Zolertia Orion Router main website - https://github.com/Zolertia/Resources/wiki/Orion
[19] SparkFun GPS Logger Shield description - https://www.sparkfun.com/products/13750
[20] Contiki OS main website - http://contiki-os.org/

3. Connection between the INTER-HARE access network module and the HARE protocol stack through an API.
4. Access/collaboration with other elements of the INTER-IoT system to ensure proper system operation.
5. Serial communication routines with the main element of the *transport network*.

3.3.3.24 Deployment

Performance evaluation will be performed in an *ad hoc* testbed located on the 2nd floor, right wing of the *Tanger building* at UPF facilities21 (see Figure 68). The proposed testbed will consist of the elements depicted in Table 57, which will run their own version of the INTER-HARE platform.

Device	Quantity
INTER-IoT gateways	1
Cluster-heads **(868 MHz & 2.4 GHz)**	2 or more
Data acquisition devices **(2.4 GHz)**	4 or more per cluster-head (8 in total)
Relay devices (868 MHz)	At least 1

Table 57: Estimated pilot equipment

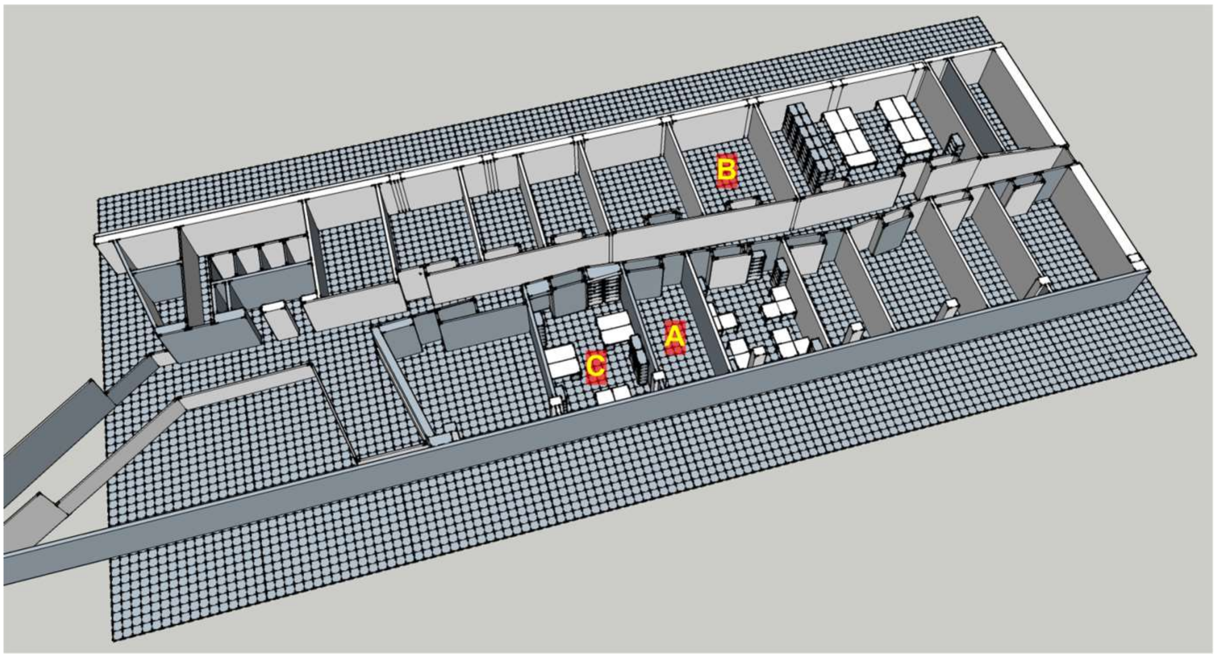


Figure 68: 3D model of UPF facilities

[21] UPF communication campus main website - https://www.upf.edu/campus/en/comunicacio/tanger.html

If no otherwise specified, all tests will be executed considering no mobility and follow these criteria:

- **'A'** office from Figure 68 will act as a reefer container
- **'B'** office from Figure 68 will also act as a reefer container
- '**C'** office from Figure 68 will act as the room where the INTER-IoT gateway is located

Dimensions of typical Maersk 40' reefer containers are 40' x 8' x 9'6"; that is 12.192 m. x 2.438 m. x 2.591 m. Dimensions of the offices that will be employed in the FAT tests are shown in Figure 69, Figure 70, and Figure 71. In addition, Table 58 summarizes offices' dimensions.

Office	Depth (m.)	Height (m.)	Width (m.)
A	6.761	2.655	3.040
B	6.500	2.655	4.686
C	7.066	2.655	4.630
Maersk 40' reefer container	12.192	2.438	2.591

Table 58: Summary of offices' dimensions

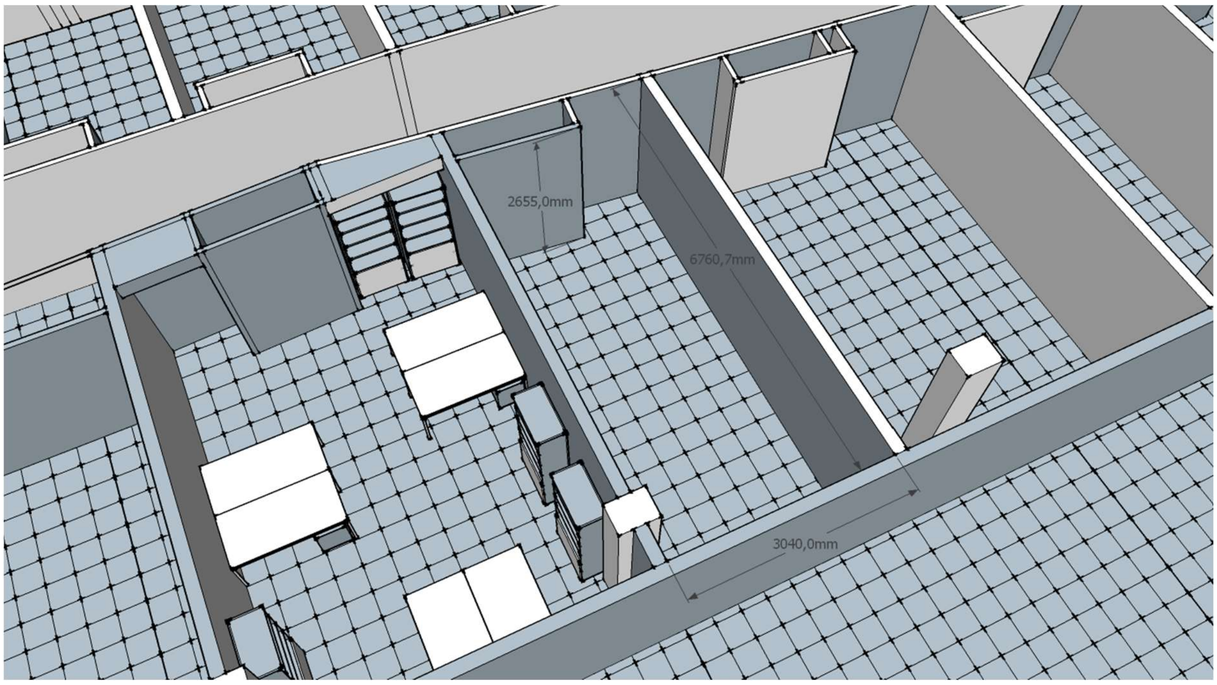


Figure 69: Office 'A' detail and dimensions

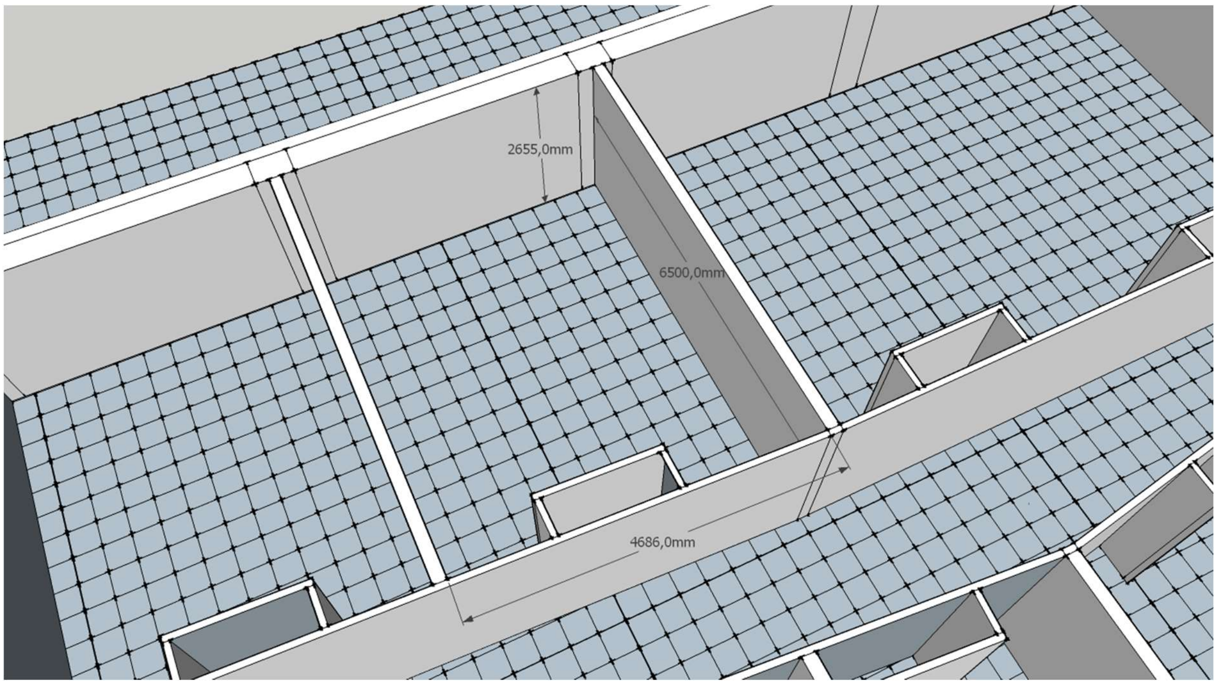


Figure 70: Office 'B' detail and dimensions

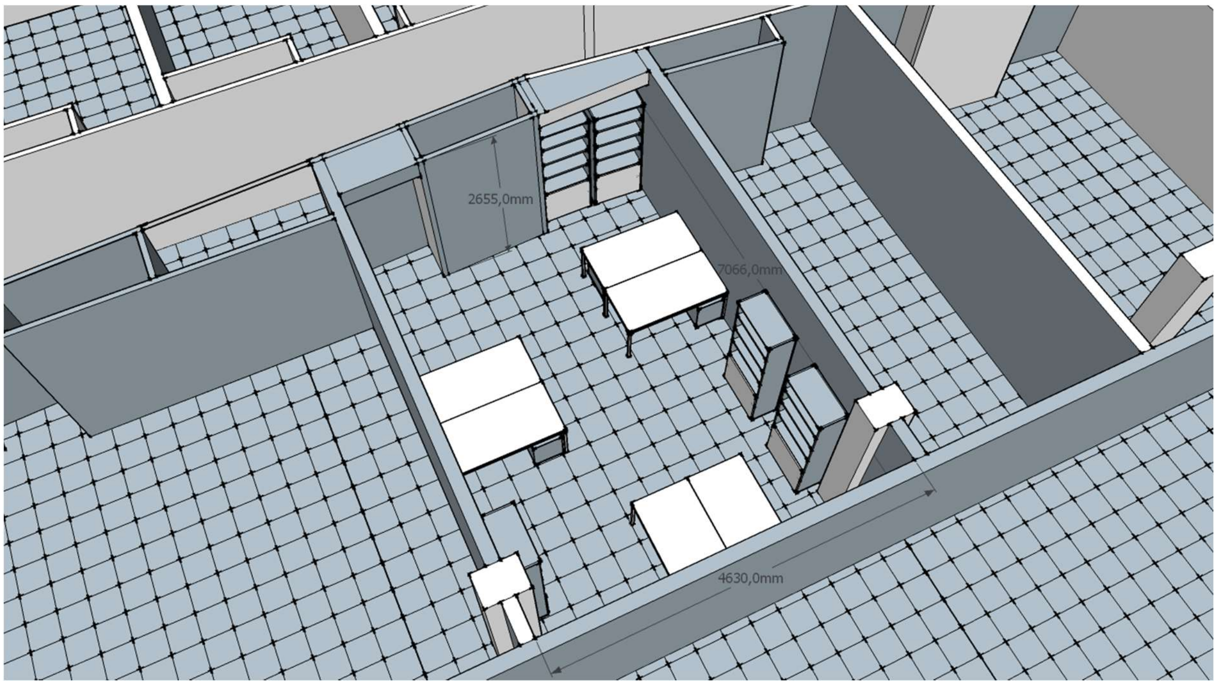


Figure 71: Office 'C' detail and dimensions

All STAs will be powered by batteries except the INTER-IoT gateway, which will be permanently powered by the PC or an alternative power supply. Additional power supply for cluster-heads will be studied depending on the incorporation of a GPS module. Similarly, additional energy requirements of relay devices will be assessed.

The testbed deployment will be based on the architecture shown in Figure 72, with two differentiated networks: the *integration network* and the *transport network* (the latter one running INTER-HARE).

An INTER-IoT physical gateway will be built, so that it will be able to simultaneously interact with both networks. While its conforming BeagleBone will act as a *client* in the *client/server* communication scheme with the rest of the INTER-IoT system in the *integration network*, its conforming Zolertia RE-Mote will act as the brain of the whole INTER-HARE platform in the *transport network*.

The *transport network* will also consist of at least two cluster-heads (A and B), optionally powered with GPS modules, which will be responsible for receiving the data gathered by the data acquisition devices deployed in their 2.4 GHz range coverage area. It is worth noting here that those two range coverage areas will be separated enough to not create interferences between them.

All data acquisition devices will include a DHT22 temperature and humidity sensor, from which they will collect environmental data through an analog connection already developed in the Zolertia RE-Motes. Lastly, they will send this collected data to their corresponding cluster-head periodically.

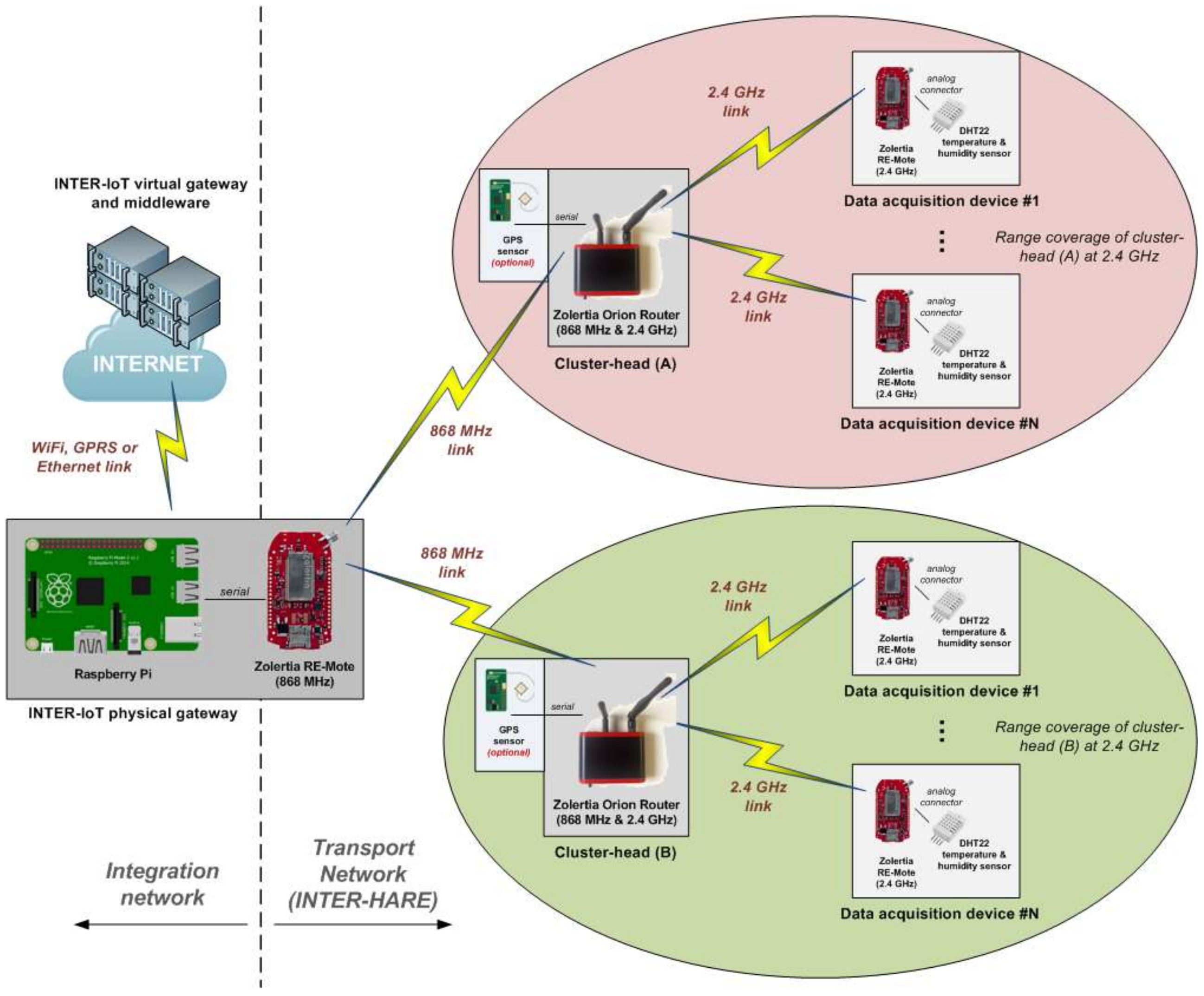

INTER-IoT virtual gateway and middleware
INTERNET
WiFi, GPRS or Ethernet link
Raspberry Pi
serial
Zolertia RE-Mote (868 MHz)
INTER-IoT physical gateway
868 MHz link
868 MHz link
GPS sensor (optional)
serial
Zolertia Orion Router (868 MHz & 2.4 GHz)
Cluster-head (A)
Cluster-head (B)
2.4 GHz link
analog connector
Zolertia RE-Mote (2.4 GHz)
DHT22 temperature & humidity sensor
Data acquisition device #1
Data acquisition device #N
Range coverage of cluster-head (A) at 2.4 GHz
Range coverage of cluster-head (B) at 2.4 GHz
Integration network
Transport Network (INTER-HARE)

Figure 72: Integration and Factory test setup overview

3.3.3.25 Test setups, tools, hooks and probes

The following paragraphs will define the used setups, tools, hooks and probes which will be referred to by the test descriptions.

Test setups

Test setup	Description
TS_01	Point-to-point topology (868 MHz)
TS_02	Point-to-point topology (2.4 GHz)
TS_03	Double CH
TS_04	Point-to-point topology (868 MHz & 2.4 GHz)
TS_05	Relay topology
TS_06	Multi-hop topology (I)
TS_07	Multi-hop topology (II)
TS_08	Full INTER-IoT topology

Table 59: Test setups summary

TS_01 Point-to-point topology (868 MHz)

The point-to-point topology consists of only 1 GW and 1 CH, both working at 868 MHz frequency band. It is mainly intended to determine the channel conditions at this frequency band and the maximum coverage range achievable.

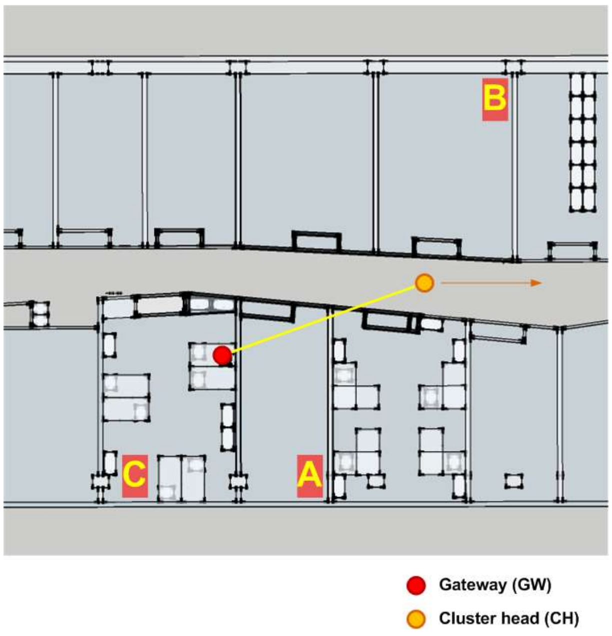


Figure 73: TS_01 network topology

TS_02 Point-to-point topology (2.4 GHz)

Similar to the previous topology, only two devices are used in this setup: 1 CH and 1 DAD. In this case, both devices work at 2.4 GHz frequency band and their main purpose is to determine the channel conditions.

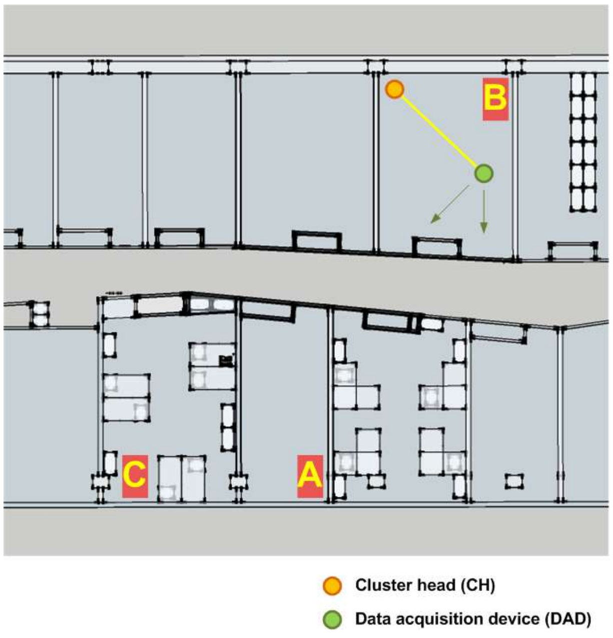


Figure 74: TS_02 network topology

TS_03 Double CH

Two CH are used in this setup in order to determine the interference level produced by them on a single DAD, which may vary its position on the scenario.

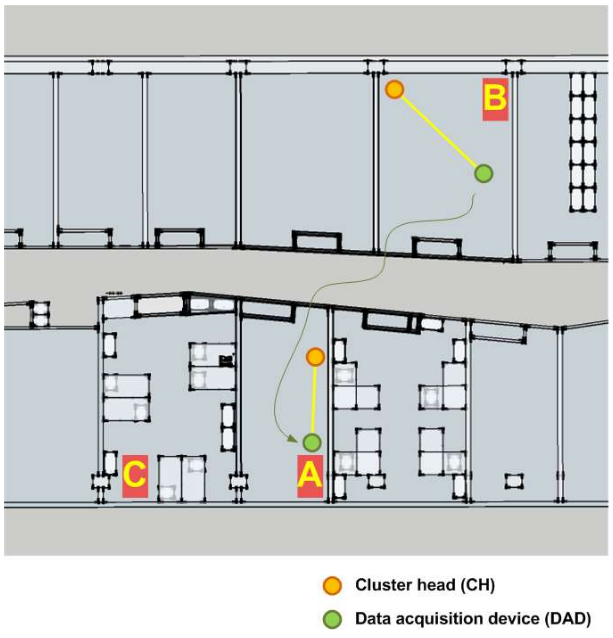


Figure 75: TS_03 network topology

TS_04 Point-to-point topology (868 MHz & 2.4 GHz)

This topology establishes full communication between a GW and a DAD through an intermediate CH. While the GW is placed in the room 'C', the other two devices are placed in the room 'B'.

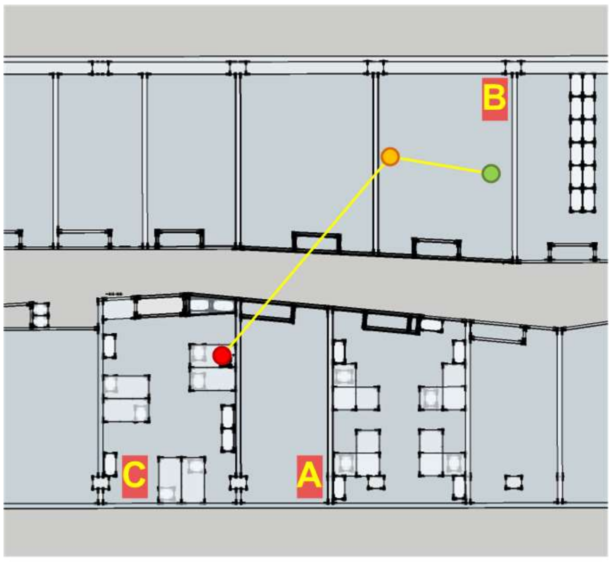


Figure 76: TS_04 network topology

TS_05 Relay topology

As indicated by its name, the relay topology incorporates a relay in the room 'A' which extends communications at 868 MHz. More specifically, its main purpose is to retransmit information from (to) the GW of room 'C' to (from) the CH of room 'B'. In addition, some DADs are located in room 'B'.

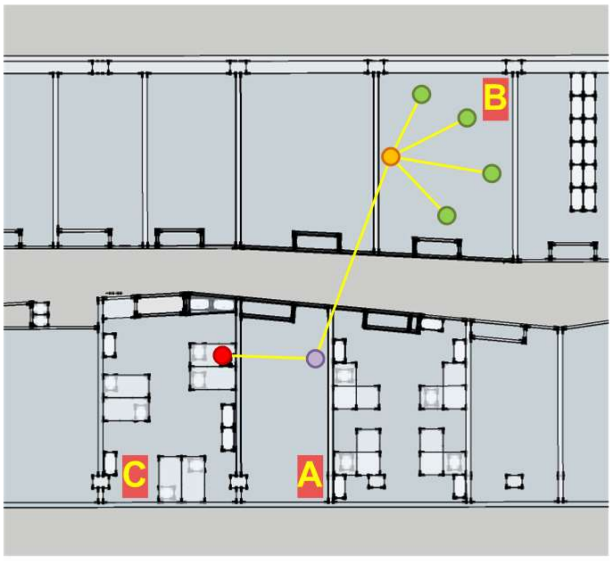


Figure 77: TS_05 network topology

TS_06 Multi-hop topology (I)

The multi-hop topology (I) replaces the relay of the TS_05 with a new CH in room 'A'. Some additional DADs are placed in room 'A', too.

It is worth noting here that CH of room ‘A’ should aggregate data from CH of room ‘B’ to the data produced by the new DADs placed in room ‘A’.

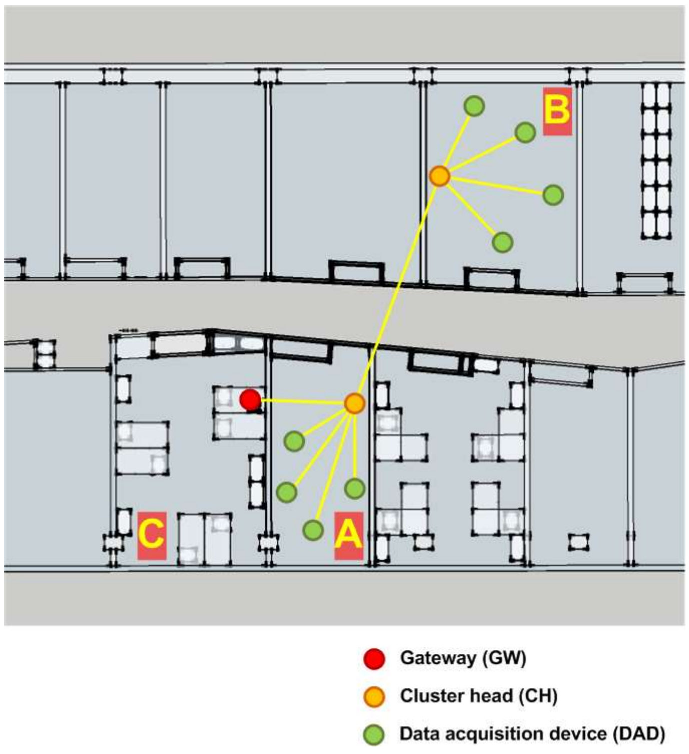


Figure 78: TS_06 network topology

TS_07 Multi-hop topology (II)

The last test setup establishes direct communication between the two CH and the GW. Information gathered from DADs could be this way transmitted directly to the GW.

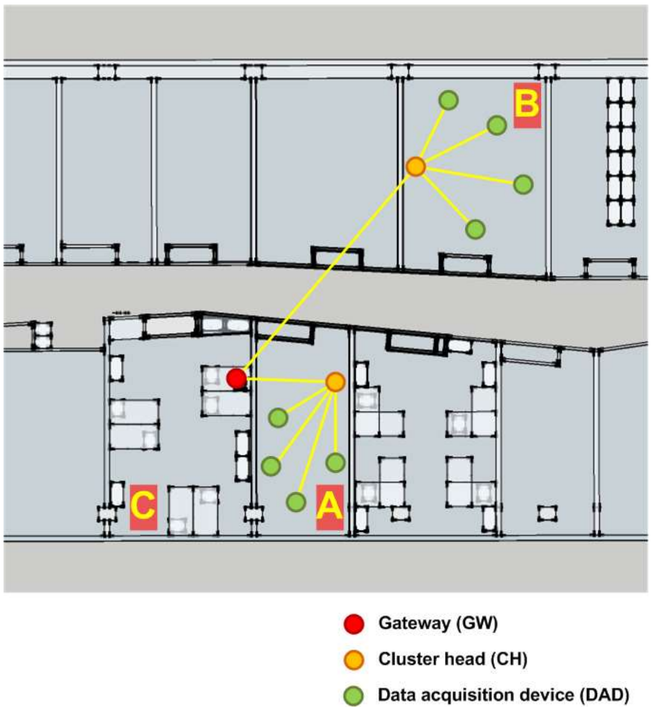


Figure 79: TS_07 network topology

TS_08 Full INTER-IoT topology

Lastly, all the elements of the INTER-HARE platform are interconnected in this test setup, from the data acquisition devices to the virtual gateway. As described in the system's architecture, two different networks collaborate to ensure end-to-end communications:

A. The transport network, consisting of the DADs, the CHs and the GW.
B. The integration network, consisting of the GW (considered in the deployment as the *physical* gateway), and the virtual gateway with its related middleware.

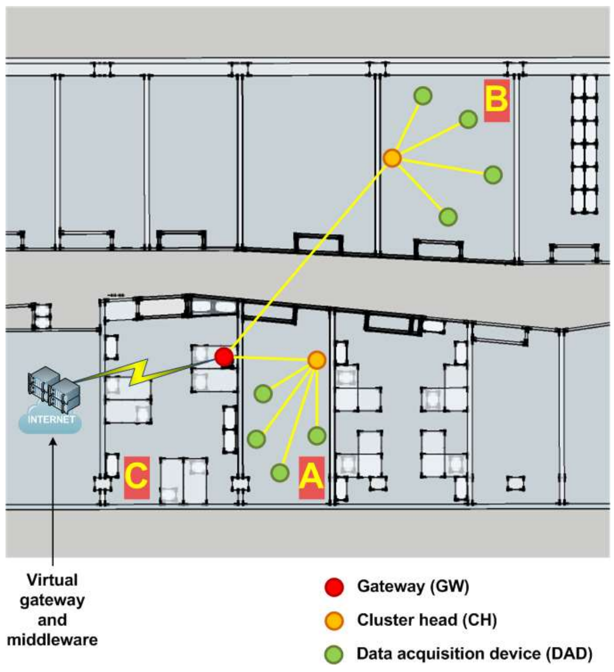


Figure 80: TS_08 network topology

Test tools

Test setup	Description
TT_01	Java viewer tool
TT_02	Zolertia RE-Mote leds
TT_03	Aaronia Spectran HF-6065 spectrum analyzer
TT_04	MCS Spectrum analyzer
TT_05	Wireshark
TT_06	Development and demonstration environments setup

Table 60: Test tools summary

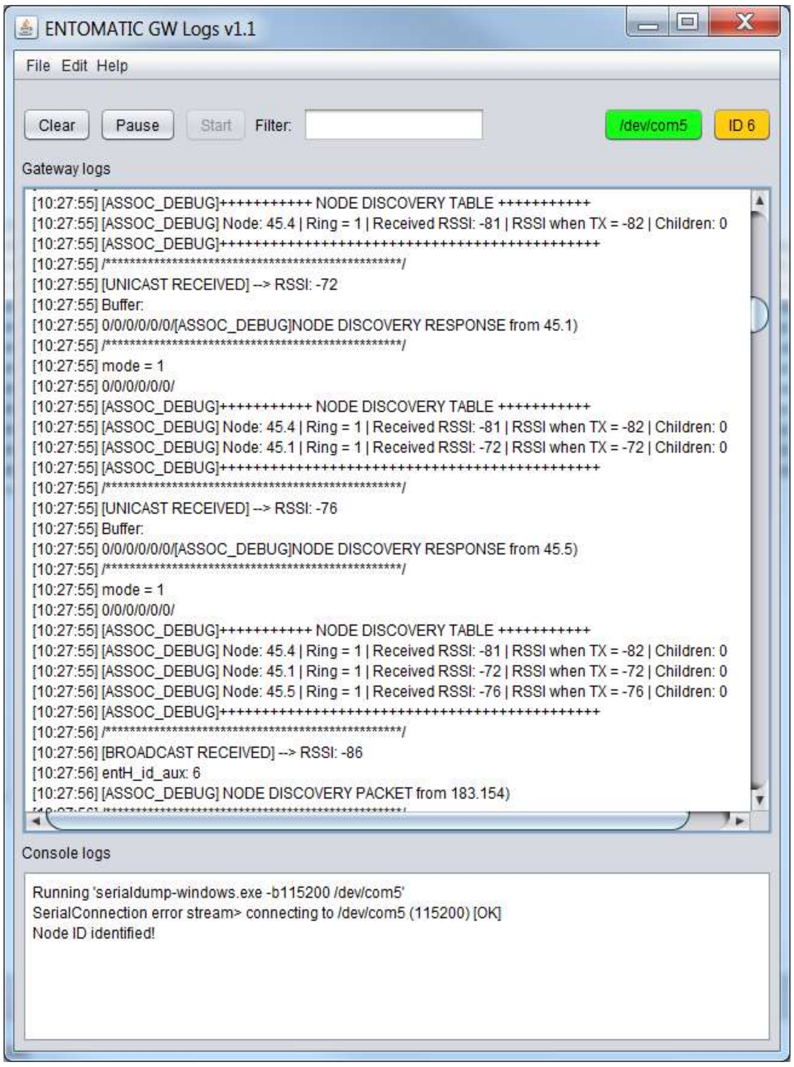
TT_01 Java viewer tool

To facilitate the debug of the programmed protocols, code will contain messages and flags that will be transmitted via serial output by each system's device. All Zolertia platforms have a serial-to-USB converter on-board, meaning that devices can be connected to one USB port of a PC without any additional hardware but a cable. Serial output implemented in Contiki OS is supported by the standard C library API for printing.

However, to see the messages generated by devices is necessary to maintain a Linux terminal session active, which in turn makes difficult to store the received information. For this reason, it has been necessary to define the requirements of a new monitoring tool:

- Multi-platform (Windows, Linux, Mac OS)
- No necessity of Contiki OS previously installed
- Connection via USB
- Log recording
- Data filtering
- Debugging

The result is a Java-based monitoring tool able to gather, show and store all the log messages emitted by an STA directly connected to a PC via an USB cable. As can be seen in Figure 81, the tool informs about the COM port in which the STA is connected (for instance, /dev/com5) and show the ID of the STA once it has been properly recognized (in the example, ID is 6). Different filters based on character strings can be applied on the gathered information in order to show only some relevant data or those debugging flags placed in the code run by STAs. Information can also be exported to a .txt file for post-processing purposes.



```
[10:27:55] [ASSOC_DEBUG]+++++++++++ NODE DISCOVERY TABLE +++++++++++
[10:27:55] [ASSOC_DEBUG] Node: 45.4 | Ring = 1 | Received RSSI: -81 | RSSI when TX = -82 | Children: 0
[10:27:55] [ASSOC_DEBUG]++++++++++++++++++++++++++++++++++++++++++++++++
[10:27:55] /**************************************************/
[10:27:55] [UNICAST RECEIVED] --> RSSI: -72
[10:27:55] Buffer:
[10:27:55] 0/0/0/0/0/0/[ASSOC_DEBUG]NODE DISCOVERY RESPONSE from 45.1)
[10:27:55] /**************************************************/
[10:27:55] mode = 1
[10:27:55] 0/0/0/0/0/0/
[10:27:55] [ASSOC_DEBUG]+++++++++++ NODE DISCOVERY TABLE +++++++++++
[10:27:55] [ASSOC_DEBUG] Node: 45.4 | Ring = 1 | Received RSSI: -81 | RSSI when TX = -82 | Children: 0
[10:27:55] [ASSOC_DEBUG] Node: 45.1 | Ring = 1 | Received RSSI: -72 | RSSI when TX = -72 | Children: 0
[10:27:55] [ASSOC_DEBUG]++++++++++++++++++++++++++++++++++++++++++++++++
[10:27:55] /**************************************************/
[10:27:55] [UNICAST RECEIVED] --> RSSI: -76
[10:27:55] Buffer:
[10:27:55] 0/0/0/0/0/0/[ASSOC_DEBUG]NODE DISCOVERY RESPONSE from 45.5)
[10:27:55] /**************************************************/
[10:27:55] mode = 1
[10:27:55] 0/0/0/0/0/0/
[10:27:55] [ASSOC_DEBUG]+++++++++++ NODE DISCOVERY TABLE +++++++++++
[10:27:55] [ASSOC_DEBUG] Node: 45.4 | Ring = 1 | Received RSSI: -81 | RSSI when TX = -82 | Children: 0
[10:27:55] [ASSOC_DEBUG] Node: 45.1 | Ring = 1 | Received RSSI: -72 | RSSI when TX = -72 | Children: 0
[10:27:56] [ASSOC_DEBUG] Node: 45.5 | Ring = 1 | Received RSSI: -76 | RSSI when TX = -76 | Children: 0
[10:27:56] [ASSOC_DEBUG]++++++++++++++++++++++++++++++++++++++++++++++++
[10:27:56] /**************************************************/
[10:27:56] [BROADCAST RECEIVED] --> RSSI: -86
[10:27:56] entH_id_aux: 6
[10:27:56] [ASSOC_DEBUG] NODE DISCOVERY PACKET from 183.154)
```

```
Running 'serialdump-windows.exe -b115200 /dev/com5'
SerialConnection error stream> connecting to /dev/com5 (115200) [OK]
Node ID identified!
```

Figure 81: INTER-HARE monitoring tool based on Java

RE-Mote leds

LEDs are a simple but important tool to communicate with users or to debug programs. Each one of the hardware platforms over which the INTER-HARE platform will contain its own set of led codes.

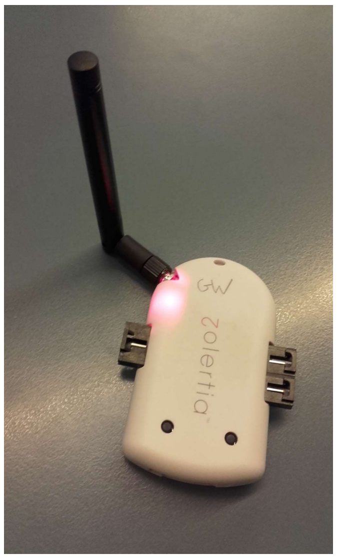


Figure 82: A Zolertia RE-Mote device with its led switched on in red

TT_03 Aaronia Spectran HF-6065 spectrum analyzer

The Aaronia Spectran HF-6065 spectrum analyzer measures the magnitude of an input signal versus frequency within the full frequency range of the instrument. The primary use is to measure the power of the spectrum of known and unknown signals.

Figure 83: Aaronia Spectran HF-6065 spectrum analyzer

The spectrum analyzer will be very useful to determine the strength of the different signals emitted by the devices employed in the different tests as well as to determine possible interferences between signals.

TT_04 MCS Spectrum analyzer

The Aaronia MCS software is an advanced control and reporting software for the Spectran series of Aaronia spectrum analyzer devices. The main features of the software are detailed in the following lines:

- Runs with any operation system like MAC OS, Linux and Windows.
- Real-Time remote control with any Spectran Spectrum Analyzer.
- Supports an unlimited number of Pre-Compliance limits displays like EN55011, EN55022, etc., and including various separate limit curves and bar displays.
- Multi-window support.
- Powerful *undo* feature.
- Channel and provider display.
- Fully customizable skins and look.
- Report and record function.

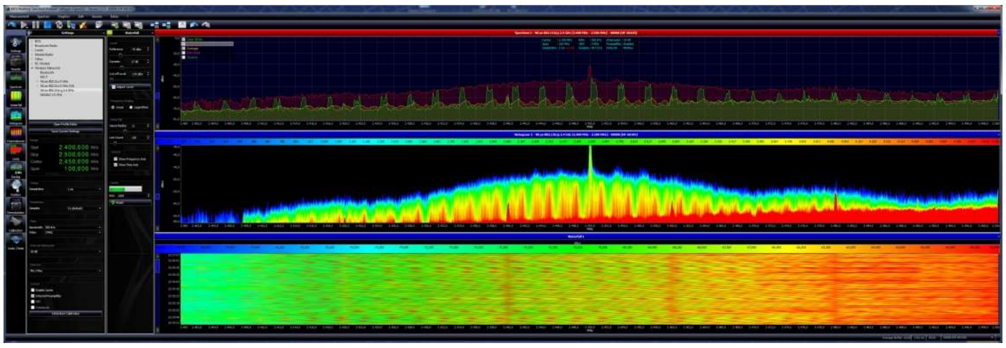


Figure 84: Screenshot of MCS spectrum analyzer

TT_05 Wireshark

Wireshark is a free and open source packet analyzer. It is used for network troubleshooting, analysis, software and communications protocol development, and education. As both the LPWAN and the LPLANs considered in the project are based on the IEEE 802.15.4 PHY layer, it has been necessary to adapt the Wireshark packet analyzer to this specific technology.

Sensniff [22] has been the selected tool to capture live traffic in IEEE 802.15.4 networks. It consists of two components:

- **Peripheral**: This is an embedded device with an IEEE 802.15.4 transceiver which captures all network frames and streams them over to the host.
- **Host**: This is a python script which runs on a PC. It reads network packets captured by the peripheral, converts them to PCAP and pipes them to Wireshark.

Other than network packet capture, the host can send commands to the peripheral to achieve secondary functionality e.g. change radio channel.

TT_06 Development and demonstration environments setup

As defined in ***Deliverable 3.2. Methods for Interoperability and Integration***, to enable an homogeneous controlled environment for the development of the different layers, a development cloud environment has been set up.

This environment is based in the Microsoft Azure Cloud and comprises a set of 7 commodity servers to decouple the different software modules development and, at the same time, making the latest features available to be tested. These servers are all accessed through a unique stepping-stone server. The access to this environment is securized through Microsoft Azure standard security mechanisms.

[22] Sensniff main website in Github - https://github.com/g-oikonomou/sensniff

3.3.3.26 Test hooks

Test setup	Description
TH_01	Iterative switching on
TH_02	Continuous traffic injection
TH_03	Random traffic injection
TH_04	Error addition
TH_05	Occasional switching off
TH_06	User browsing

Table 61: Test hooks summary

TH_01 Iterative switching on

A device is switched on repeatedly, executing its initialization process. Although each considered device (gateway, cluster head, relay and data acquisition device) is programmed with different functions, they all will try firstly to send a discovery message to their immediate parent and start their corresponding mechanism in order to be registered in the network.

In some tests, this iterative switching on will be performed in the CHs or the DADs from increasingly distances to their closer GW or CH, respectively. It will be useful to determine the maximum distance from which that device can establish a connection with its parent.

TH_02 Continuous traffic injection

Continuous traffic injection consists in the periodic data acquisition and/or simulation by DADs and its corresponding transmission to their immediate parent. By following the beacon scheduling of the INTER-HARE platform, DADs will send a data packet every cycle.

TH_03 Random traffic injection

Random traffic injection is conceived as random processes executed in both the GW and the DADs, generating query-driven and event-driven traffic, respectively.

- Query-driven traffic is generated at the GW and consists of a data request which must be transmitted to a specific DAD of the network (for instance, a user asks for the temperature value of a determined station).
- Event-driven traffic is generated at DADs when a predetermined threshold has been surpassed (for instance, the temperature sensor has detected a value over 40 °C).

TH_04 Error addition

The whole system can be altered with the arbitrarily introduction of a certain error probability when sending both application packets and their corresponding ACKs (it is worth noting here that neither messages implied in the association process nor statistics packets are affected by arbitrary generated errors to not artificially disturb the network setup nor the collection of operation information).

Errors are generated through a uniformly distributed random variable according to mean error values from four different error configurations (see Table 62). Before sending a message, STAs compute this value and discard messages accordingly.

Error configuration	Data Error	ACK Error
$E_{0/0}$	0%	0%
$E_{10/5}$	10%	5%
$E_{20/10}$	20%	10%
$E_{30/15}$	30%	15%

Table 62: Definition of error configurations

TH_05 Occasional switching off

In some tests, an STA will be deliberately switched off in order to analyze the behavior of the rest of the network to overcome this issue and rearrange the routing paths to the GW.

TH_06 User browsing

The user of the platform freely browses the different network configuration options and receives information regarding the network state and main functionalities.

3.3.3.27 Test probes

Test setup	Description
TP_01	Gateway log
TP_02	Cluster Head log
TP_03	Relay log
TP_04	Data Acquisition Device log
TP_05	INTER-FW portal

Table 63: Test probes summary

TP_01 Gateway log

The Gateway log includes information regarding both the transport and the integration network. It is stored in separated files according to the file system implemented in the device.

A summarized version of this log, only containing net information and network events could be optionally stored in a microSD connected to the device.

TP_02 Cluster Head log

The Cluster Head log is a file created by this device where all the gathered information as well as all hooks and flags are stored. It also includes information about the DADs directly associated to that device.

TP_03 Relay log

Similarly, to other logs, the Relay log is a file which contains all the information received and transmitted by this device.

TP_04 Data Acquisition Device log

The Data Acquisition Device log is created each time this device is started and includes information with regard to the data acquired from sensors and to the connection with the corresponding CH.

TP_05 INTER-FW portal

As described in the example demo of ***Subsection 4.2.6. Demo*** from ***Deliverable 3.2. Methods for Interoperability and Integration,*** the INTER-FW portal is used to demonstrate how the data flows through all the test setup.

3.3.3.28 Test description

S4 – Monitoring reefer containers

The objective of this scenario is to interoperate and use a shipping line's container IoT platform that is currently able to monitor reefer containers along its journey with the IoT platforms of the road hauliers or container terminals. This integration will allow a quick reaction in case of an alarm regarding the functioning of refrigerated goods and it will benefit container terminals and road haulier companies (drivers in this case) to avoid the periodic human inspection required for reefer containers.

Interoperability in this scenario is required to connect the shipping lines, the container terminals and the road hauliers IoT platforms.

The resulting service will be obtained by the integration of:

- Carrier IoT platform who is owner of the container
- Container terminal IoT platform
- Road haulier cloud IoT platform

In the following lines it is described how the port environment (i.e., reefer containers and container terminal) is emulated in an *ad-hoc* testbed located on UPF facilities and the list of considered use cases together with their associated tests. As described in Subsection 3.3.3.24, while the smallest rooms of our offices are considered as *reefer containers*, the *container terminal* has been located in a larger one.

Use case	Associated test
[19] User interacts with sensors or devices	T4.19.1 User interaction
[46] Device failure detection	T4.46.1 Resilience against failures
[60] Device registry	T4.60.1 Range coverage at 868 MHz
	T4.60.2 Range coverage at 2.4 GHz
	T4.60.3 Interference analysis at 2.4 GHz
	T4.60.4 (Physical) Gateway registration
	T4.60.5 Container registration
	T4.60.6 Sensor registration
	T4.60.7 Multiple sensor registration
[61] Platform Configuration on the Gateway	T4.61.1 Platform setup and simulation
[62] Device (sensor) triggers information	T4.62.1 Event-driven data delivery model test
	T4.62.2 Continuous data delivery model test
	T4.62.3 Hybrid data delivery model test
	T4.62.4 Data aggregation test
	T4.62.5 Relay operation test

[63] Platform requests information from a device (sensor)	T4.63.1 Query-driven data delivery model test (requests)
[64] Platform sends information to device (actuator)	T4.64.1 Query-driven data delivery model test (responses)

Table 64: Summary of FAT tests and definition

Concept	Test code	Test name	Test setup	Tools	Hooks	Probes	Outcomes
A. Range coverage	T4.60.1	Range coverage at 868 MHz	TS_01	TT_01, TT_02, TT_03, TT_04	TH_01	TP_01, TP_02	Max. distance
	T4.60.2	Range coverage at 2.4 GHz	TS_02	TT_01, TT_02, TT_03, TT_04	TH_01	TP_02, TP_04	Max. distance
	T4.60.3	Interference analysis at 2.4 GHz	TS_03	TT_01, TT_02, TT_03, TT_04, TT_05	TH_01	TP_02, TP_04	Interference map
B. Association & Registration	T4.60.4	(Physical) Gateway registration	TS_04	TT_01, TT_02, TT_06	TH_01	TP_01, TP_05	Pass/Fail, Assoc. delay
	T4.60.5	Container registration	TS_04	TT_01, TT_02, TT_06	TH_01	TP_01, TP_02, TP_05	
	T4.60.6	Sensor registration	TS_04	TT_01, TT_02, TT_06	TH_01	TP_01, TP_02, TP_04, TP_05	
	T4.60.7	Multiple sensor registration	TS_07	TT_01, TT_02, TT_06	TH_01	TP_01, TP_05	
C. Data transmission	T4.63.1	Query-driven data delivery model test (requests)	TS_07	TT_01, TT_06	TH_03, TH_04	TP_01, TP_05	

	T4.64.1	Query-driven data delivery model test (responses)	TS_07	TT_01, TT_06	TH_03, TH_04	TP_01, TP_05	Pass/Fail, PDR (%), Delay, Throughput, PRM, Energy consumption
	T4.62.1	Event-driven data delivery model test	TS_07	TT_01, TT_06	TH_03, TH_04	TP_01, TP_05	
	T4.62.2	Continuous data delivery model test	TS_07	TT_01, TT_06	TH_02, TH_04	TP_01, TP_05	
	T4.62.3	Hybrid data delivery model test	TS_07	TT_01, TT_06	TH_02, TH_03, TH_04	TP_01, TP_05	
	T4.62.4	Data aggregation test	TS_06	TT_01, TT_06	TH_02, TH_03, TH_04	TP_01, TP_05	
	T4.62.5	Relay operation test	TS_05	TT_01, TT_05, TT_06	TH_02, TH_03, TH_04	TP_01, TP_03, TP_05	
D. Resilience	T4.46.1	Resilience against failures	TS_07	TT_01, TT_06	TH_02, TH_03, TH_05	TP_01, TP_05	Pass/Fail
E. Integration network	T4.19.1	User interaction	TS_08	TT_02, TT_06	TH_02, TH_03, TH_06	TP_05	Pass/Fail
	T4.61.1	Platform setup and simulation	TS_08	TT_02, TT_06	TH_01, TH_02, TH_03, TH_05, TH_06	TP_05	Pass/Fail

Table 65: Diagram compiling the different test setups

CONCEPT	TEST CODE	ARCHITECTURE			COMMUNICATIONS											FUNCTIONALITY										API	INTEROPERABILITY							LEGALITY	OPERATIONAL						PERFORMANCE	SECURITY				VIRTUALIZATION	
		2	6	9	7	14	15	17	18	39	45	80	153	232	233	11	19	20	21	22	23	25	26	43	89	243	13	16	55	56	93	138	226	29	57	75	204	205	206	207	72	27	28	95	98	242	244
A. Range Coverage	T4.60.1	X								X																								X													
	T4.60.2	X																																X													
	T4.60.3								X								X																	X													
B. Association & Registration	T4.60.4					X	X			X	X																					X															X
	T4.60.5	X				X				X	X																					X															
	T4.60.6	X				X				X	X					X				X	X							X				X															
	T4.60.7	X	X	X		X		X			X				X	X				X	X							X				X								X						X	
C. Data transmission	T4.63.1	X	X	X								X	X	X	X	X		X	X			X	X							X					X		X		X		X	X	X			X	
	T4.64.1	X	X	X								X	X	X	X	X		X	X			X	X							X					X		X		X		X	X	X			X	
	T4.62.1	X	X	X									X	X	X	X		X	X			X	X							X					X		X	X	X		X	X	X			X	
	T4.62.2	X	X	X									X	X	X	X		X	X											X					X	X	X		X		X	X	X	X		X	
	T4.62.3	X	X	X			X					X	X	X	X	X		X	X			X	X		X					X	X				X	X	X	X	X	X	X	X	X	X		X	
	T4.62.4	X	X	X									X	X	X	X		X	X			X	X		X					X	X				X	X	X	X	X	X	X	X	X	X		X	
	T4.62.5	X	X	X									X	X	X	X		X	X			X	X		X					X	X				X	X	X	X	X	X	X	X	X	X		X	
D. Resilience	T4.46.1		X	X	X			X					X	X	X	X		X	X	X		X	X		X					X	X				X	X	X	X	X	X	X	X	X	X		X	
E. Integration network	T4.19.1											X				X						X		X		X			X			X	X														X
	T4.61.1							X				X				X	X	X	X	X	X	X		X		X	X	X	X			X	X								X				X	X	X

Table 66: List of requirements to be analyzed in each test

3.3.3.29 U19 – User interacts with sensors or devices

The whole platform is accessed remotely by the user, who can change some configuration settings. The user experience is analyzed.

T4.19.1 User interaction

ID	T4.19.1
Test	User experience analysis when configuring the INTER-HARE platform
Type	E. Integration network
Setup	Need test setup TS_08
Start	All DADs, CHs and the GW are already registered.
Req.	[80], [11], [25], [43], [243], [55], [138], [226], [244]
Input	Test hooks TH_02, TH_03, and TH_06
Output	Check system's ability to configure the parameters selected by the user.
Logs	E. Integration\T4.19.1_ux.txt
Outcome	Pass / Fail

3.3.3.30 U46 – Device failure detection

The system is able to detect problems in intermediate devices (CHs and DADs) and to act consequently, by reconstructing routing paths and ensuring data transmissions.

T4.46.1 Resilience against failures

ID	T4.46.1
Test	INTER-HARE resilience analysis against failures
Type	D. Resilience
Setup	Need test setup TS_07
Start	All DADs, CHs and the GW are already registered.
Req.	[6], [9], [7], [17], [153], [232], [233], [11], [20], [21], [22], [25], [26], [89], [56], [93], [57], [75], [204], [205], [206], [207], [72], [27], [28], [95], [242]
Input	Test hooks TH_02, TH_03, and TH_05
Output	Check if the system is able to rebuild routing routes after one (or more) DAD (or CH) switches off. Check if the system is able to maintain high reliability levels after one (or more) DAD (or CH) switches off.
Logs	D. Resilience\T4.46.1_resilience.txt

Outcome	Pass / Fail

3.3.3.31 U60 – Device registry

Devices are able to determine if they are within the range coverage of their immediate parent. If so, they execute the association process to be part of the network by receiving the corresponding network address and listening to the schedule beacons.

T4.60.1 Range coverage at 868 MHz

ID	T4.60.1
Test	Analysis of range coverage at 868 MHz
Type	A. Range coverage
Setup	Need test setup TS_01
Start	GW located in a fixed position. CH initially located close to the GW. CH is moved further from the GW.
Req.	[2], [39], [29]
Input	Test hook TH_01 in different CH positions
Output	CH inside or outside the range coverage of the GW.
Logs	A. Range\T4.60.1_868.txt
Outcome	Max. distance

T4.60.2 Range coverage at 2.4 GHz

ID	T4.60.2
Test	Analysis of range coverage at 2.4 GHz
Type	A. Range coverage
Setup	Need test setup TS_02
Start	CH located in a fixed position. DAD initially located close to the CH. DAD is moved further from the CH.
Req.	[2], [29]
Input	Test hook TH_01 in different DAD positions

Output	DAD inside or outside the range coverage of the CH.
Logs	A. Range\T4.60.2_24.txt
Outcome	Max. distance

T4.60.3 Interference analysis at 2.4 GHz

ID	T4.60.3
Test	Evaluation of interference between two LPLANs
Type	A. Range coverage
Setup	Need test setup TS_03
Start	2 CHs located in different rooms ('containers'). DAD initially located close to one CH. DAD is moved to the other CH.
Req.	[18], [19], [29]
Input	Test hook TH_01 in different DAD positions
Output	DAD receives a certain RSSI level from one or both CHs.
Logs	A. Range\T4.60.3_interference.txt
Outcome	Interference 'map'

T4.60.4 (Physical) Gateway registration

ID	T4.60.4
Test	GW registration into the INTER-IoT network
Type	B. Association & Registration
Setup	Need test setup TS_04
Start	GW located in a fixed position.
Req.	[14], [15], [39], [45], [138], [244]
Input	Test hook TH_01
Output	Check proper GW registration into the INTER-IoT network.
Logs	B. Association\T4.60.4_gw_reg.txt
Outcome	Pass / Fail

	Assoc. delay

T4.60.5 Container registration

ID	**T4.60.5**
Test	CH (container) registration into the INTER-IoT network
Type	B. Association & Registration
Setup	Need test setup TS_04
Start	GW & CH located in a fixed position.
Req.	[2], [14], [39], [45], [138]
Input	Test hook TH_01
Output	Check proper CH registration into the INTER-IoT network.
Logs	B. Association\T4.60.5_ch_reg.txt
Outcome	Pass / Fail
	Assoc. delay

T4.60.6 Sensor registration

ID	**T4.60.6**
Test	DAD (sensor) registration into the INTER-IoT network
Type	B. Association & Registration
Setup	Need test setup TS_04
Start	GW, CH & DAD located in a fixed position.
Req.	[2], [14], [39], [45], [11], [22], [23], [16], [138]
Input	Test hook TH_01
Output	Check proper DAD registration into the INTER-IoT network.
Logs	B. Association\T4.60.6_dad_reg.txt
Outcome	Pass / Fail
	Assoc. delay

T4.60.7 Multiple sensor registration

ID	T4.60.7
Test	Multiple DAD (sensor) registration into the INTER-IoT network
Type	B. Association & Registration
Setup	Need test setup TS_07
Start	1 GW, 2 CHs & multiple DADs located in a fixed position.
Req.	[2], [6], [9], [14], [17], [45], [233], [11], [22], [23], [16], [138], [207], [242]
Input	Test hook TH_01
Output	Check proper CHs & DADs registration into the INTER-IoT network.
Logs	B. Association\T4.60.7_multiple_reg.txt
Outcome	Pass / Fail
	Assoc. delay

3.3.3.32 U61 – Platform Configuration on the Gateway

The whole platform is accessed remotely by the user. The user configures the system, activates the devices and controls the execution, even applying setup changes and/or sending specific requests to selected DADs.

T4.61.1 Platform setup and simulation

ID	T4.61.1
Test	INTER-HARE setup and full simulation
Type	E. Integration network
Setup	Need test setup TS_08
Start	All devices (1 GW, 2 CHs and multiple DADs) are located in a fixed position but are not associated to the network yet.
Req.	[17], [80], [11], [19], [20], [21], [22], [23], [25], [43], [243], [13], [16], [55], [138], [226], [72], [98], [242], [244]
Input	Test hooks TH_01, TH_02, TH_03, TH_05, and TH_06
Output	Check system's ability to configure the parameters selected by the user. Check correct transmission of packets from/to DADs according to the hybrid data delivery model.
Logs	E. Integration\T4.61.1_platform_conf.txt
Outcome	Pass / Fail

3.3.3.33 U62 – Device (sensor) triggers information

A device, typically a sensor, triggers an event sending determined information to the gateway in order to be stored in the platform.

T4.62.1 Event-driven data delivery model test

ID	T4.62.1
Test	Performance analysis of event-driven traffic
Type	C. Data transmission
Setup	Need test setup TS_07
Start	All DADs, CHs and the GW are already registered.
Req.	[2], [6], [9], [153], [232], [233], [11], [20], [21], [25], [26], [56], [57], [204], [205], [206], [72], [27], [28], [242]
Input	Test hooks TH_03 and TH_04
Output	Check performance of different network metrics.
Logs	C. Transmission\T4.62.1_event.txt
Outcome	Pass / Fail PDR (%) Delay Throughput PRM Energy consumption

T4.62.2 Continuous data delivery model test

ID	T4.62.2
Test	Performance analysis of continuous traffic
Type	C. Data transmission
Setup	Need test setup TS_07
Start	All DADs, CHs and the GW are already registered.
Req.	[2], [6], [9], [153], [232], [233], [11], [20], [21], [56], [57], [75], [204], [206], [72], [27], [28], [95], [242]
Input	Test hooks TH_02 and TH_04

Output	Check performance of different network metrics.
Logs	C. Transmission\T4.62.2_continuous.txt
Outcome	Pass / Fail PDR (%) Delay Throughput PRM Energy consumption

T4.62.3 Hybrid data delivery model test

ID	**T4.62.3**
Test	Performance analysis of hybrid traffic
Type	C. Data transmission
Setup	Need test setup TS_07
Start	All DADs, CHs and the GW are already registered.
Req.	[2], [6], [9], [15], [80], [153], [232], [233], [11], [20], [21], [25], [26], [89], [56], [93], [57], [75], [204], [205], [206], [207], [72], [27], [28], [95], [242]
Input	Test hooks TH_02, TH_03, and TH_04
Output	Check performance of different network metrics.
Logs	C. Transmission\T4.62.3_hybrid.txt
Outcome	Pass / Fail PDR (%) Delay Throughput PRM Energy consumption

T4.62.4 Data aggregation test

ID	**T4.62.4**
Test	Performance analysis of data aggregation mechanism

Type	C. Data transmission
Setup	Need test setup TS_06
Start	All DADs, CHs and the GW are already registered.
Req.	[2], [6], [9], [153], [232], [233], [11], [20], [21], [25], [26], [89], [56], [93], [57], [75], [204], [205], [206], [207], [72], [27], [28], [95], [242]
Input	Test hooks TH_02, TH_03, and TH_04
Output	Check performance of different network metrics.
Logs	C. Transmission\T4.62.4_aggregation.txt
Outcome	Pass / Fail PDR (%) Delay Throughput PRM Energy consumption

T4.62.5 Relay operation test

ID	**T4.62.5**
Test	Performance analysis of relay device & mechanism
Type	C. Data transmission
Setup	Need test setup TS_05
Start	All DADs, CHs and the GW are already registered. The Relay is activated.
Req.	[2], [6], [9], [153], [232], [233], [11], [20], [21], [25], [26], [89], [56], [93], [57], [75], [204], [205], [206], [207], [72], [27], [28], [95], [242]
Input	Test hooks TH_02, TH_03, and TH_04
Output	Check performance of different network metrics.
Logs	C. Transmission\T4.62.5_relay.txt

Outcome	Pass / Fail PDR (%) Delay Throughput PRM Energy consumption

3.3.3.34 U63 – Platform requests information from a device (sensor)

The user asks for data from a specific DAD.

T4.63.1 Query-driven data delivery model test (requests)

ID	T4.63.1
Test	Performance analysis of query driven traffic (requests)
Type	C. Data transmission
Setup	Need test setup TS_07
Start	All DADs, CHs and the GW are already registered.
Req.	[2], [6], [9], [80], [153], [232], [233], [11], [20], [21], [25], [26], [56], [57], [204], [206], [72], [27], [28], [242]
Input	Test hooks TH_03 and TH_04
Output	Check performance of different network metrics.
Logs	C. Transmission\T4.63.1_query_requests.txt
Outcome	Pass / Fail PDR (%) Delay Throughput PRM Energy consumption

3.3.3.35 U64 – Platform sends information to device (actuator)

Once the DAD receives the data request, it transmits the information through its pre-established path to the GW.

T4.64.1 Query-driven data delivery model test (responses)

ID	T4.64.1
Test	Performance analysis of query driven traffic (responses)
Type	C. Data transmission
Setup	Need test setup TS_07
Start	All DADs, CHs and the GW are already registered.
Req.	[2], [6], [9], [80], [153], [232], [233], [11], [20], [21], [25], [26], [56], [57], [204], [206], [72], [27], [28], [242]
Input	Test hooks TH_03 and TH_04
Output	Check performance of different network metrics.
Logs	C. Transmission\T4.64.1_query_responses.txt
Outcome	Pass / Fail PDR (%) Delay Throughput PRM Energy consumption

3.3.3.36 Test outcome overview

Test outcome

The following table will provide an overview of the test result of all the performed tests in this FAT.

Concept	Test	Description	Outcome	Value
A. Range coverage	T4.60.1	Range coverage at 868 MHz	Max. distance	
	T4.60.2	Range coverage at 2.4 GHz	Max. distance	
	T4.60.3	Interference analysis at 2.4 GHz	Interference map	
B. Association & Registration	T4.60.4	(Physical) Gateway registration	Pass / Fail	
			Assoc. delay	
	T4.60.5	Container registration	Pass / Fail	
			Assoc. delay	

	T4.60.6	Sensor registration	Pass / Fail	
			Assoc. delay	
	T4.60.7	Multiple sensor registration	Pass / Fail	
			Assoc. delay	
C. Data Transmission	T4.63.1	Query-driven data delivery model test (requests)	Pass / Fail	
			PDR (%)	
			Delay	
			Throughput	
			PRM	
			Energy consumption	
	T4.64.1	Query-driven data delivery model test (responses)	Pass / Fail	
			PDR (%)	
			Delay	
			Throughput	
			PRM	
			Energy consumption	
	T4.62.1	Event-driven data delivery model test	Pass / Fail	
			PDR (%)	
			Delay	
			Throughput	
			PRM	
			Energy consumption	
	T4.62.2	Continuous data delivery model test	Pass / Fail	
			PDR (%)	
			Delay	
			Throughput	
			PRM	
			Energy consumption	
	T4.62.3	Hybrid data delivery model test	Pass / Fail	

			PDR (%)	
			Delay	
			Throughput	
			PRM	
			Energy consumption	
	T4.62.4	Data aggregation test	Pass / Fail	
			PDR (%)	
			Delay	
			Throughput	
			PRM	
			Energy consumption	
	T4.62.5	Relay operation test	Pass / Fail	
			PDR (%)	
			Delay	
			Throughput	
			PRM	
			Energy consumption	
D. Resilience	T4.46.1	Resilience against failures	Pass / Fail	
E. Integration network	T4.19.1	User interaction	Pass / Fail	
	T4.61.1	Platform setup and simulation	Pass / Fail	
	FAT Outcome		**Pass / Fail**	

Table 67: Test outcome overview

3.3.3.37 Integration ethics and security

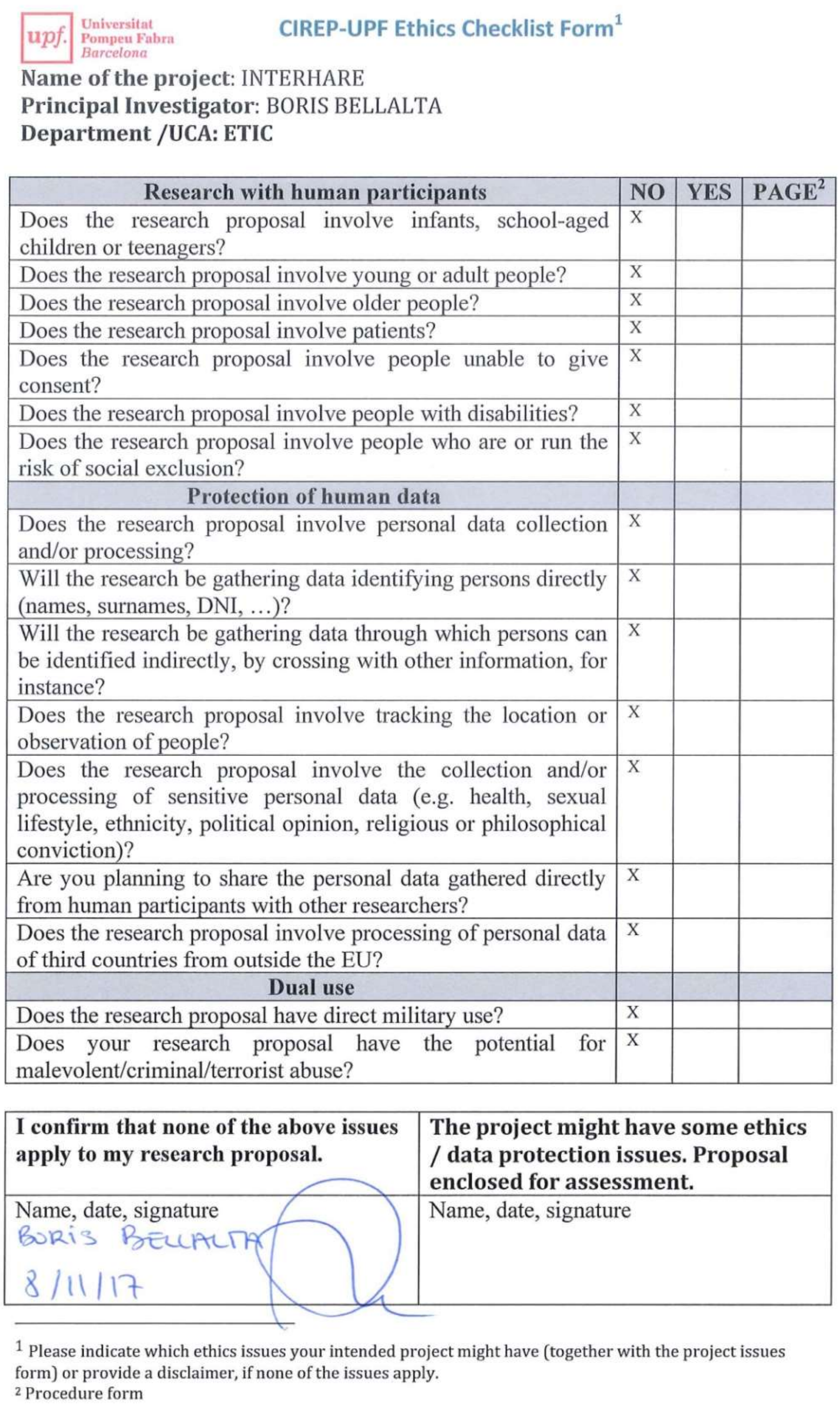
Our research group, as a member of the ***Universitat Pompeu Fabra***, has established guidelines regarding ethics in all its projects. The university created the Internal Committee of Projects (CIREP-UPF) in December 2014. The committee wants to improve the evaluation of the ethical standards and personal data protection in research activities and academics practices.

Every research project performed by our group is previously analyzed by the members of our group following the guidelines of CIREP. The procedure is the following:

1. A self-assessment is performed by the principal investigators by filling an Ethics Checklist Form. If a YES answer is checked, the IP must send the form and proceed to step 2.
2. The IP has to prepare the following documents for the ethics committee: a summary information form that includes the title of the project and a summary of the research activity (including all the activities that involve any personal data or humans participating in them); a procedure form, that includes a detailed description of the methodology that the project will use in order to be evaluated by the CIREP; and, finally, an informed consent form that will be used during the research activities.
3. An external evaluation by a reviewer in the specific research field is performed. A peer review will be submitted by him/her to CIREP members.
4. Finally, the CIREP experts meet and discuss the researcher's documents and the peer review, to issue an Ethics Review Report to the researcher.

Our research project has been analysed following the Ethics checklist form, and, as it neither contemplates the participation of humans nor uses any human data, the ethics of our project has been evaluated and the basics requirements are fulfilled.

On the other hand, the use of confidential data along the pilot is not either contemplated. Hence, the security is not compromised. The user IDs in our network are codified, so that any external intruder with non-authorized access to the sensor network will not have any knowledge on the data due to the codification of the elements belonging to the network.



Universitat Pompeu Fabra Barcelona

CIREP-UPF Ethics Checklist Form[1]

Name of the project: INTERHARE
Principal Investigator: BORIS BELLALTA
Department /UCA: ETIC

Research with human participants	NO	YES	PAGE[2]
Does the research proposal involve infants, school-aged children or teenagers?	X		
Does the research proposal involve young or adult people?	X		
Does the research proposal involve older people?	X		
Does the research proposal involve patients?	X		
Does the research proposal involve people unable to give consent?	X		
Does the research proposal involve people with disabilities?	X		
Does the research proposal involve people who are or run the risk of social exclusion?	X		
Protection of human data			
Does the research proposal involve personal data collection and/or processing?	X		
Will the research be gathering data identifying persons directly (names, surnames, DNI, …)?	X		
Will the research be gathering data through which persons can be identified indirectly, by crossing with other information, for instance?	X		
Does the research proposal involve tracking the location or observation of people?	X		
Does the research proposal involve the collection and/or processing of sensitive personal data (e.g. health, sexual lifestyle, ethnicity, political opinion, religious or philosophical conviction)?	X		
Are you planning to share the personal data gathered directly from human participants with other researchers?	X		
Does the research proposal involve processing of personal data of third countries from outside the EU?	X		
Dual use			
Does the research proposal have direct military use?	X		
Does your research proposal have the potential for malevolent/criminal/terrorist abuse?	X		

I confirm that none of the above issues apply to my research proposal.	**The project might have some ethics / data protection issues. Proposal enclosed for assessment.**
Name, date, signature BORIS BELLALTA 8/11/17 [illegible]	Name, date, signature

[1] Please indicate which ethics issues your intended project might have (together with the project issues form) or provide a disclaimer, if none of the issues apply.
[2] Procedure form

3.3.3.38 Annex A: Zolertia RE-Mote

Re-Mote is a powerful development board to build real IoT projects and solutions. It can be considered as an Ultra-Low Power Wireless platform for 2.4 GHz and 863-950 MHz IEEE 802.15.4, 6LoWPAN, and ZigBee Applications.

Re-Mote IoT hardware board was developed jointly with universities and industrial partners from different countries in the context of a European Project to create powerful IoT hardware for Smart cities, logistics, lighting and industrial project. Fully compatible with main IoT operation systems, RE-Mote is the perfect hardware platform to create and work in real IoT.

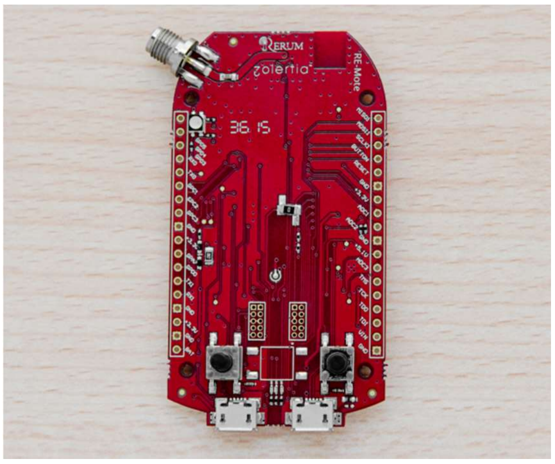


Figure 85: Zolertia RE-Mote front view

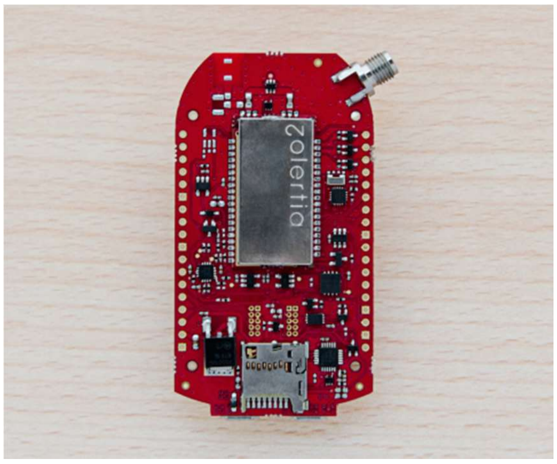


Figure 86: Zolertia RE-Mote back view

- **Microcontroller**
  - Powerful ARM® Cortex® -M3
  - 32-MHz Clock Speed
  - 512KB In-SystemProgrammable Flash
  - 32KB of RAM (16KB With Retention)
  - USB 2.0 Full-Speed Device (12 Mbps)
  - Security Hardware acceleration (AES-128/256, SHA2, ECC-128/256, RSA Hardware Acceleration Engine for Secure Key Exchange)
  - 1 x I2C, 1 x SPI, 1 x UART, µDMA, 12-Bit ADC with configurable resolution
  - General Purpose Pin (GPIO) 4/20mA
- **Features**
  - Micro-SD support
  - Shutdown Mode (190nA)
  - Real Time Clock Calendar (RTCC)
  - External Watchdog Timer and battery monitor (optional)
  - Built-in LiPo Battery Charger
  - RF switch to programatically drive either 2.4 GHz or Sub-1GHz RF interface to RP-SMA connector
- **RF 2.4 GHz**
  - ISM 2.4 GHz IEEE 802.15.4 & Zigbee/Thread compliant
  - 2394-2507 MHz frequency range with 1MHz/5MHz programable steps
  - Sensitivity -97 dBm, ACR 44 dB, up to 7 dBm transmission power
  - Data Rate 250 Kbps with DSSS Modulation

- **RF Sub-1 GHz**
  - ISM 863-868, 915-, 920-, 950 MHz ISM/SRD Band, IEEE 802.15.4g compliant
  - Sensitivity -109 dBm (50Kbps), down to -123 dBm (1.2 Kbps), ACR up to 60 dB, up to +16 dBm transmission power
  - Channels from 12.5 KHz - 1.66 MHz
  - Supported modulations 2-FSK, 2-GFSK, 4-FSK, 4-GFSK, MSK, OOK
  - Supports Data Rate Up to 1.25 Mbps in Transmit and Receive Modes

- **Operational values**

Parameter	Minimum	Average	Maximum	Unit
Operation supply voltage	3.4	4.7	16	V
General purpose pin voltage output	0	3.3	-	V
General purpose pin current output	0	4	20	mA
Shutdown mode	150	-	-	nA
PM3 power mode	-	0.4	-	μA
PM1 power mode	-	0.6	-	mA
Active mode (2.4 GHz RX, CPU idle)	-	20	-	mA
Active mode (2.4 GHz TX, 0 dBm, CPU idle)	-	24	-	mA
Operation Temperature	-40	25	100	ºC

Table 68: Zolertia RE-Mote operational values

- **Pin-out distribution and components**
  - Front view

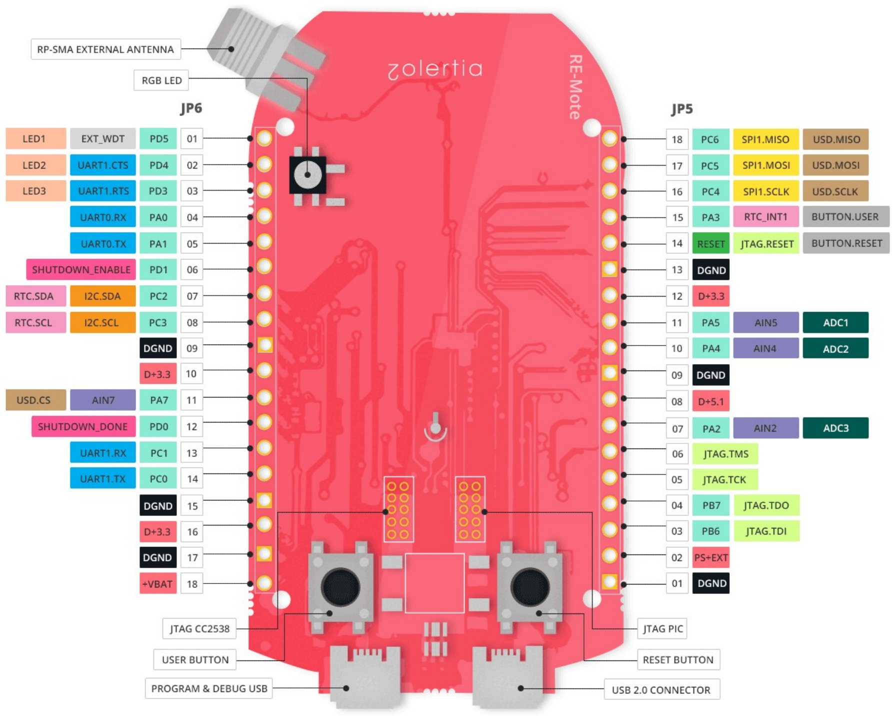


Figure 87: Zolertia RE-Mote schematic front view

- Back view

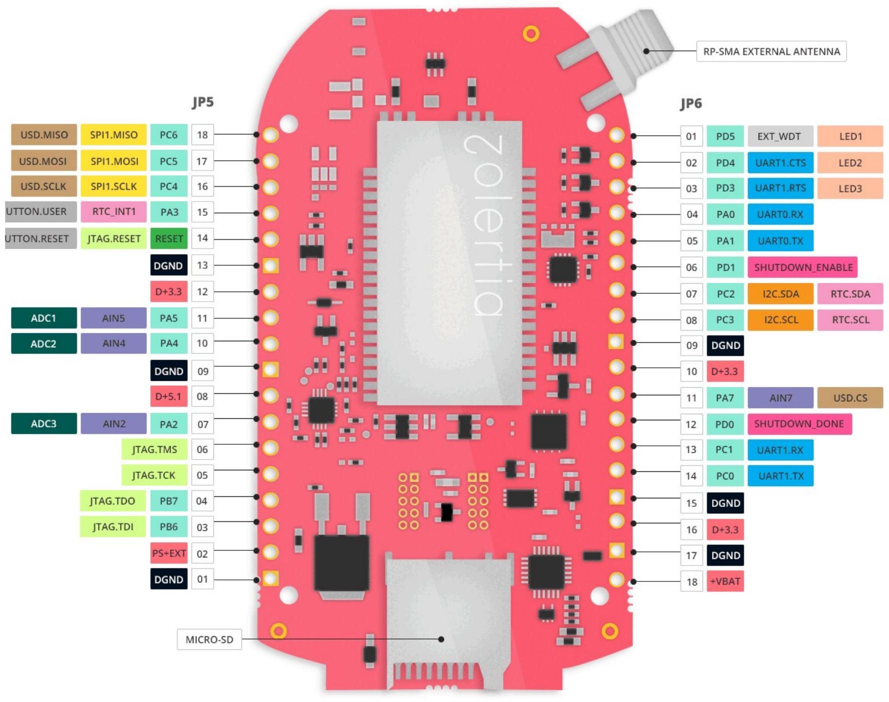


Figure 88: Zolertia RE-Mote schematic back view

- **Compliances**
  - Europe: ETSI EN 300 220, ETSI EN 54-25, EN 300 328, EN 300 440.

- US: FCC CFR47 Part 15, FCC CFR47 Part 90, 24, 101.
- Japan: ARIB RCR STD-T30, ARIB STD-T66, ARIB STD-T67, ARIB STD-T108

- **Development tools**
  - Contiki OS: contiki-os.org
  - RIOT OS: www.riot-os.org
  - Thingsquare Platform
  - OpenWSN (upcoming)
  - Eclipse IDE for C/C++
  - Code Composer Studio™
  - IAR Embedded Workbench ® for ARM
  - SmartRF™ Studio
  - SmartRF Flash Programmer
  - Built-in programming support over USB
- **Zolertia resources**
  - Zolertia site: www.zolertia.io
  - Zolertia Online Store: zolertia.io/store
  - Resource Page: www.zolertia.io/resources
  - Zolertia Github repository: github.com/Zolertia
  - Hackster Page: www.hackster.io/zolertia

3.3.3.39 Annex B: Zolertia Orion Router

The Orion Router is a capable IPv4/IPv6 routing device, with an Ethernet interface and dual wireless radio, powered either via micro-USB or Power Over Ethernet (POE). The device integrates the ENC28J60 Ethernet module, and an external POE module, supporting up to 48VDC.

The Zolertia Ethernet Router exposes a Power DPDT push button, a RESET button to reboot the CC2538 system-on-chip, and a programmable USR button. The RP-SMA connector for a 2.4GHz external antenna is located next to the RJ Ethernet connector, whereas the SMA connector for sub-GHz antenna is located on the opposite side. The micro-USB connector is used for both powering the device over USB (5VDC), and programming/debugging.

The female RJ-45 Ethernet support active Power Over Ethernet (POE) 802.3af, up to 48VDC. Using POE allows to carry both data and power over the same cabling, reducing the number of elements required to connect and power the Zolertia Ethernet Router.

A non-populated JTAG connector is also available to further debug the device using an external JTAG tool. Two on-board LEDs are available over a light-guide. The green LED next to the Power switch shows the device power status (lit when on, else off).

- **Features**
  - ARM Cortex-M3 with 512KB flash and 32KB RAM (16KB retention), 32MHz
  - ISM 2.4-GHz IEEE 802.15.4 & Zigbee compliant
  - ISM 868-, 915-, 920-, 950-MHz ISM/SRD Band
  - RP-SMA connector for a 2.4GHz external antenna
  - SMA connector for a 868/915MHz external antenna
  - RJ45 ethernet connector
  - Ethernet 10BASE-T IPv4/IP64
  - AES-128/256, SHA2 Hardware Encryption Engine
  - ECC-128/256, RSA Hardware Acceleration Engine for Secure Key Exchange

- On-board CP2104/PIC to flash over its micro-USB connector
- User and reset buttons
- Power on/off button and LED to show power state
- RGB LED to allow more than 7 colour combinations
- Indoor enclosure
- Layout 40.29 x 73.75 mm

- **Mezzanine**
  - Two radios to use both residential/indoor and long-range applications. The maximum range is between 100 meters and 20 km, with highly configurable radio parameters such as modulation, data rate, transmission power, etc.
  - Two radios (short and long range), compatible with existing and trending protocols such asThread, but you can also develop your own applications on top of very well supported protocols like 6LoWPAN and IEEE 802.15.4, without vendor restrictions or licenses.
  - Increased security with on-board hardware security (SHA2, AES-128/256, ECC-128/256 and RSA for secure key exchange).
  - Out of the box connectivity with IPv6 and IPv4 networks and services, using IP64 (NAT64, DHCP64).
  - Power the device using Active POE (up to 48VDC, using auto-negotiation), or over its micro-USB connector with any regular USB charger.
  - Slick indoor enclosure.
  - Compatibility with 6lbr.

- **Pin-out distribution and components**

  There are additional connectors (not populated as default) with 2.54mm pitch spacing, exposing the CC2538 pins. The JP4 and JP6 connectors exposes pins mostly to interface sensors, actuators and communication buses. The JP6 even-numbered pins are compatible with the I2C default pins used by other platforms such as the Firefly or the RE-Mote).

  The JP5 exposes the pins of the CC2538 connected to the CC1200 sub-GHz transceiver, and the ENC28J60 Ethernet module.

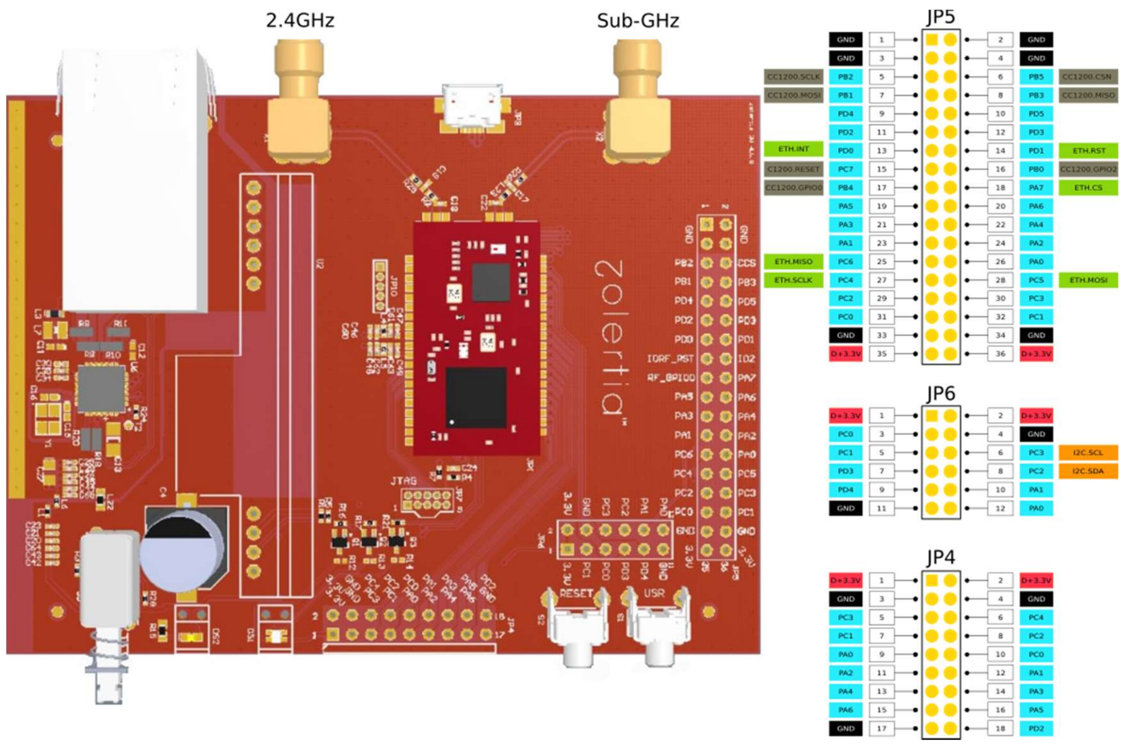


Figure 89: Zolertia Orion Router schematic view

3.3.3.40 Annex C: BeagleBone Black

The Beaglebone Black is a small computer that will control the integration network. With a size similar to a credit card, it can be used as a PC, connecting it to a keyboard, monitor and mouse, viewing video using the HDMI connector. In addition, it has Internet connection and has 92 GPIO (General Purpose Input / Output) pins to interact with sensors, buttons or any other electronic circuit.

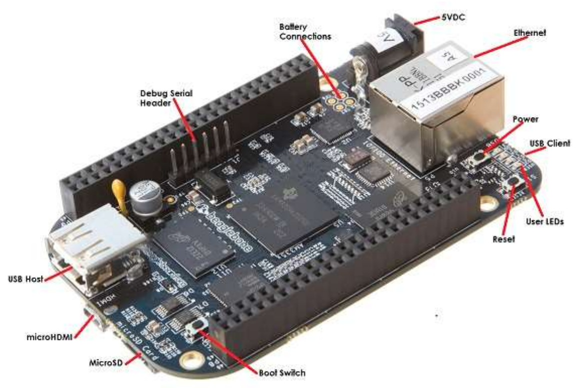


Figure 90: BeagleBone Black front view

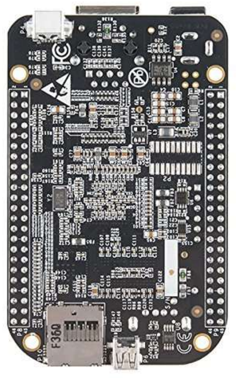
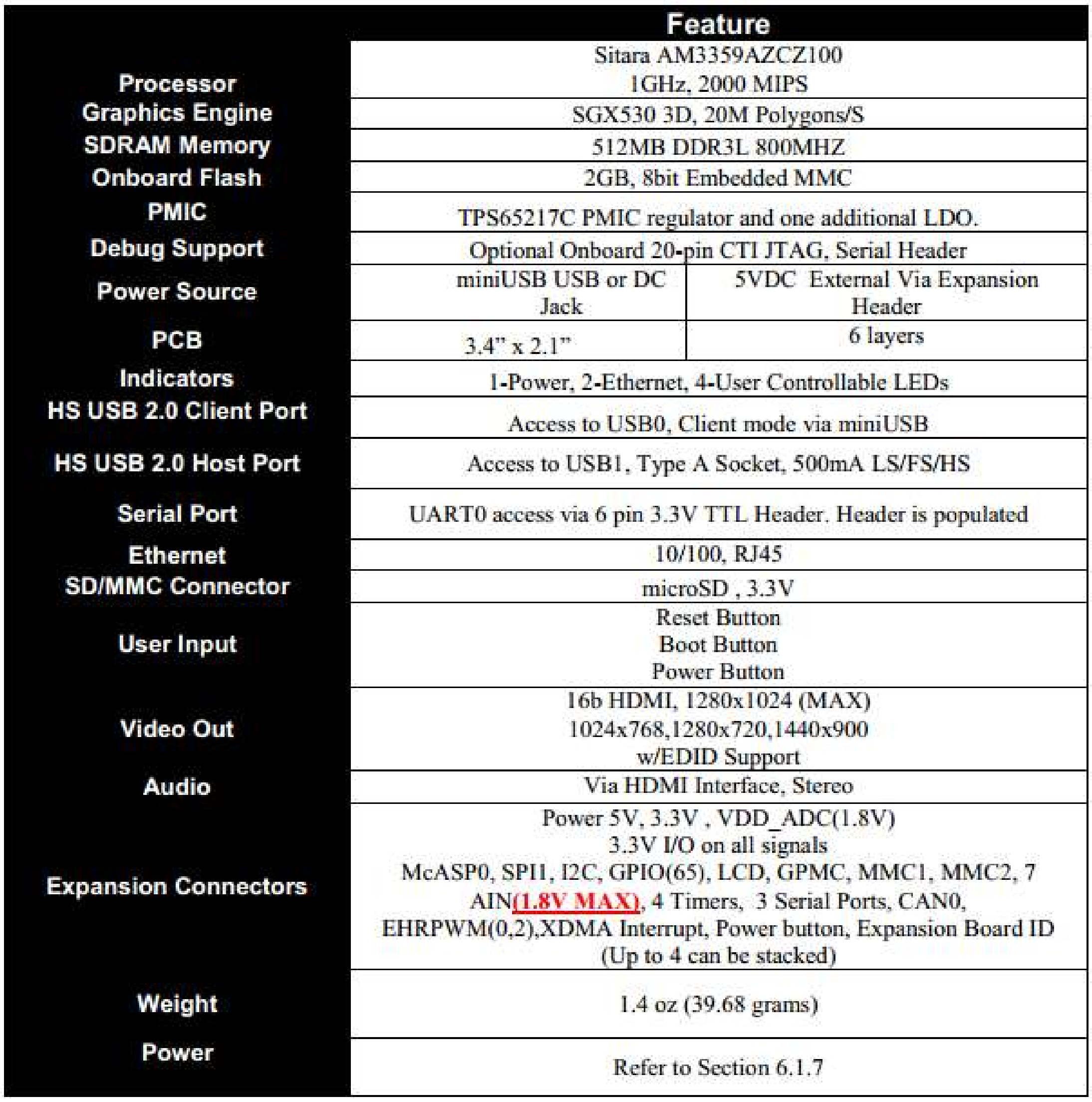


Figure 91: BeagleBone Black rear view

The microcomputer works on the basis of an ARM processor of the Sitara class of Texas Instrument. It has a 2-core CPU, Cortex A8, and runs 1GHz instructions with its 512MB of DDR3 RAM and a 3D graphics processor. It offers a number of programming ports similar to the Arduino and additionally has more than 30 "Capes" that expand its capabilities and allow connection to 3D printers, lighting controllers, Geiger counters, telerobotic submarines or LCD touch screens, among other applications.

It has a 10/100 Ethernet port, 1 USB host port and 1 OTG USB, 2GB of built-in memory and space for a MicroSD. The default operating system is Ångström Linux, but it is possible to install Android or Ubuntu. It includes preinstalled Cloud9 IDE (an open source development environment integrated in the cloud and web-based, which supports several programming languages). In addition, Beaglebone Black includes two 46-pin ports into which expansion modules can be inserted.

A detailed description of technical specifications from the BeagleBone Black Rev. A5C used in the current project is offered in Figure 92.



	Feature	
Processor	Sitara AM3359AZCZ100 1GHz, 2000 MIPS	
Graphics Engine	SGX530 3D, 20M Polygons/S	
SDRAM Memory	512MB DDR3L 800MHZ	
Onboard Flash	2GB, 8bit Embedded MMC	
PMIC	TPS65217C PMIC regulator and one additional LDO.	
Debug Support	Optional Onboard 20-pin CTI JTAG, Serial Header	
Power Source	miniUSB USB or DC Jack	5VDC External Via Expansion Header
PCB	3.4" x 2.1"	6 layers
Indicators	1-Power, 2-Ethernet, 4-User Controllable LEDs	
HS USB 2.0 Client Port	Access to USB0, Client mode via miniUSB	
HS USB 2.0 Host Port	Access to USB1, Type A Socket, 500mA LS/FS/HS	
Serial Port	UART0 access via 6 pin 3.3V TTL Header. Header is populated	
Ethernet	10/100, RJ45	
SD/MMC Connector	microSD , 3.3V	
User Input	Reset Button Boot Button Power Button	
Video Out	16b HDMI, 1280x1024 (MAX) 1024x768,1280x720,1440x900 w/EDID Support	
Audio	Via HDMI Interface, Stereo	
Expansion Connectors	Power 5V, 3.3V , VDD_ADC(1.8V) 3.3V I/O on all signals McASP0, SPI1, I2C, GPIO(65), LCD, GPMC, MMC1, MMC2, 7 AIN(1.8V MAX), 4 Timers, 3 Serial Ports, CAN0, EHRPWM(0,2),XDMA Interrupt, Power button, Expansion Board ID (Up to 4 can be stacked)	
Weight	1.4 oz (39.68 grams)	
Power	Refer to Section 6.1.7	

Figure 92: BeagleBone Black Rev. A5C technical specifications

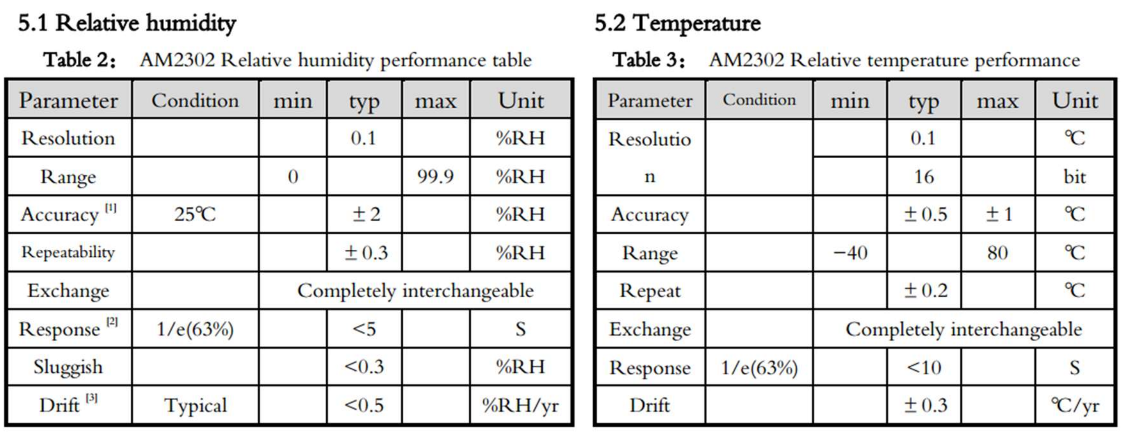
3.3.3.41 Annex D: DHT22 temperature and humidity sensor

The DHT22 is a basic, low-cost temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and spits out a signal on the data pin. It is fairly simple to use, but requires careful timing to grab data. The only real downside of this sensor is that new data can only be got from it once every 2 seconds; however, this is not a drawback for the INTER-HARE platform, as time between two consecutives measures will be always much higher.

Figure 93: DHT22 temperature and humidity sensor

From the datasheet of the DHT22 temperature and humidity sensor[23], we can extract its main features, which are summarized in the following lines (for more detailed information, see Figure 94 and Figure 95):

- Supply Voltage: 3.3 - 5.5V
- Temperature Range: -40 - 80 °C / resolution 0.1 °C / error < ±0.5 °C
- Humidity Range: 0 - 100% RH / resolution 0.1% RH / error ±2% RH
- Wiring map: VCC, GND, S
- Size: 38 x 20mm (1.50x0.79")



5.1 Relative humidity

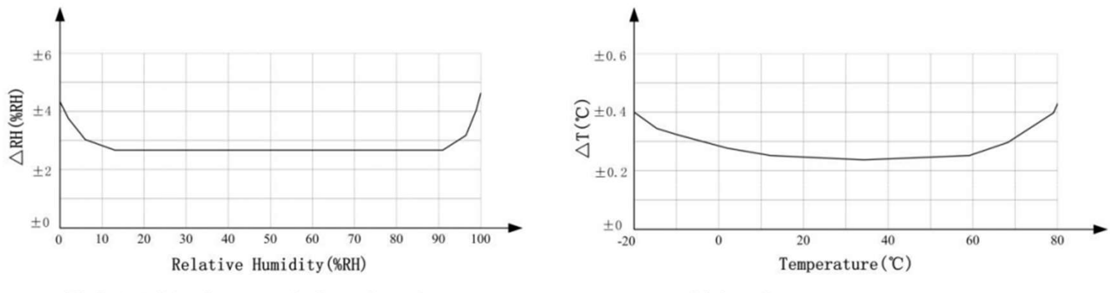
Table 2: AM2302 Relative humidity performance table

Parameter	Condition	min	typ	max	Unit
Resolution			0.1		%RH
Range		0		99.9	%RH
Accuracy [1]	25℃		±2		%RH
Repeatability			±0.3		%RH
Exchange		Completely interchangeable			
Response [2]	1/e(63%)		<5		S
Sluggish			<0.3		%RH
Drift [3]	Typical		<0.5		%RH/yr

5.2 Temperature

Table 3: AM2302 Relative temperature performance

Parameter	Condition	min	typ	max	Unit
Resolutio			0.1		℃
n			16		bit
Accuracy			±0.5	±1	℃
Range		−40		80	℃
Repeat			±0.2		℃
Exchange		Completely interchangeable			
Response	1/e(63%)		<10		S
Drift			±0.3		℃/yr



Pic2: At25℃ The error of relative humidity

Pic3: The maximum temperature error

Figure 94: Main features of the DHT22 temperature and humidity sensor

[23] AOSONG, "Temperature and humidity module DHT22/AM2302 Product Manual," [Online]. Available: http://akizukidenshi.com/download/ds/aosong/AM2302.pdf.

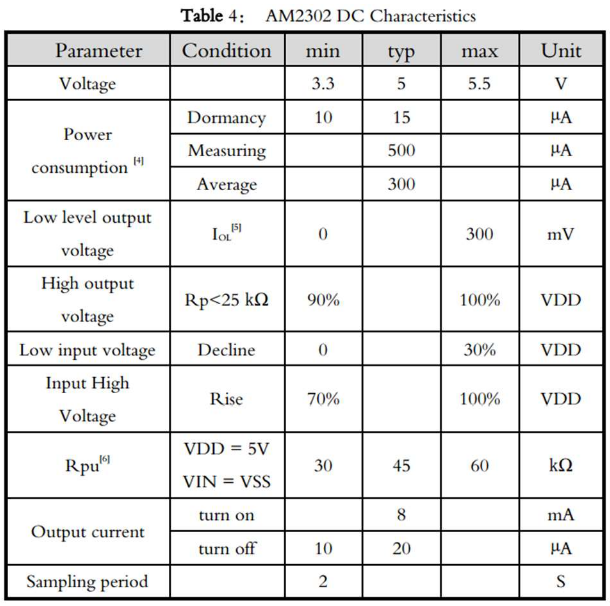


Table 4: AM2302 DC Characteristics

Parameter	Condition	min	typ	max	Unit
Voltage		3.3	5	5.5	V
Power consumption [4]	Dormancy	10	15		μA
	Measuring		500		μA
	Average		300		μA
Low level output voltage	I_{OL}[5]	0		300	mV
High output voltage	Rp<25 kΩ	90%		100%	VDD
Low input voltage	Decline	0		30%	VDD
Input High Voltage	Rise	70%		100%	VDD
Rpu[6]	VDD = 5V VIN = VSS	30	45	60	kΩ
Output current	turn on		8		mA
	turn off	10	20		μA
Sampling period		2			S

Figure 95: Electrical parameters of the DHT22 temperature and humidity sensor

The RE-Mote has 2 x ADC available ports that can be used with the Molex 3-pin WM4901-ND male header connector, providing the normally used GND and VCC pins to connect analogue sensors. Depending on the sensor power operation requirement you can use the ADC1 (3.3V) or the ADC3 (5V). The pins are 2.54 mm spaced and the connector has the following pin-out:

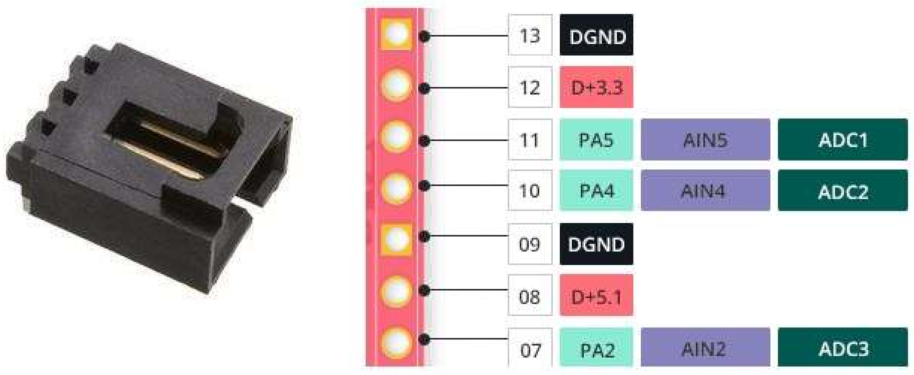


Figure 96: Zolertia Re-mote's analog connector pin-out

In this case, the Zolertia RE-Mote uses one of the two ADC connectors to communicate with the DHT22 temperature and humidity sensor breakout. ADC communication libraries included in Contiki OS porting of the Zolertia RE-Mote have been used for this purpose.

3.3.4 Third Party: Mission Critical operations based on IoT analytics

The "Mission Critical operations based on IoT analytics" (MiCrOBIoTa) project, aims at exploiting the INTER-IoT platform as a means to gather information from heterogeneous sensors in a converged way. As a result, Nemergent will integrate a new "IoT monitoring and analytics" component in its mission critical product portfolio, and especially into the Nemergent Control Room application. Figure 97 illustrates the overall Nemergent mission critical applications framework and the specific scope of the work proposed in MiCrOBIoTa.

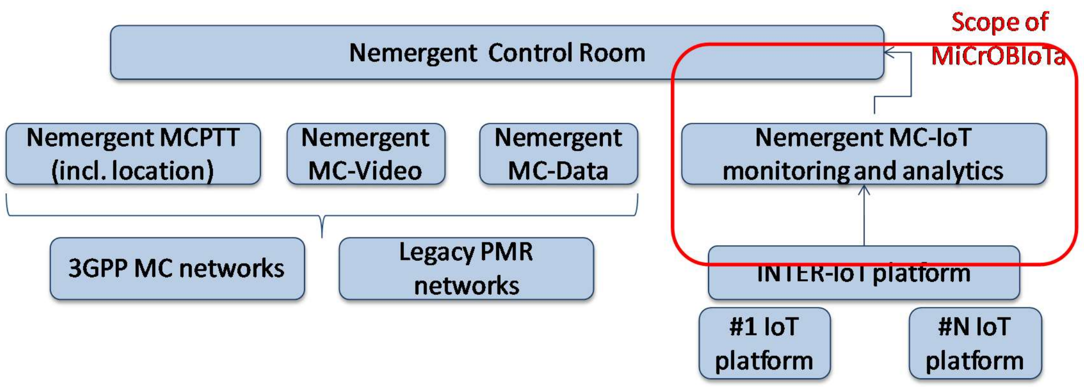


Figure 97: Scope of MiCrOBIoTa activities.

Figure 98 depicts an overall description of the main system to be used, tested and integrated in the scope of the INTER-IoT project.

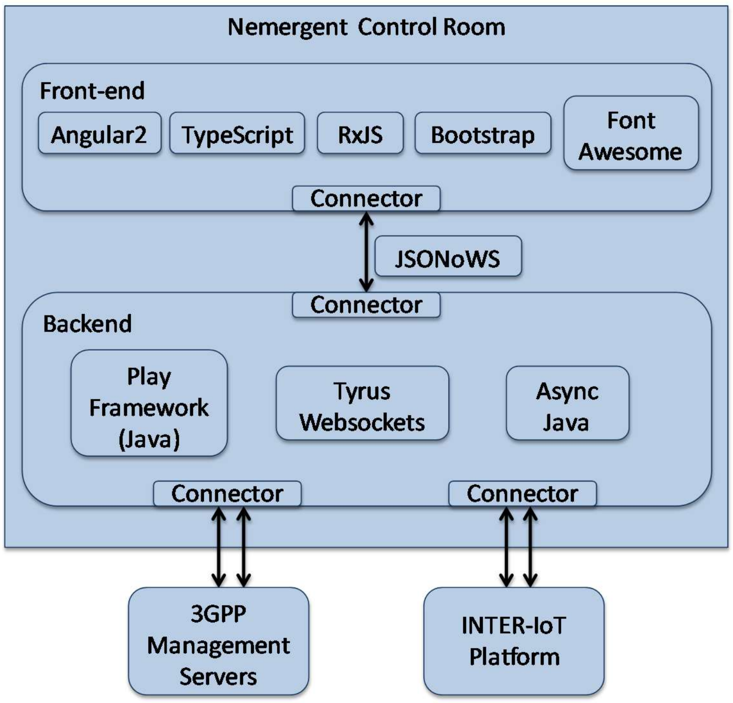


Figure 98: Overall system description.

The Nemergent Control Room system is made up of two main components.

The Nemergent Frontend component is an extensible HTML5-capable control interface that communicates through a Java backend with a plethora of different services and data providers. This component is built upon a series of state-of-the-art web technologies. The protocol used to sync real-time data between the Java backend and the HTML frontend is defined as a series of JSON messages encapsulated through a WebSocket tunnel, in order to provide bidirectional real-time flows between the final user and the rest of the system.

The Nemergent Back-end is a Java-based component that implements most of the business logic related to data gathering and the specific interfaces to the different underlying systems. Similarly, to the front-end, the back-end component uses several modern technologies and implements specific connectors to each underlying communications system.

In the scope of INTER-IoT, a new connector is being developed to cover the communication with the INTER-IoT platform. This connector will implement all the necessary calls to the INTER-IoT API in order to gather the selected information with the corresponding message formats and data types.

The anticipated benefits of interoperable “Mission Critical operations based on IoT analytics” are unquestionable. A typical situation in mission critical operations support systems is to include information coming from specifically deployed devices to gather environmental measurements. Examples of these devices are temperature sensors, meteorological and hydrological probes, traffic monitoring cameras, etc. We propose to add the Mission Critical IoT (MC-IoT) system, which includes a new monitoring and analytics component and an evolved Control Room interface tailored to the specific needs of the use case. In the case of a simulated crisis, significant information from on-body health-related sensors and port logistics devices will provide life-saving information to the mission critical operations support system. Besides, the available mission critical communications components can be used to demonstrate the crisis handling use case.

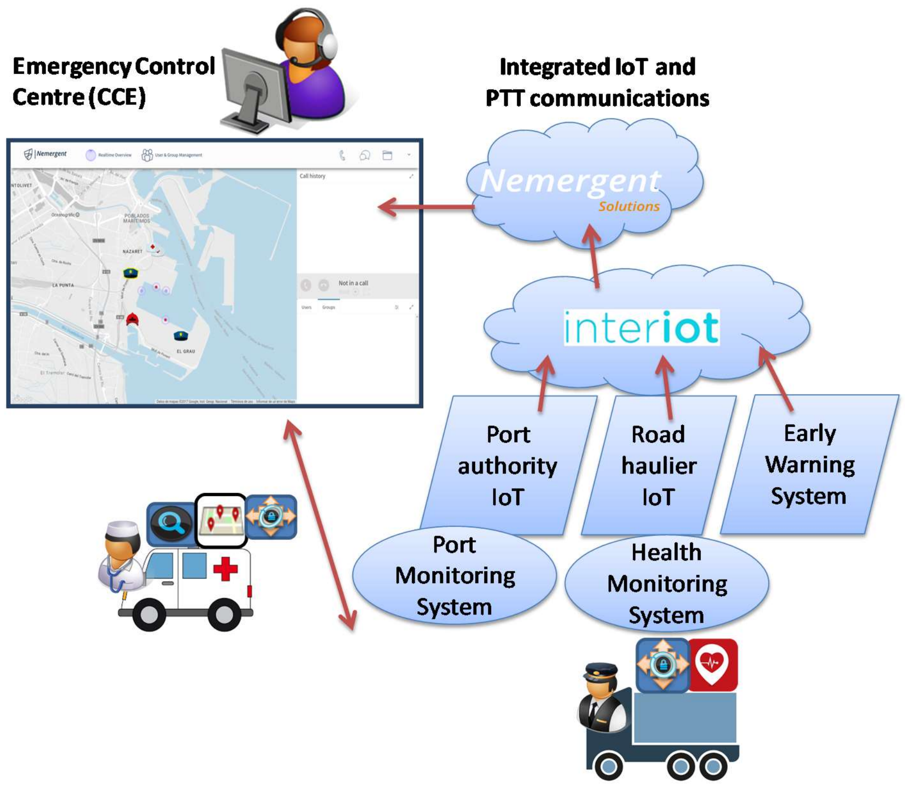


Figure 99. IoT-aided Mission Critical operations scenario.

Taking into account the overall picture and the availability of different IoT platforms, a complex use case could be created for an emergency simulation exercise. This scenario would include a typical emergency intervention, enhancing the operations support through the use of new communication technologies over commercial networks.

An example operational procedure is provided hereafter:

1. A road haulier comes into the port area. Upon an incident / health issue, the on-board health monitoring sensor reports the anomalous data to the INTER-IoT system through the road haulier company IoT platform.
2. The relevant data arrives at the Port Authority emergency control centre (CCE), which manages incidents taking place within the port and coordinates with other first responders (police, firefighters, ambulances, etc.).
3. The CCE operator accesses the MC-IoT system through the web-based Graphical User Interface (GUI), which will provide different types of icons for the different sources of information, and different views targeted at different emergency response units.
4. The CCE operator can use this platform to communicate with field response units (e.g., ambulance driver), providing them not only with location and navigation support but also with specific context information useful for the intervention.
5. Besides the IoT-related data processing, the extended use case will make use of the Nemergent Mission-Critical Push-To-Talk (MCPTT) communication systems in order to resemble real-time communication between the different entities involved.

3.3.4.1 Integration of IoT framework

From a high level architectural standpoint, the integration of the MC-IoT external application is depicted hereafter.

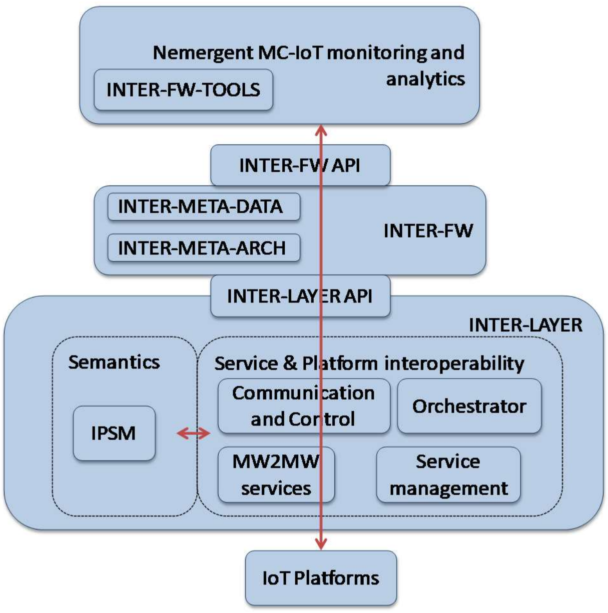


Figure 100. High-level perspective of the integration.

The MC-IoT system will run as an external application to the INTER-IoT platform. In order to access the heterogeneous data from different IoT platforms, the MC-IoT application needs to interact with the INTER-LAYER "Platform interoperability" functional component, which involves the "Communication and Control" and "MW2MW services" INTER-LAYER components. This MC-IoT component will also need to interaction with the "Semantics" functional component, which is implemented through the INTER-LAYER IPSM module.

Additionally, the MC-IoT external application may need the use of composite IoT-related services. Thus, the invoking of the “Service interoperability” functional component may be needed. This would require the involvement of the INTER-LAYER “Orchestrator” and “Service management” components.

In order to interoperate with the INTER-IoT platform (e.g., registration, authorisation, etc.) and being able to invoke the API functions, the MC-IoT external application will make use of the INTER-FW tools and API. To some extent, the INTER-FW API acts as a wrapper of the INTER-LAYER API, exposing only those methods available to the external applications.

From a use case perspective, the external application interfaces with the INTER-IoT platforms in order to gain access to two / potentially three types of information:

1. Access to the port monitoring information (WSO2 platform).
2. Access to the health monitoring information installed on a haulier company (data coming from MyDriving and universAAL systems and integrated through Microsoft Azure platform).
3. Access to well-formed incident / emergency information, as potentially detected by a third party Early Warning System and submitted to the INTER-IoT platform.

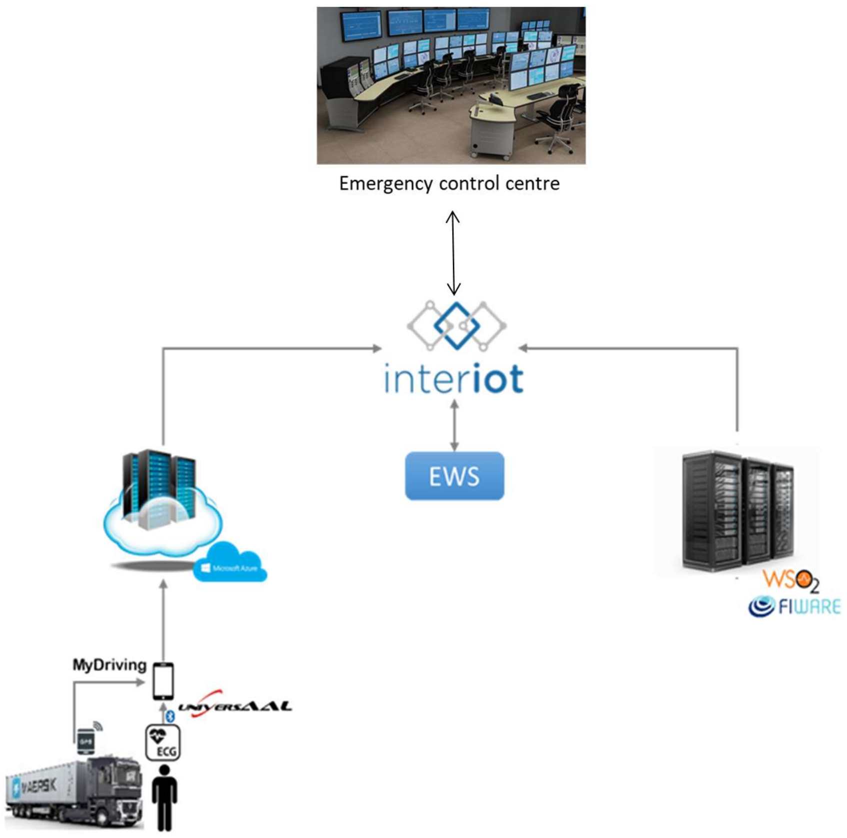


Figure 101. Use case perspective of the integration.

Following the proposed pilot, the trucks will be monitored once they are in the port facilities. In case an accident or a medical problem is detected, the system will publish a notification to the port authority in a standard format (EDXL). Once the emergency control centre receives the notification, it can start communication with the driver with a push to talk protocol in the driver’s mobile.

The main benefits we can get from this scenario are: apply in the port communications a standard format in accident reporting like EDXL, real time identification of the location of the accident, direct communication with the closest control centre when an accident occurs and monitoring driver's health if it is necessary.

3.3.4.2 Deliverables and version overview

The following table contains a deliverable list which need to be signed of before FAT testing commences.

ID	Description	Check
Documents		
1	Validation and Test reports of the IoT system components	
2	Validation and Test reports of the Pilot system components	
Hardware		
3	Gigabyte Barebone on which the Nemergent System is installed	
4	Additional Laptop with basic Internet Browsing Capabilities	
5	Connection cables (Ethernet connection between both systems and Internet)	
Tools		
7	Preferably a recent Linux OS	
8	TestNG Java	
9	Jenkins 2	

Table 69: Deliverable checklist

The following table shows the software components and version of which the system release version 1.0 consists of.

ID	Description	Version	Check
Nemergent CtrlRoom Frontend			
1	JSONoWS API	V1.3.0	
2	WebRTC API	V1.0.0	
3	GUI	V1.0.0	
Nemergent CtrlRoom Backend			
4	JSONoWS API	V1.3.0	
5	MCPTT MS API	V1.0.1	
6	InterIOT API	V1.0.0	

Table 70: Component version overview

3.3.4.3 Requirements, scenarios and use cases

The following table provides an overview of the relation between the requirements and the test(s) that validate their implementation.

ID	Description	Covered by
47	API for third-party developers	T1.1.1, T1.1.2, T1.2.1, T1.2.2, T1.2.3, T1.2.4, T1.3.1, T1.3.2, T1.3.3
51	API for data publication	T1.2.3, T1.2.4, T1.3.1, T1.3.2, T1.3.3
53	Location of sensor and measurement is included in semantic models	T1.2.1, T1.2.2

123	Use of standards	T1.1.1, T1.1.2, T1.2.3, T1.2.4, T1.3.1, T1.3.2, T1.3.3

Table 71: Requirements vs test mapping

The following table provides an overview of the relation between the scenarios and the test(s) that validate their implementation.

ID	Scenario name	Covered by
5	Monitoring of containers carrying sensitive goods	T1.2.1, T1.2.3, T1.3.1
9	Accident at the port area	T1.2.1, T1.2.3, T1.3.1
17	Health monitoring system with passengers aboard a train	T1.2.1, T1.2.3, T1.3.1
32	Third party developer using INTER-FW to access data from two different platforms	T1.1.1, T1.1.2, T1.2.1, T1.2.3, T1.2.4, T1.3.1, T1.3.2, T1.3.3

Table 72: Scenario vs test mapping

3.3.4.4 Test environment

Introduction

To test the functionality of the collaboration module in combination with the IoT framework a representative test system is needed. The test system needs to approach the "real world" as much as possible. The pilot setups must be recreated and proven. This chapter will describe this environment and the used hardware, software, tools and platforms.

Test environment

This paragraph describes the test environment and the complete system setup used during this FAT.

Test setups, tools, hooks and probes

This paragraph describes the test setups, tools, hooks and probes used in the by this document defined tests.

A test setup defines a setup used for a test, and describe the used system parts, interface etc.

Tools define the used COTS tooling that will be used during testing e.g. Logic analyzers, packet sniffer like e.g. Whireshark, developed scripts, etc.

Hooks are used to inject data or control parts of the system used to get the system to handle a needed scenario. This can be hidden, debug or service interfaces which contain the needed functionality.

Probes are used to gather logging from parts of the system under test to provide system feedback and to prove that the system is handling the scenario as it should.

The following paragraphs will define the used setups, tools, hooks and probes which will be referred to by the test descriptions.

TS_01 – Dedicated Docker Image

All of the Nemergent testing activity will be performed inside an ad-hoc developed Docker image, containing all of the neccesary software tools to perform this testing phase.

This image will contain the Jenkins instance, and will have a copy of the software repositories freshly obtained in the moment of the Docker image building. This way, Nemergent can assure maximal compatibility between hardware vendors, software distributions and network environments. This image itself it's prone to be tested inside a continuous integration scheme.

With a command launched in the host Linux system, the integrated Jenkins tool will execute alll the test's sequentially, making detailed reports of the results available through it's web interface, along with performance measurements and precise timings. This Jenkins builds will archive as artifacts all the network recordings, profiling files and log files produced during the test executions.

The Nemergent CtrlRoom software will communicate with the InterIOT components through the virtual network interface offered by Docker to the container, helping with the network debugging issue.

TT_01 - TestNG Java

As a testing environment we will use different suites of TestNG Java testing framework, which will implement the different business logic associated with the working model. For example, a different test suite can be generated for each different EDXL document type, thus assuring maximal code coverage for each module and case.

TT_02 - Jenkins 2

The TestNG Java tests will be orchestrated though a Jenkins 2 server, using the newly released Pipelines functionality, providing expressive description of the test case scenarios.

TH_01 - Jenkins Pipelines

Jenkins Pipelines definitions support parameterized runs, which can be used to inject relevant EDXL messages right into the message broker, depending on target of the tests.

TP_01 – Protractor

Graphical testing can be performed through the implementation of Protractor tests. These tests can assure that frontend graphics being rendered on a virtual screen corresponds to what the requirements require. This would provide similar confidence as a real human visual testing, for example that a specific EDXL document triggers the area and information painting on the Port Control Centre.

3.3.4.5 Test description

Test output log files… Folder “`Tx_Output`”, prefix “`Tx.y.1_`”

T1.1.1 Boot up and first contact

ID	T1.1.1
Test	Nemergent MC-IoT Module boots up and contacts INTER-IOT Platform
Type	System Testing
Setup	Need test setup TS_01
Start	Nemergent CtrlRoom is not launched

Req.	[47], [122]
Input	Launch the jenkins job labeled as "**T1.1.1**"
Output	Check Jenkins job output, wether the job has succeeded or it has failed. Check outputted artifacts for debugging purposes.
Logs	Jenkins Job artifacts section: • backend_log.txt • frontend_log.txt • network_eth0.pcap • network_capture_logs.txt • test_report.html
Outcome	Pass / Fail

T1.1.2 INTER-FW Authentication

ID	**T1.1.2**
Test	Nemergent MC-IoT Module authenticates correctly against INTER-FW gateway.
Type	System Testing
Setup	Need test setup TS_01
Start	Nemergent CtrlRoom is already launched, but module has only made first contact.
Req.	[47], [122]
Input	Launch the jenkins job labeled as "**T1.1.2**"
Output	Check Jenkins job output, wether the job has succeeded or it has failed. Check outputted artifacts for debugging purposes.
Logs	Jenkins Job artifacts section: • backend_log.txt • frontend_log.txt • network_eth0.pcap • network_capture_logs.txt • test_report.html
Outcome	Pass / Fail

T1.2.1 Obtain a list of trackeable entities

ID	**T1.2.1**
Test	Nemergent MC-IoT Module queries INTER-FW for a list of authorised trackable entities.

Type	Service Discovery
Setup	Need test setup TS_01
Start	Nemergent CtrlRoom is properly launched and authenticated
Req.	[47], [53]
Input	Launch the jenkins job labeled as "**T1.2.1**"
Output	Check Jenkins job output, wether the job has succeeded or it has failed. Check outputted artifacts for debugging purposes.
Logs	Jenkins Job artifacts section: • backend_log.txt • frontend_log.txt • network_eth0.pcap • network_capture_logs.txt • test_report.html
Outcome	Pass / Fail

T1.2.2 Paint all of the trackeable entities obtained

ID	**T1.2.2**
Test	Nemergent MC-IoT Module passes the processed data to the Nemergent CtrlRoom main module and paints the listed entities on a map.
Type	System Testing
Setup	Need test setup TS_01
Start	Nemergent CtrlRoom is properly launched and authenticated
Req.	[47], [53]
Input	Launch the jenkins job labeled as "**T1.2.2**"
Output	Check Jenkins job output, wether the job has succeeded or it has failed. Check outputted artifacts for debugging purposes.
Logs	Jenkins Job artifacts section: • backend_log.txt • frontend_log.txt • network_eth0.pcap • network_capture_logs.txt • test_report.html
Outcome	Pass / Fail

T1.2.3 Subscribe to the event streams

ID	T1.2.3
Test	Nemergent MC-IoT Module performs subscription for the appropiate entities in order to maintain updated information about them, but without over saturating network and system resources.
Type	System Testing
Setup	Need test setup TS_01
Start	Nemergent CtrlRoom is properly launched and authenticated
Req.	[47], [51], [122]
Input	Launch the jenkins job labeled as "**T1.2.3**"
Output	Check Jenkins job output, wether the job has succeeded or it has failed. Check outputted artifacts for debugging purposes.
Logs	Jenkins Job artifacts section: • backend_log.txt • frontend_log.txt • network_eth0.pcap • network_capture_logs.txt • test_report.html
Outcome	Pass / Fail

T1.2.4 Receive updates from INTER-FW

ID	T1.2.4
Test	Nemergent MC-IoT Module passes the processed data to the Nemergent CtrlRoom main module and repaints the listed entities on a map.
Type	System Testing
Setup	Need test setup TS_01
Start	Nemergent CtrlRoom is properly launched and authenticated
Req.	[47], [51], [122]
Input	Launch the jenkins job labeled as "**T1.2.4**"
Output	Check Jenkins job output, wether the job has succeeded or it has failed. Check outputted artifacts for debugging purposes.
Logs	Jenkins Job artifacts section: • backend_log.txt • frontend_log.txt • network_eth0.pcap

	• network_capture_logs.txt • test_report.html
Outcome	Pass / Fail

T1.3.1 Receive detailed sensor data stream from INTER-FW

ID	T1.3.1
Test	Nemergent MC-IoT Module queries INTER-FW gateway for data streams specific to a device or sensor of an entity.
Type	System Testing
Setup	Need test setup TS_01
Start	Nemergent CtrlRoom is properly launched and authenticated
Req.	[47], [51], [122]
Input	Launch the jenkins job labeled as "**T1.3.1**"
Output	Check Jenkins job output, wether the job has succeeded or it has failed. Check outputted artifacts for debugging purposes.
Logs	Jenkins Job artifacts section: • backend_log.txt • frontend_log.txt • network_eth0.pcap • network_capture_logs.txt • test_report.html
Outcome	Pass / Fail

T1.3.2 Receive detailed sensor updates from INTER-FW

ID	T1.3.2
Test	Nemergent MC-IoT Module passes the processed data to the Nemergent CtrlRoom main module and displays detailed sensor info (ECG, Speed, Temp) in a detailed view.
Type	System Testing
Setup	Need test setup TS_01
Start	Nemergent CtrlRoom is properly launched and authenticated
Req.	[47], [51], [122]
Input	Launch the jenkins job labeled as "**T1.3.2**"
Output	Check Jenkins job output, wether the job has succeeded or it has failed.

	Check outputted artifacts for debugging purposes.
Logs	Jenkins Job artifacts section: • backend_log.txt • frontend_log.txt • network_eth0.pcap • network_capture_logs.txt • test_report.html
Outcome	Pass / Fail

T1.3.3 Stop receiving detailed sensor data stream from INTER-FW

ID	**T1.3.3**
Test	Nemergent MC-IoT Module queries INTER-FW gateway to stop (unsubscribe) from sensor data stream.
Type	System Testing
Setup	Need test setup TS_01
Start	Nemergent CtrlRoom is properly launched and authenticated
Req.	[47], [51], [122]
Input	Launch the jenkins job labeled as "**T1.3.3**"
Output	Check Jenkins job output, wether the job has succeeded or it has failed. Check outputted artifacts for debugging purposes.
Logs	Jenkins Job artifacts section: • backend_log.txt • frontend_log.txt • network_eth0.pcap • network_capture_logs.txt • test_report.html

3.3.4.6 Integration ethics and security

Introduction

During the Ethical Review it was requested that ethics had to be included within the project scope. This section addresses recommendations and proposed solutions to the ethical and security points of the integration.

Mission Critical operations based on IoT analytics

During the pilot execution, all the sensitive information (e.g., making reference to the INTER-IOT system or to personal data of people taking part of the pilot) should be hidden from non-authorized eyes at the moment of displaying the information at the Nemergent CtrRoom. The Nemergent CtrlRoom, as the user-endpoint of the system, is the last step in a chain of information processing, where all the information should have been secured and automated by design. Thus, special effort will be paid to ensuring that user privacy and system security is enforced at the Nemergent CtrlRoom GUI.

3.3.5 Third Party: Early Warning System (EWS)

The goal of the H2020 INTER-IoT project is to support interoperability between heterogeneous IoT platforms across the logistics and e-health domains. For demonstration and validation purposes, the project described scenarios to decrease the risk of fatal accidents at the port of Valencia, improving health prevention and enabling quick reaction by reducing time response . The goal of this scenario is to exploit how e-Health and e-Care can use IoT platforms dedicated to logistics to prevent the occurrence of accidents and to support evacuation or attention in case of emergency situations.

An early warning system (EWS) is a distributed system that monitors the physical world and issues warnings if it detects abnormal situations. The Internet-of-Things (IoT) offers opportunities to improve monitoring capabilities of EWS and to realize (near) real-time warning and response. The INTER-IoT-EWS goal is to detect accident risks with trucks that deliver goods at the Valencia port area, interoperating different IoT platforms. The solution addresses the semantic integration of a variety of data sources with processing in safety-critical applications for effective emergency response. The solution considers existing domain-specific ontologies and standards, along with their serialization formats. Accident risks are assessed by monitoring the drivers' vital signs with ECG medical wearables, and the trucks' position with speed and accelerometer data. Use cases include the detection of health issues and vehicle collision with dangerous goods. This EWS is developed with the SEMIoTICS framework, which is composed of a model-driven architecture that guides the application of data representations, transformations and distributed software components. This framework enables EWSs to be a semantic broker for situation-aware decision support.

Early Warning System (EWS)

A EWS is a system for "the provision of timely and effective information, through identified institutions, that allows individuals exposed to a hazard to take action to avoid or reduce their risk and prepare for effective response". An effective EWS must be people-centered and integrate knowledge about the risks, risks' monitoring and warning, dissemination of meaningful warnings and public awareness. Modern EWSs comprise software and hardware for data acquisition, situation awareness, decision making, and information dissemination. Current experimental prototypes incorporate IoT technology to improve their functionality. The conceptual architecture of EWS typically consists of three parts:

1. **Upstream data acquisition:** Distributed sensor systems transform observations into digital signals, pre-process the associated data values to ensure that they represent relevant information for decision making and transmit these data values to a message- and/or event-oriented middleware (broker).
2. **Decision support:** The data is stored in a data storage and is subjected to rules to detect situations of interest. The rules are represented as models, which can be deterministic (rule-based) and/or non-deterministic (machine learning) approaches. Once a situation is detected, the EWS considers the requirements of the alert targets to assess the risk and determine the emergency response.
3. **Downstream information dissemination:** Different target groups, comprising humans (e.g. the public) and machines (e.g. sirens), receive adequate messages.

Interoperability is an important feature of effective EWSs for the integration of internal components and interworking of different EWSs. The level of interoperability depends on the standardization of interfaces, data exchange formats and protocols . The design problem to be addressed here is the improvement of IoT EWSs' interoperability with data sources, alert targets, and other EWS to detect emergency risks.

3.3.5.1 INTER-IoT-EWS

Requirements and use cases

The requirements given in the scenario are:

(FR1) IoT platforms should be able to coordinate with emergency systems by detecting risks of accidents and accidents with trucks within the port area (collision and drivers' health issues), alerting urgency and severity. The acceptance criterion is to check if the port IoT platform is able to coordinate with emergency systems located in the vicinity.

(FR2) The haulier IoT platform and the port IoT platform should be able to share health information about the driver, monitored in real-time through an ECG device. These data need to be integrated with the port emergency control system.

(NFR1) IoT platforms should be semantically and syntactically interoperability. The acceptance criterion is the existence of a mechanism to translate data format and semantics of exchanged message to achieve communication with common understanding on both sides.

(NFR2) E-Health and logistics should be integrated at the application and semantics level, including primitives for data interpretation of medical and transportation data. **(NFR3)** The energy consumption of the devices being used for the situation identification mechanism should be monitored

Five use cases were conceived to test these requirements:

- **(UC01)** Vehicle collision detection: use of accelerometer data of the truck from mobile phone and health device;
- **(UC02)** Hazardous health changes: detect occurrences of stress and arrhythmia (e.g. bradycardia and tachycardia);
- **(UC03)** Temporal relations between UC01 and UC02: detect if a health issue occurred before, during or after a vehicle collision;
- **(UC04)** Wrong-way driving: integrate the trucks location data and the streets' direction within the port;
- **(UC05)** Accidents with dangerous goods: monitor dangerous goods being transported (according to UN list of dangerous goods) in all use cases (1-4), adding the adequate information regarding emergency procedures for effective response.

Note that UC03 has situations that require the integration of data from both domains (health and logistics) and can represent complex behaviors. For example, there is a possibility that bradycardia is detected followed by continuous decrease of the heart rate after a vehicle collision is detected. This situation reflects a car collision where the driver got injured and is classified as extreme severe with immediate urgency. In this situation the vehicle collision will be identified with both accelerometers from the ECG device and from the smartphone, considering device features, as accuracy and energy consumption.

3.3.5.2 Semantic IoT EWS framework

The "SEmantic Model-driven development for IoT Interoperability of emergenCy serviceS" (SEMIoTICS) is a framework to improve the semantic interoperability within and among EWSs. It consists of an architecture, technologies and guidelines based on model-driven engineering (MDE). SEMIoTICS uses the Endsley's situation awareness theory, which is harmonized with the Unified Foundational Ontology (UFO) and aligned to the semantic healthcare system architecture.

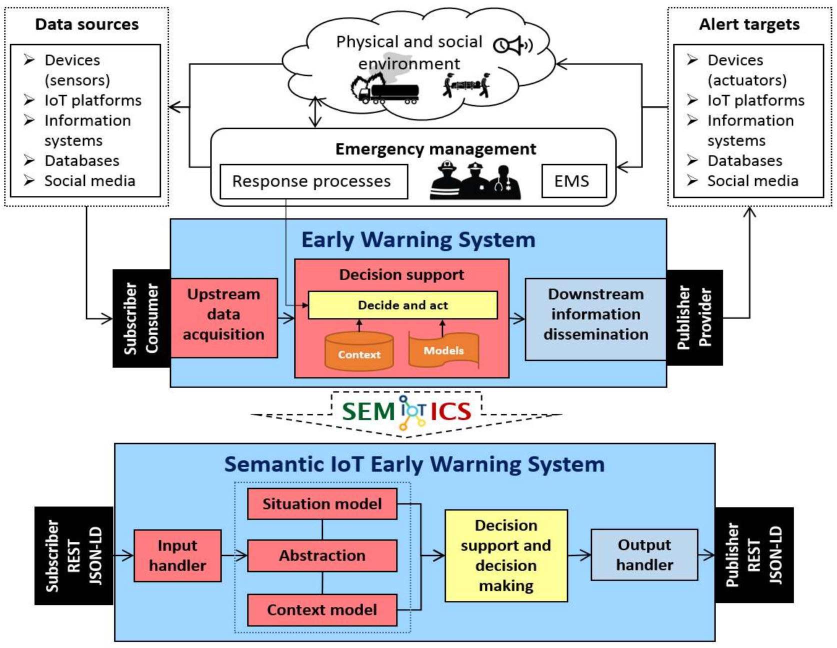


Figure 102: Typical EWS architecture (top) and the SEMIoTICS architecture (bottom).

The architecture has six elements addressing the 3 main functions of an EWS: (1) Input handler: upstream data acquisition through adaptors; (2) Abstraction: foundational ontology; (3) Context model: domain ontology; (4) Situation model: complex event processing; (5) Situation awareness: data flows, (6) Output handler: downstream emergency notification. It follows the publisher/subscriber pattern and RESTful services with JSON-LD and XML.

Adaptors are implemented as syntactic and semantic translations. The input handler is responsible for message translation, which relies on the syntax of each ontology being used and, therefore, will also require semantic as well as syntactic translations, e.g. from RDF/XML to JSON-LD and from HL7 to EDXL. Messages are translated from the original ontologies to our context model (core ontology), which is aligned to W3C SSN and incorporates terms from EDXL and HL7. This approach aims on optimizing the data and semantics maintenance when integrating distinct domains. The abstraction component refers to foundational ontologies, which are designed to maximize the support for interoperability of high level categories, e.g. event, process, physical object and system. The core ontology and SSN are grounded on the UFO (through OntoUML) and DOLCE Ultralite (DUL), respectively. UFO and DOLCE share the same definitions for some conceptualizations, facilitating the alignment between the ontologies extended with them.

The situation model is responsible for the situation identification mechanism, i.e. the formalization of the emergency risk detection. We adopted a rule-based approach, allowing the specification and implementation of complex event processing (CEP). CEP is a common component of IoT platforms to correlate data using temporal predicates (events' relations), as Cepheus, the CEP engine of FIWARE IoT platform. Guidelines describe how CEP technologies can implement the situation models, e.g. in Java ESPER24 and Drools Fusion technologies. Decision support is enabled by the adoption of a workflow management system that enables the end user to design business processes, e.g. emergency plans, as data flows. Big data integration tools for workflow development automatically generates code and is able

[24] http://www.espertech.com/

to deploy data flows at runtime, e.g. Node-Red25. This component also covers the deployment and execution of the data flows for decision making. The output handler is responsible for brokering the emergency risk notifications to the correct targets, according to the emergency procedures defined on the decision support component. For each predetermined risk, targets are enumerated with their information requirements. The data format of the notifications follows EDXL standards serialized as JSON-LD. The risk notification services are exposed as data publishers.

3.3.5.3 Solution

The solution architecture Figure 103 includes the Shimmer ECG 3 device[26] to collect ECG data from drivers. This device has high accuracy and usability, and is compatible with the SPINE framework[27], which was chosen by the project consortium. SPINE comprises a TinyOS application responsible for transmitting data from the ECG device to a mobile phone through Bluetooth. SPINE also includes an Android application to receive and then forwards the data to the cloud, as a gateway. These data is sent to the cloud and published in a broker as RDF/XML messages following the ETSI SAREF[28] ontology extended with HL7 aECG, supported by the UniversAAL IoT platform[29].

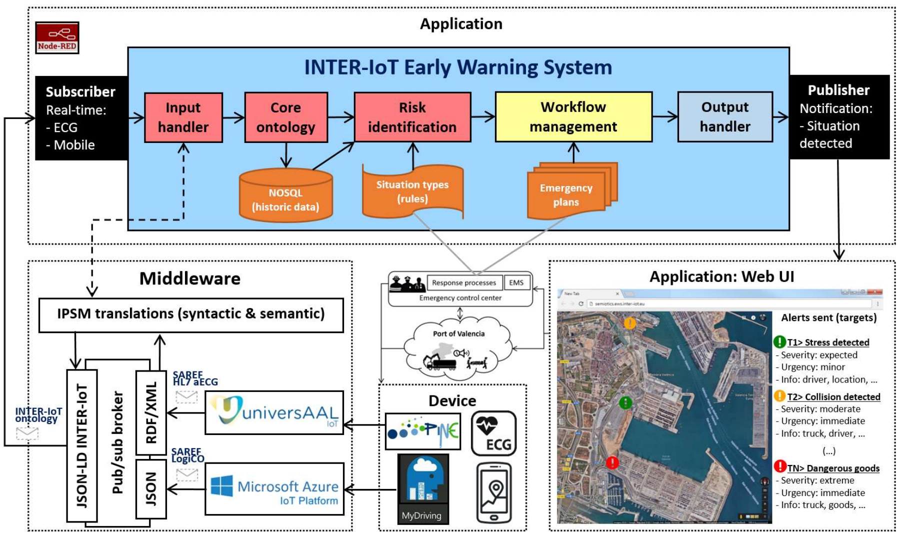


Figure 103: EWS to detect accident risks and accidents at the port of Valencia.

Similarly, the MyDriving mobile application for logistics (open use case of Azure IoT platform30) transmits the data about the truck position, speed, accelerometer and goods information, to the cloud infrastructure. These logistics data are serialized as JSON messages, following the structure of SAREF ontology aligned to LigiCO ontology31. SAREF was chosen because of its capabilities for tracking devices' energy consumption. IPSM module is responsible for syntactically and semantically translate these data: from JSON and RDF/XML

[25] https://nodered.org/
[26] http://www.shimmersensing.com/products/ecg-development-kit
[27] http://spine.deis.unical.it/spine.html
[28] http://ontology.tno.nl/saref/
[29] http://www.universaal.info/
[30] https://azure.microsoft.com/en-us/campaigns/mydriving/
[31] http://ontology.tno.nl/logico/

to the INTER-IoT JSON-LD syntax, which is structured JSON-LD (two @graph) with middleware information; and from SAREF to the INTER-IoT core ontology semantics, which is aligned to SSN. Theses translations are configured priory in IPSM by the application developer through a REST service.

The data represented as INTER-IoT JSON-LD syntax and INTER-IoT core ontology semantics are published in the broker in a topic, which the EWS subscribes to receive real-time data. Then, the EWS input handler certifies whether new translations to harmonize the data in the SEMIoTICS core ontology are necessary and, if so, the input handler requests the translations to IPSM.

The data is annotated with the core ontology and stored in a NOSQL database (MongoDB) for historic data storage. Both real-time data and historic data are used by the risk identification component, i.e. the NESPER CEP engine. Situation types are defined a priori, as rule sets, describing the risky situations of interest based on emergency plans. Each situation type is linked to a response process, i.e. the specific workflow to be executed once a situation is identified. Therefore, the risk identification component triggers the workflow management, which executes the linked processes. The workflow component is responsible for checking the information requirements of each alert target, passing this information to the output handler, which is responsible to transform the data to EDXL compliant messages semantically enriched. Therefore, the output handler enables the brokering of notifications of situations detected, following the JSON-LD syntax and the EDXL structure, which is able to link to the semantics used. A web UI application shows each alert sent by the EWS with its severity and urgency, and other information, including the targets that received the notification and the message sent to each target. The EWS is developed with NodeJS and Node-Red. Table 1 summarizes the involved data.

External	Health	Logistics
Data	Driver's ECG, accelerometer	Position, speed, accelerometer, goods
Device	Shimmer (SPINE), Mobile	Mobile (MyDriving Android or iOS)
IoT platform	UniversAAL	MS Azure IoT
Ontologies	ETSI SAREF, HL7/FHIR aECG	ETSI SAREF, LogiCO

Table 73. Data sources.

3.3.5.4 Validation plan

Validating the achievement of providing effective situation awareness and emergency response requires a comparison whether the response processes triggered through the workflow management is adherent to emergency procedures, reflecting pragmatic interoperability between the EWS and an emergency manager. This will measure whether the system works for the intended risks' detection and warning. So, it includes simulation of the use cases (test cases) with multiple target groups with different information requirements. EDXL validators will be used to check that proper EDXL messages are generated, e.g. Google's EDXL-CAP validator.

The validation plan is organized as (a) factory acceptance tests: in a lab environment the EWS will be deployed in a cloud environment and the components integration will be tested through mock objects; and (b) site acceptance tests: a pilot in the port, where accidents will be simulated in accordance with the port emergency exercises, e.g. vehicle collision through hard breaks, bradycardia/tachycardia by decreasing the thresholds and adequate response procedures for accidents with dangerous goods.

The validation plan includes the performance evaluation of data transfer, processing and storing JSON-LD as payload. Total time to be observed: (i) for data acquisition; (ii) to semantically translate a message; (iii) to annotate data with the core ontology and insert into the NOSQL database; (iv) to access data (from memory and database) and process (serialize and deserialize) for risk identification; and (v) to create the alert messages, i.e. serialize the output data as EDXL linked to ontologies. The brokering performance will also be evaluated in terms of scalability and resilience for single cluster and multi-broker, as the semantic IoT EWS approach.

Validating the achievement of semantic interoperability depends on whether the components of the solution have the same "understanding" of the data. Since the approach is based on multiple semantic translations, semantic loss will be a variable to calculate semantic interoperability, measured by executing the transformations in sequence from ontology A (e.g. SAREF) to ontology B (e.g. SSN) and from B to A, i.e. check how x is different to $T(T(x)_{A>B})_{B>A}$, where $T(x)_{A>B}$ represents the semantic translation function from A to B.

This plan includes data management with the FAIR data principles.

3.3.5.5 Deliverables and version overview

The following table contains a deliverable list which need to be signed of before FAT testing commences.

ID	Description	Check
Documents		
1	Validation and Test reports of the IoT system components	
2	Validation and Test reports of the Pilot system components	
Hardware		
4	Application server (EWS orchestrator)	
5	Application server (CEP NESPER)	
6	Database server	
Tools		
7	EWS (see section 3.2 and 6.2)	

Table 74: Deliverable checklist

3.3.5.6 Requirements, scenarios and use cases

The following table provides an overview of the relation between the requirements and the test(s) that validate their implementation.

ID	Description	Covered by
Application		
239	Support Service choreography and Service Orchestration	All functional
Architecture		
6	Efficiency of the processing of information	Non-functional (7.1.6.2-4)
Functionality		
20	Real time support	All functional
21	Real time output	All functional
23	Device semantic definition	All functional and 7.1.6.1

179	IoT Platform Semantic Mediator supports platform to platform communication and communication between platforms and an external actor	All functional
Interoperability		
13	Extensibility	Tests involving SAREF
Operational		
57	Device monitoring and self-awareness of the system	All functional
73	Analyzing data from heterogeneous platforms	7.1.3
76	Interoperability between things from different administrative/management domains	7.1.3
178	IoT Platform Semantic Mediator provides data and semantic interoperability functionality accessible with a set of interfaces	All functional and 7.1.6.1
Performance		
72	Communication should be done using protocols that are efficient in terms of amount of exchanged information over message size	All functional (track battery consumption)
Semantics		
163	Design support for semantic interoperability	All functional (semantic translations) and 7.1.6.1.
180	Semantic and syntactic interoperability	All functional
186	Design of required ontologies	All functional
224	Location semantic support for mobile smart objects	SAREF and SSN support
225	Special considerations in the semantic ontology to objects with low resources	All functional (battery consumption tracking with SAREF).

Table 75: Requirements vs test mapping

The following table provides an overview of the relation between the scenarios and the test(s) that validate their implementation.

ID	Scenario name	Covered by
1	Chronic disease prevention	
2	IoT support for transport planning and execution	
3	IoT Weighbridges	
4	Monitoring reefer container	
5	Monitoring of containers carrying sensitive goods	
6	Dynamic lighting in the port	
7	SCADA port sensor system integration with IoT platforms	
8	SEAMS integration with IoT platforms	
9	Accident at the port area	**[All tests]**
10	Health monitoring system with passengers aboard a ferry	
11	Primary prevention of cognitive decline	
12	Health failure disease and mild Alzheimer disease	
13	IoT interoperability for Vessel Arrivals	
15	Surveillance systems for prevention programs	
16	Elderly monitoring	
17	Health monitoring system with passengers aboard a train	
18	Containership is entering the harbour region	
19	Transport on truck breaks down or is hijacked	

20	Damage or problems to the container during shipment	
21	Low risk of developing chronic diseases.	
22	Increased risk of developing chronic diseases	
23	High risk of developing chronic diseases	
24	Very high risk of developing chronic diseases	
25	Extremely high risk of developing chronic diseases	
26	Alcohol / Drug testing for truck/ bus drivers	
27	Vitamins intake analyser	
28	Calories / nutrition mixer / cookware counter	
29	Reliable control of robotic cranes and trucks in port terminals	
30	IoT access control, traffic and operational assistance	

Table 76: Scenario vs test mapping

3.3.5.7 Test environment

Introduction

To test the functionality of the T_AccidentPort module in combination with the IoT framework a representative test system is needed. The test system needs to approach the "real world" as much as possible. The pilot setups must be recreated and proven. This chapter will describe this environment and the used hardware, software, tools and platforms.

As illustrated in Figure 103, the system under tests is the INTER-IoT early warning system (INTER-IoT-EWS or EWS), thus, the focus of this validation plan is to evaluate whether the INTER-IoT-EWS addresses the involved requirements.

How FAT will be performed: The EWS will be deployed in a cloud environment, organized as:

1. Workflow/dataflow deployed in NodeRed test environment of INTER-IoT.
2. The EWS deployed in an application server.
3. The MongoDB NOSQL deployed in a database server.
4. The CEP engine deployed in an application server.

Where FAT will be performed: in the U.Twente laboratory, accessing the cloud environment described above.

Test environment

This paragraph describes the test environment and the complete system setup used during this FAT.

As described above, the EWS components will be deployed in 3 servers in the cloud, which will be coordinated through the NodeRed test environment of INTER-IoT[32]. The exact servers are still being analyzed, but the options are:

(a) Internal (private) cloud environment of U.Twente;
(b) Cloud services of Amazon AWS or MS Azure or Google Cloud, according to the privacy issues required.

The application server of the EWS will require Windows OS with .NET framework (4.5.2), being able to access the data provided by INTER-MW through NodeRed workflows (REST service). The EWS will also require access to the database and CEP servers, to store and reason over

[32] https://nodered.inter-iot.eu

the data. The EWS will publish emergency notification messages in INTER-MW through NodeRed (and/or NodeJS).

3.3.5.8 Test setups, tools, hooks and probes

This paragraph describes the test setups, tools, hooks and probes used in the by this document defined tests.

A test setup defines a setup used for a test, and describe the used system parts, interface etc.

Tools define the used COTS tooling that will be used during testing e.g. Logic analyzers, packet sniffer like e.g. Whireshark, developed scripts, etc.

Hooks are used to inject data or control parts of the system used to get the system to handle a needed scenario. This can be hidden, debug or service interfaces which contain the needed functionality.

Probes are used to gather logging from parts of the system under test to provide system feedback and to prove that the system is handling the scenario as it should.

The following paragraphs will define the used setups, tools, hooks and probes which will be referred to by the test descriptions.

Obs.: Test data generation will be detailed in the data management plan.

TS_01 EWS deployment with I/O of messages with files

Basic environment for deployment of the EWS components:

- Application server running the EWS (Node-Red, NodeJS)
- Application server running CEP (Drools SCENE or NESPER)
- Database server running NOSQL database (MongoDB)

TS_02 EWS deployment consuming from INTER-MW

Same as TS_01, but input data acquired with INTER-MW test environment.

TS_03 EWS deployment publishing on EMS

Same as TS_01, but output data being consumed by an emergency system (either EMS or incident management system or equivalent).

TS_04 EWS deployment consuming from INTER-MW and publishing on EMS

Conjunction of TS_02 with TS_03.

TT_01 Visual Studio unit tests

Unit tests coded in Visual Studio will be used to generate test data from the devices being used (smartphone and Shimmer ECG).

TT_02 NodeJS unit tests

Mocha and Chai libraries for creating unit tests for each component of the EWS

https://www.codementor.io/davidtang/unit-testing-and-tdd-in-node-js-part-1-8t714s877

TT_03 Wireshark

For testing issues regarding network level, especially on the communication of the EWS components and between the EWS with input provider and output consumer.

https://www.wireshark.org/

TT_04 MongoDB NodeJS unit test

Mongo-unit test library for NodeJS will be used to test the storage of data, i.e. basic CRUD methods.

https://www.npmjs.com/package/mongo-unit

TT_05 CEP unit test

Unit test tooling support includes CEP tests.

If SCENE (Drools) is used in the solution, then Junit will be considered:

https://www.packtpub.com/books/content/testing-your-jboss-drools-business-rules-using-unit-testing

If EPL (Event Processing Language) is used (with another CEP engine, e.g. NESPER), then an EPL-based unit test library will be used, such as:

https://github.com/antoinewaugh/aunit

http://www.apamacommunity.com/unit-testing-with-aunit/

TT_06 EDXL validator

An EDXL validator will be used to syntactically validate the messages generated, such as Google's EDXL-CAP validator:

https://cap-validator.appspot.com/

And SharpEDXL:

https://edxlsharp.codeplex.com/

TT_06 Google MyMaps and MyGeoData

Common routes in the port area of trucks (from the involved haulier company) will be generated with support of Google MyMaps (http://google.com/mymaps). For example, this route from Noatum to TVC terminal:

https://drive.google.com/open?id=1GiFTkT1-YhJqb7Y3GMWHeWTaeyLRX8zI&usp=sharing

Once the routes are created, each route will be exported as KML (KMZ) and then converted either for GeoJSON with support of MyGeoData (https://mygeodata.cloud/conversion). The route points in the converted data file will be used for the representation of "trip points" within the input test messages.

3.3.5.9 Test description

Test output log files… Folder "`Tx_Output`", prefix "`Tx.y.1_`"

Scenario: accidents at the port area [id.9]

The functional goal of this scenario is to decrease the risk of fatal accidents at the port of Valencia, improving health prevention and enabling quick reaction by reducing time response.

The non-functional goal of this scenario is to exploit how e-Health and e-Care can use IoT platforms dedicated to logistics to prevent the occurrence of accidents and to support evacuation or attention in case of emergency situations: "interoperate the wearable medical devices with IoT platforms (…) to react quickly, thus reducing time responses during accidents and health prevention" [INTER-IoT deliverable 2.4].

Interoperability in this scenario is required to connect the port authority (including emergency systems) and the road hauliers IoT platforms. The haulier solution is composed by two IoT platforms: one representing logistics data (implemented with Azure IoT) and one representing health data (implemented with BodyCloud (including SPINE) and UniversAAL).

This scenario involves these requirements:

ID	Description	Covered by
6	Efficiency of the processing of information	
20	Real time support	
23	Device semantic definition	
179	IoT Platform Semantic Mediator supports platform to platform communication and communication between platforms and an external actor	
73	Analyzing data from heterogeneous platforms	
72	Communication should be done using protocols that are efficient in terms of amount of exchanged information over message size	
163	Design support for semantic interoperability	
180	Semantic and syntactic interoperability	[T1.3], [T2.1]…
186	Design of required ontologies	
249	NFR: Semantic interoperability among platforms	[D2.3]
251	FR: IoT platforms to coordinate with emergency systems	[D2.3]

3.3.5.9.1 UC01: Vehicle collision detection

Monitor the truck's location and detect possible collisions (impacts). In general, the approaches use an accelerometer within the vehicle to collect time series data about its location, i.e. the device's acceleration about the corresponding axes (X, Y, Z), allowing the calculation of the G-Force felt in each instant. Then, for each instant, the detection mechanism compares if the G-Force is above a certain threshold, which is usually 3G for devices deployed in the vehicle chassis. According to the patent for "vehicle security with accident notification and embedded driver analytics" (US 9491420 B2), "instances of high acceleration/deceleration are due to a large change in velocity over a very short period of time. These speeds are hard to attain if a vehicle is not controlled by a human driver, which simplifies accident detection since we can assume any instance of high acceleration constitutes a collision involving human drivers". An approach using a smartphone sharing accelerometer data is described. The Shimmer ECG 3 also provides accelerometer data, thus, it can also provide accelerometer data, providing an opportunity to integrate the health and logistics solutions.

Classification of severity and urgency according to accelerometer data (A) and threshold (B) is described in the table below. In summary, if the cross-axial energy computed is greater than the threshold and less than 20% above the threshold, then it might be a light collision (minor severity). If it is in-between 20% and 40%, then the collision is greater (moderate severity), if it is in-between 40% and 60%, then the collision is severe. Above 60% represents a strong impact, thus, an extreme severity, which probably needs immediate urgency for emergency response.

Range	Severity	Urgency
B < A <= B * 1.2	Minor	Expected
B * 1.2 < A <= B * 1.4	Moderate	Immediate

B * 1.4 < A <= B * 1.6	Severe	Immediate
B * 1.6 < A	Extreme	Immediate

Each test case has an equivalent input and output data file, named TX.Y. json (input and output folders). The type of all test cases here are system testing using scripted data.

This use case involves these requirements: [23], [72], [180], [249] and [251].

Detected with smartphone accelerometer

ID	T1.1
Test	Vehicle collision detected with smartphone accelerometer data.
Type	System testing using scripted data
Setup	Need test setup TS_01
Start	Vehicle is in one of the port gates (entering the port area).
Req.	[180], [251]
Input	`T1_input/T1.1.json`: during a trip within the port area the accelerometer changes above the threshold, i.e. accelerometer value within an one trip point.
Output	`T1_output/T1.1.json`: EDXL (SitRep and TEP)
Logs	`T1_output/T1.1_logname`
Outcome	Pass / Fail

Detected with medical wearable accelerometer

ID	T1.2
Test	Vehicle collision detected with smartphone accelerometer data.
Type	System testing using scripted data
Setup	Need test setup TS_01
Start	Vehicle is in one of the port gates (entering the port area).
Req.	[180], [251]
Input	`T1_input/T1.2.json`: during a trip within the port area the accelerometer changes above the threshold, i.e. accelerometer value within an one trip point.
Output	`T1_output/T1.2.json`: EDXL (SitRep and TEP)
Logs	`T1_output/T1.2_logname`
Outcome	Pass / Fail

Detected with smartphone and medical wearable accelerometer

ID	**T1.3**
Test	Vehicle collision detected by using both accelerometer data within a window time.
Type	System testing using scripted data
Setup	Need test setup TS_01
Start	Vehicle is in one of the port gates (entering the port area).
Req.	[180], [249], [251]
Input	`T1_input/T1.3_logistics.json`, `T1.3_health.json`: during a trip within the port area the accelerometer changes above the threshold, i.e. accelerometer value within an one trip point.
Output	`T1_output/T1.3.json`: EDXL (SitRep and TEP)
Logs	`T1_output/T1.3_logname`
Outcome	Pass / Fail

Detected with smartphone or medical wearable accelerometer

ID	**T1.4**
Test	Vehicle collision detected through the rule that checks the battery consumption of the devices and decides which accelerometer data should be used.
Type	System testing using scripted data
Setup	Need test setup TS_01
Start	Vehicle is in one of the port gates (entering the port area).
Req.	[180], [249], [251]
Input	`T1_input/T1.4_logistics.json`, `T1.4_health.json`: during a trip within the port area the accelerometer changes above the threshold, i.e. accelerometer value within an one trip point.
Output	`T1_output/T1.4.json`: EDXL (SitRep and TEP)
Logs	`T1_output/T1.4_logname`
Outcome	Pass / Fail

Detected according to T1.1 with 4 classifications of severity + urgency

ID	**T1.5**

Test	Four executions of T1.1 resulting on the classifications of severity and urgency below (green, yellow, light red, dark red).
Type	System testing using scripted data
Setup	Need test setup TS_01
Start	Vehicle is in one of the port gates (entering the port area).
Req.	[180], [251]
Input	`T1_input/T1.5_level[1,2,3,4].json`: during a trip within the port area the accelerometer changes above the threshold according to the levels of urgency/severity (table XX).
Output	`T1_output/T1.5_level[1,2,3,4].json`: EDXL (SitRep and TEP)
Logs	`T1_output/T1.5_logname`
Outcome	Pass / Fail

Detected according to T1.2 with 4 classifications of severity + urgency

ID	**T1.6**
Test	Four executions of T1.2 resulting on the classifications of severity and urgency above (green, yellow, light red, dark red).
Type	System testing using scripted data
Setup	Need test setup TS_01
Start	Vehicle is in one of the port gates (entering the port area).
Req.	[180], [251]
Input	`T1_input/T1.6_level[1,2,3,4].json`: during a trip within the port area the accelerometer changes above the threshold according to the levels of urgency/severity (table XX).
Output	`T1_output/T1.6_level[1,2,3,4].json`: EDXL (SitRep and TEP)
Logs	`T1_output/T1.6_logname`
Outcome	Pass / Fail

Detected according to T1.3 with 4 classifications of severity + urgency

ID	**T1.7**
Test	Four executions of T1.3 resulting on the classifications of severity and urgency above (green, yellow, light red, dark red).

Type	System testing using scripted data
Setup	Need test setup TS_01
Start	Vehicle is in one of the port gates (entering the port area).
Req.	[180], [249], [251]
Input	`T1_input/T1.7_level[1,2,3,4].json`: during a trip within the port area the accelerometer changes above the threshold according to the levels of urgency/severity (table XX).
Output	`T1_output/T1.7_level[1,2,3,4].json`: EDXL (SitRep and TEP)
Logs	`T1_output/T1.7_logname`
Outcome	Pass / Fail

Detected according to T1.4 with 4 classifications of severity + urgency

ID	**T1.8**
Test	Four executions of T1.4 resulting on the classifications of severity and urgency above (green, yellow, light red, dark red).
Type	System testing using scripted data
Setup	Need test setup TS_01
Start	Vehicle is in one of the port gates (entering the port area).
Req.	[180], [249], [251]
Input	`T1_input/T1.8_level[1,2,3,4].json`: during a trip within the port area the accelerometer changes above the threshold according to the levels of urgency/severity (table XX).
Output	`T1_output/T1.8_level[1,2,3,4].json`: EDXL (SitRep and TEP)
Logs	`T1_output/T1.8_logname`
Outcome	Pass / Fail

3.3.5.9.2 UC02: Hazardous health changes

Detect medical issues with the driver by monitoring his/her ECG and derived heart rate, checking possible cardiovascular emergencies. Cardiovascular emergencies are life-threatening disorders that must be recognized as soon as possible to minimize morbidity and mortality. By allowing the EWS to detect cardiovascular emergencies with trucks' drivers, it is possible to reduce the risk of an accident at the port area. The EWS provides messages that include the information of the cardiovascular emergency situation.

This can be achieved, basically, by using the the INTER-Health IoT solution with Shimmer ECG device attached to the driver's chest, wired to electrodes, and an Android-based mobile phone, both part of the Body module of the BodyCloud approach implemented with the SPINE framework. Thresholds used by the detection mechanism should be based on existing classifications to detect health risks. For example, target heart rates used for fitness is a classification of indicators that can be used as a baseline for thresholds. The table below illustrates such indicators (red, green, yellow, blue) depending on the person's age. Besides these thresholds, this use case also considers the situations which the driver presents bradycardia and tachycardia, which can be detected with the ECG device (event monitor)[33].

Classification of severity and urgency according to ComputeBPM output (A) and the threshold (B) is described in the table below. In summary, if the BPM calculated is greater than the threshold and less than threshold more 10%, then it might be a light tachycardia (minor severity). If it is in-between 10% and 20%, then the tachycardia is greater (moderate severity), if it is in-between 20% and 30%, then the tachycardia is severe. Greater than 30% represents a strong tachycardia, thus, an extreme severity, which probably needs immediate urgency for emergency response.

Range	Severity	Urgency
B < A <= B * 1.1	Minor	Expected
B * 1.1 < A <= B * 1.2	Moderate	Immediate
B * 1.2 < A <= B * 1.3	Severe	Immediate
B * 1.3 < A	Extreme	Immediate

Each test case has an equivalent input and output data file, named TX.Y. json (input and output folders). The type of all test cases here are system testing using scripted data.

This use case involves these requirements: [23], [72], [180], [249] and [251].

Bradycardia detected with fixed threshold

ID	**T2.1**
Test	From ECG data, the heart rate is calculated and compared to a threshold.
Type	System testing using scripted data
Setup	Need test setup TS_01
Start	Vehicle is in one of the port gates (entering the port area).
Req.	[180], [249], [251]
Input	`T2_input/T2.1_level[1,2,3,4].json`: during a trip within the port area the heart rate is below the threshold according to the levels of urgency/severity (table XX).
Output	`T2_output/T2.1_level[1,2,3,4].json`: EDXL (SitRep and TEP)
Logs	`T2_output/T2.1_logname`

[33] http://www.mayoclinic.org/diseases-conditions/bradycardia/diagnosis-treatment/diagnosis/dxc-20321665
http://www.mayoclinic.org/diseases-conditions/tachycardia/diagnosis-treatment/diagnosis/dxc-20253919

Outcome	Pass / Fail

Bradycardia detected with dynamic threshold

ID	T2.2
Test	From ECG data, the heart rate is calculated and compared to a threshold.
Type	System testing using scripted data
Setup	Need test setup TS_01
Start	Vehicle is in one of the port gates (entering the port area).
Req.	[180], [249], [251]
Input	`T2_input/T2.2_level[1,2,3,4].json`: during a trip within the port area the heart rate is below the threshold according to the levels of urgency/severity (table XX).
Output	`T2_output/T2.2_level[1,2,3,4].json`: EDXL (SitRep and TEP)
Logs	`T2_output/T2.2_logname`
Outcome	Pass / Fail

Tachycardia detected with fixed threshold

ID	T2.3
Test	From ECG data, the heart rate is calculated and compared to a threshold.
Type	System testing using scripted data
Setup	Need test setup TS_01
Start	Vehicle is in one of the port gates (entering the port area).
Req.	[180], [249], [251]
Input	`T2_input/T2.3_level[1,2,3,4].json`: during a trip within the port area the heart rate is above the threshold according to the levels of urgency/severity (table XX).
Output	`T2_output/T2.3_level[1,2,3,4].json`: EDXL (SitRep and TEP)
Logs	`T2_output/T2.3_logname`
Outcome	Pass / Fail

Tachycardia detected with dynamic threshold

ID	T2.4

Test	From ECG data, the heart rate is calculated and compared to a threshold.
Type	System testing using scripted data
Setup	Need test setup TS_01
Start	Vehicle is in one of the port gates (entering the port area).
Req.	[180], [249], [251]
Input	`T2_input/T2.4_level[1,2,3,4].json`: during a trip within the port area the heart rate is above the threshold according to the levels of urgency/severity (table XX).
Output	`T2_output/T2.4_level[1,2,3,4].json`: EDXL (SitRep and TEP)
Logs	`T2_output/T2.4_logname`
Outcome	Pass / Fail

Multiple occurrences of bradycardia detected with fixed threshold

ID	T2.5
Test	Several occurrences of T2.1 over a window of time (5 min).
Type	System testing using scripted data
Setup	Need test setup TS_01
Start	Vehicle is in one of the port gates (entering the port area).
Req.	[180], [249], [251]
Input	`T2_input/T2.5_level[1,2,3,4].json`: during a trip within the port area the heart rate is below the threshold for a time window, according to the levels of urgency/severity (table XX).
Output	`T2_output/T2.5_level[1,2,3,4].json`: EDXL (SitRep and TEP)
Logs	`T2_output/T2.5_logname`
Outcome	Pass / Fail

Multiple occurrences of bradycardia detected with dynamic threshold

ID	T2.6
Test	Several occurrences of T2.2 over a window of time (5 min).
Type	System testing using scripted data
Setup	Need test setup TS_01

Start	Vehicle is in one of the port gates (entering the port area).
Req.	[180], [249], [251]
Input	`T2_input/T2.6_level[1,2,3,4].json`: during a trip within the port area the heart rate is below the threshold for a time window, according to the levels of urgency/severity (table XX).
Output	`T2_output/T2.6_level[1,2,3,4].json`: EDXL (SitRep and TEP)
Logs	`T2_output/T2.6_logname`
Outcome	Pass / Fail

Multiple occurrences of tachycardia detected with fixed threshold

ID	T2.7
Test	Several occurrences of T2.3 over a window of time (5 min).
Type	System testing using scripted data
Setup	Need test setup TS_01
Start	Vehicle is in one of the port gates (entering the port area).
Req.	[180], [249], [251]
Input	`T2_input/T2.7_level[1,2,3,4].json`: during a trip within the port area the heart rate is above the threshold for a time window, according to the levels of urgency/severity (table XX).
Output	`T2_output/T2.7_level[1,2,3,4].json`: EDXL (SitRep and TEP)
Logs	`T2_output/T2.7_logname`
Outcome	Pass / Fail

Multiple occurrences of tachycardia detected with dynamic threshold

ID	T2.8
Test	Several occurrences of T2.4 over a window of time (5 min).
Type	System testing using scripted data
Setup	Need test setup TS_01
Start	Vehicle is in one of the port gates (entering the port area).
Req.	[180], [249], [251]

Input	`T2_input/T2.8_level[1,2,3,4].json`: during a trip within the port area the heart rate is above the threshold for a time window, according to the levels of urgency/severity (table XX).
Output	`T2_output/T2.8_level[1,2,3,4].json`: EDXL (SitRep and TEP)
Logs	`T2_output/T2.8_logname`
Outcome	Pass / Fail

Large variation of heart rate

ID	**T2.9**
Test	Detect whether variations occur. If there is a variation of more than 50% of the heart rates collected during a period of 5 minutes, then this situation is detected.
Type	System testing using scripted data
Setup	Need test setup TS_01
Start	Vehicle is in one of the port gates (entering the port area).
Req.	[180], [249], [251]
Input	`T2_input/T2.9.json`: during a trip within the port area the heart rate suffers large variation.
Output	`T2_output/T2.9.json`: EDXL (SitRep and TEP)
Logs	`T2_output/T2.9_logname`
Outcome	Pass / Fail

Detect high level of stress

ID	**T2.10**
Test	Based on UNICAL solution with Cardiac Defense Response (CDR).
Type	System testing using scripted data
Setup	Need test setup TS_01
Start	Vehicle is in one of the port gates (entering the port area).
Req.	[180], [249], [251]
Input	`T2_input/T2.10.json`: during a trip within the port area high-level of stress is detected.
Output	`T2_output/T2.10.json`: EDXL (SitRep and TEP)
Logs	`T2_output/T2.10_logname`

Outcome	Pass / Fail

3.3.5.9.3 UC03: Temporal relations (UC01 ~ UC02)

This use case exploits the possible temporal relations between UC01 and UC02 for detection of an accidents in the port area. For example, if a truck collision is detected from the accelerometers of the medical and mobile devices (T1.3) and right after (e.g. within 1-2 minutes) detecting large variation of heart rate (T2.9) then there is a high probability that a severe accident occurred, the driver is injured and he/she requires urgent medical help. Notice that the temporal relationship ("right after") is crucial to integrate these use cases.

Vehicle collision followed by bradycardia

ID	**T3.1**
Test	Slow heart rate right after (within 2 minutes) a collision is detected can represent that an accident just occurred and the driver is probably injured.
Type	System testing
Setup	<Describe the needed setup, tools, hooks and probes needed for this test>
Start	<Describe the system state for the start of the test>
Req.	<Define the requirements involved in [x], format>
Input	<Define the test input steps, e.g. Inject data x in interface y>
Output	<Define the expected result, e.g. Expected to receive event z>
Logs	<Define where the output log is stored, e.g.: Folder "`Tx_Output`", prefix "`Tx.y.1_logname`" >
Outcome	Pass / Fail

3.3.5.9.4 UC04: Wrong-way driving

This use case exploits the possible temporal relations between UC01 and UC02 for detection of an accidents in the port area.

Truck on opposite direction of a street within the port

ID	**T4.1**
Test	From the position data (mobile), the EWS will check the street direction and compare to the truck's position change within 30 seconds.
Type	System testing
Setup	<Describe the needed setup, tools, hooks and probes needed for this test>

Start	<Describe the system state for the start of the test>
Req.	<Define the requirements involved in [x], format>
Input	<Define the test input steps, e.g. Inject data x in interface y>
Output	<Define the expected result, e.g. Expected to receive event z>
Logs	<Define where the output log is stored, e.g.: Folder "`Tx_Output`", prefix "`Tx.y.1_logname`" >
Outcome	Pass / Fail

3.3.5.9.5 UC05: Accident involving dangerous goods

This use case will extend the use cases UC01-04 by checking whether dangerous goods are being transported, which will increase the situation urgency and severity and include the dangerous goods classification according to UNECE[34]. Data test will include simulation of trips including the transportation of class 1 (explosives), 3 (flammable liquids), 4 (flamed solids), 6 (toxic and infectious) and 7 (radioactive).

3.3.5.9.6 UC01 with dangerous goods

ID	**T5.1**
Test	Tests of UC01 incremented with a check whether dangerous goods are being transported.
Type	System testing
Setup	<Describe the needed setup, tools, hooks and probes needed for this test>
Start	<Describe the system state for the start of the test>
Req.	<Define the requirements involved in [x], format>
	Input
	Output
Logs	<Define where the output log is stored, e.g.: Folder "`Tx_Output`", prefix "`Tx.y.1_logname`" >
Outcome	Pass / Fail

[34] https://www.unece.org/fileadmin/DAM/trans/danger/publi/unrec/rev17/English/Rev17_Volume1.pdf

3.3.5.9.7 UC02 with dangerous goods

ID	T5.2
Test	Tests of UC01 incremented with a check whether dangerous goods are being transported.
Type	System testing
Setup	<Describe the needed setup, tools, hooks and probes needed for this test>
Start	<Describe the system state for the start of the test>
Req.	<Define the requirements involved in [x], format>
Input	<Define the test input steps, e.g. Inject data x in interface y>
Output	<Define the expected result, e.g. Expected to receive event z>
Logs	<Define where the output log is stored, e.g.: Folder "`Tx_Output`", prefix "`Tx.y.1_logname`" >
Outcome	Pass / Fail

3.3.5.9.8 UC03 with dangerous goods

ID	T5.3
Test	Tests of UC03 incremented with a check whether dangerous goods are being transported.
Type	System testing
Setup	<Describe the needed setup, tools, hooks and probes needed for this test>
Start	<Describe the system state for the start of the test>
Req.	<Define the requirements involved in [x], format>
Input	<Define the test input steps, e.g. Inject data x in interface y>
Output	<Define the expected result, e.g. Expected to receive event z>
Logs	<Define where the output log is stored, e.g.: Folder "`Tx_Output`", prefix "`Tx.y.1_logname`" >
Outcome	Pass / Fail

3.3.5.9.9 UC04 with dangerous goods

ID	T5.4
Test	Tests of UC04 incremented with a check whether dangerous goods are being transported.
Type	System testing
Setup	<Describe the needed setup, tools, hooks and probes needed for this test>
Start	<Describe the system state for the start of the test>
Req.	<Define the requirements involved in [x], format>
Input	<Define the test input steps, e.g. Inject data x in interface y>
Output	<Define the expected result, e.g. Expected to receive event z>
Logs	<Define where the output log is stored, e.g.: Folder "`Tx_Output`", prefix "`Tx.y.1_logname`" >
Outcome	Pass / Fail

3.3.5.10 Non-functional tests

The tests will include measuring semantic interoperability, availability, logging, input and output messaging capabilities of the EWS. Measurements include the total transaction time, CPU processing and memory consumption history (of the EWS components, e.g. application and CEP servers). Moreover, it will provide thresholds related to overload of resources, which can be used as input for distribution deployment calculations (e.g. multi-broker among federated clouds).

Each test case has an equivalent input and output data file, named TX.Y.json (input and output folders).

This use case involves these requirements: [6], [20], [23], [72], [180], [186] and [249].

Semantic interoperability: semantic loss

ID	T6.3
Test	Since the INTER-IoT approach is based on multiple semantic translations, semantic loss will be a variable to calculate semantic interoperability, measured by executing the transformations in sequence from ontology A (e.g. SAREF) to ontology B (e.g. SSN) and from B to A, i.e. check how x is different to $T(T(x)_{A>B})_{B>A}$, where $T(x)_{A>B}$ represents the semantic translation function from A to B [12].
Type	System testing

Setup	<Describe the needed setup, tools, hooks and probes needed for this test>
Start	<Describe the system state for the start of the test>
Req.	<Define the requirements involved in [x], format>
Input	<Define the test input steps, e.g. Inject data x in interface y>
Output	<Define the expected result, e.g. Expected to receive event z>
Logs	<Define where the output log is stored, e.g.: Folder "`Tx_Output`", prefix "`Tx.y.1_logname`" >
Outcome	Pass / Fail

Performance: load testing

ID	T6.1
Test	<load test is usually conducted to understand the behaviour of the system under a specific expected load. This load can be the expected concurrent number of users on the application performing a specific number of transactions within the set duration. This test will give out the response times of all the important business critical transactions. The database, application server, etc. are also monitored during the test, this will assist in identifying bottlenecks in the application software and the hardware that the software is installed on.>
Type	System testing
Setup	<Describe the needed setup, tools, hooks and probes needed for this test>
Start	<Describe the system state for the start of the test>
Req.	<Define the requirements involved in [x], format>
Input	<Define the test input steps, e.g. Inject data x in interface y>
Output	<Define the expected result, e.g. Expected to receive event z>
Logs	<Define where the output log is stored, e.g.: Folder "`Tx_Output`", prefix "`Tx.y.1_logname`" >
Outcome	Pass / Fail

Performance: stress testing

ID	T6.2

Test	<understand the upper limits of capacity within the system. This kind of test is done to determine the system's robustness in terms of extreme load and helps application administrators to determine if the system will perform sufficiently if the current load goes well above the expected maximum.>
Type	System testing
Setup	<Describe the needed setup, tools, hooks and probes needed for this test>
Start	<Describe the system state for the start of the test>
Req.	<Define the requirements involved in [x], format>
Input	<Define the test input steps, e.g. Inject data x in interface y>
Output	<Define the expected result, e.g. Expected to receive event z>
Logs	<Define where the output log is stored, e.g.: Folder "`Tx_Output`", prefix "`Tx.y.1_logname`" >
Outcome	Pass / Fail

Performance: soak/endurance testing

ID	T6.4
Test	<endurance testing, is usually done to determine if the system can sustain the continuous expected load. During soak tests, memory utilization is monitored to detect potential leaks. Also important, but often overlooked is performance degradation, i.e. to ensure that the throughput and/or response times after some long period of sustained activity are as good as or better than at the beginning of the test. It essentially involves applying a significant load to a system for an extended, significant period of time. The goal is to discover how the system behaves under sustained use.>
Type	System testing
Setup	<Describe the needed setup, tools, hooks and probes needed for this test>
Start	<Describe the system state for the start of the test>
Req.	<Define the requirements involved in [x], format>
Input	<Define the test input steps, e.g. Inject data x in interface y>
Output	<Define the expected result, e.g. Expected to receive event z>
Logs	<Define where the output log is stored, e.g.: Folder "`Tx_Output`", prefix "`Tx.y.1_logname`" >
Outcome	Pass / Fail

Logging tests

ID	T6.5
Test	Check whether the logging capability of the EWS works, which might be spread in the involved servers (TS_01 components)
Type	Auditing test
Setup	<Describe the needed setup, tools, hooks and probes needed for this test>
Start	<Describe the system state for the start of the test>
Req.	<Define the requirements involved in [x], format>
Input	<Define the test input steps, e.g. Inject data x in interface y>
Output	<Define the expected result, e.g. Expected to receive event z>
Logs	<Define where the output log is stored, e.g.: Folder "`Tx_Output`", prefix "`Tx.y.1_logname`" >
Outcome	Pass / Fail

3.3.5.11 Suggested: integration tests

Since device, network and middleware layers are out of the scope of the INTER-IoT-EWS validation, we list some suggestions here top enable health and logistics data to be provided according to the INTER-IoT framework.

Execute all test cases but instead of using test data from files, acquire data from IoT platforms (including network and device layers) and publishing data in IoT platform(s), i.e. Port Authority Mosquitto broker.

A use case that enables the INTER-IoT application layer (INTER-IoT-EWS) to consume data provided by the two involved platforms and their respective devices. As output of this use case it is expected that the device, network and middleware are able to provide data through a pub/sub approach (INTER-MW with IPSM), where the application layer can receive real-time data (by subscribing to topics in a broker) formatted as JSON-LD messages in accordance to INTER-IoT ontology patterns.

This use case involves these requirements: [180] and [249].

Acquire health data

ID	T1.1
Test	Vehicle collision detected with smartphone accelerometer data.
Type	System testing
Setup	Need test setup TS_01

Start	Vehicle is in one of the port gates (to enter the port area)
Req.	[X], [Y]
Input	<Define the test input steps, e.g. Inject data x in interface y>
Output	<Define the expected result, e.g. Expected to receive event z>
Logs	<Define where the output log is stored, e.g.: Folder "`Tx_Output`", prefix "`Tx.y.1_logname`" >
Outcome	Pass / Fail

Acquire logistics data

ID	**T1.1**
Test	Vehicle collision detected with smartphone accelerometer data.
Type	System testing
Setup	Need test setup TS_01
Start	Vehicle is in one of the port gates (to enter the port area)
Req.	[X], [Y]
Input	<Define the test input steps, e.g. Inject data x in interface y>
Output	<Define the expected result, e.g. Expected to receive event z>
Logs	<Define where the output log is stored, e.g.: Folder "`Tx_Output`", prefix "`Tx.y.1_logname`" >
Outcome	Pass / Fail

Publish alerts in the Port Authority IoT platform

ID	**T1.1**
Test	Execution of functional test cases (T1-T5) in Mosquitto broker.
Type	System testing
Setup	Need test setup TS_01
Start	Vehicle is in one of the port gates (to enter the port area)
Req.	[X], [Y]
Input	<Define the test input steps, e.g. Inject data x in interface y>

Output	<Define the expected result, e.g. Expected to receive event z>
Logs	<Define where the output log is stored, e.g.: Folder "`Tx_Output`", prefix "`Tx.y.1_logname`" >
Outcome	Pass / Fail

3.3.5.12 Test outcome overview

Test outcome

The following table will provide an overview of the test result of all the performed tests in this FAT.

Test	Description	Outcome
T1.1	Vehicle collision detected with smartphone accelerometer	Pass / Fail
T1.2	Vehicle collision detected with medical wearable accelerometer	Pass / Fail
T1.3	Vehicle collision detected with smartphone and medical wearable accelerometer	Pass / Fail
T1.4	Vehicle collision detected with smartphone or medical wearable accelerometer	Pass / Fail
T1.5	Vehicle collision detected according to T1.1 with 4 classifications of severity + urgency	Pass / Fail
T1.6	Vehicle collision detected according to T1.2 with 4 classifications of severity + urgency	Pass / Fail
T1.7	Vehicle collision detected according to T1.3 with 4 classifications of severity + urgency	Pass / Fail
T1.8	Vehicle collision detected according to T1.4 with 4 classifications of severity + urgency	Pass / Fail
T2.1	Bradycardia detected with fixed threshold	Pass / Fail
T2.2	Bradycardia detected with dynamic threshold	Pass / Fail
T2.3	Tachycardia detected with fixed threshold	Pass / Fail
T2.4	Tachycardia detected with dynamic threshold	Pass / Fail
T2.5	Multiple occurrences of bradycardia detected with fixed threshold	Pass / Fail
T2.6	Multiple occurrences of bradycardia detected with dynamic threshold	Pass / Fail
T2.7	Multiple occurrences of tachycardia detected with fixed threshold	Pass / Fail
T2.8	Multiple occurrences of tachycardia detected with dynamic threshold	Pass / Fail

T2.9	Large variation of heart rate	Pass / Fail
T2.10	Detect high level of stress	Pass / Fail
T3.1	Vehicle collision followed by bradycardia	Pass / Fail
T4.1	Truck on opposite direction of a street within the port	Pass / Fail
T5.1	UC01 with dangerous goods	Pass / Fail
T5.2	UC02 with dangerous goods	Pass / Fail
T5.3	UC03 with dangerous goods	Pass / Fail
T5.4	UC04 with dangerous goods	Pass / Fail
T6.1	Semantic interoperability: semantic loss	Pass / Fail
T6.2	Performance: load testing	Pass / Fail
T6.3	Performance: stress testing	Pass / Fail
T6.4	Performance: soak/endurance testing	Pass / Fail
T6.5	Logging tests	Pass / Fail
FAT Outcome		**Pass / Fail**

Table 77: Test outcome overview

3.3.5.13 Integration ethics and security

Introduction

During the Ethical Review it was requested that ethics had to be included within the project scope. This section addresses recommendations and proposed solutions to the ethical and security points of the integration.

For each pilot the ethics is discussed in paragraphs 8.2 until 8.5. The security aspects of each layer is discussed in paragraph 8.7 and 8.8.

The information for the pilots for both ethics and security comes from the partners and may be included in other documents as well.

Privacy issues of personal data from INTER-IoT-EWS

Health data (ECG) generated for the FAT will be based on public simulated data made available by the ECG device manufacturer (Shimmer) and based on data collected from the U.Twente researcher involved/responsible by this project (Joao Moreira). This researcher will sign a consent form enabling the data generated to be public.

Data about truck position, speed, acceleration, transported goods, haulier company and drivers' information will be generated through simulations. These data will be based on fictional information, avoiding any links to existing people (besides the researcher above) and companies.

Although the FAT do not include the privacy issues identified regarding health-related data and personal information, the SAT should consider the classification of the data according to the INTER-IoT data management plan (D8.4_INTER-IoT_Data_Management_Plan.pdf) in section 2.3. Therefore, after the FAT, data should be classified within the four levels of protection and the test environment changed to address the constraints of this table. For example, information of truck driver will be classified as protection level 2 (high business impact).

3.3.6 Third Party: Senshook

Smart Mosquito Trap

The port of Valencia is one of the most important hubs in the world and thus a critical point of entry of invasive species that must be monitored, according to the European Centre of Disease Control.

The pilot will consist on deploying a surveillance network of 5 observation static IoT nodes in critical points of the port of Valencia.

Each node is composed of a Smart Mosquito Trap capable of mimicking the human body (scent and respiration) and of automatically counting captured mosquitoes, identify the gender and the species. The information collected by each node is then sent to a server.

The pilot will start in May-June 2018 and will last until October 2018. This corresponds exactly with the period of the year when disease-vector mosquitoes are active and must be monitored.

Following is a diagram that gives a high-level overview of the system. When a mosquito enters the trap it gets detected by the sensor which also captures the necessary data to identify it.

The sensor is connected to a Senscape board which sends the information to the server running SensHook.

A client can retrieve the gathered information.

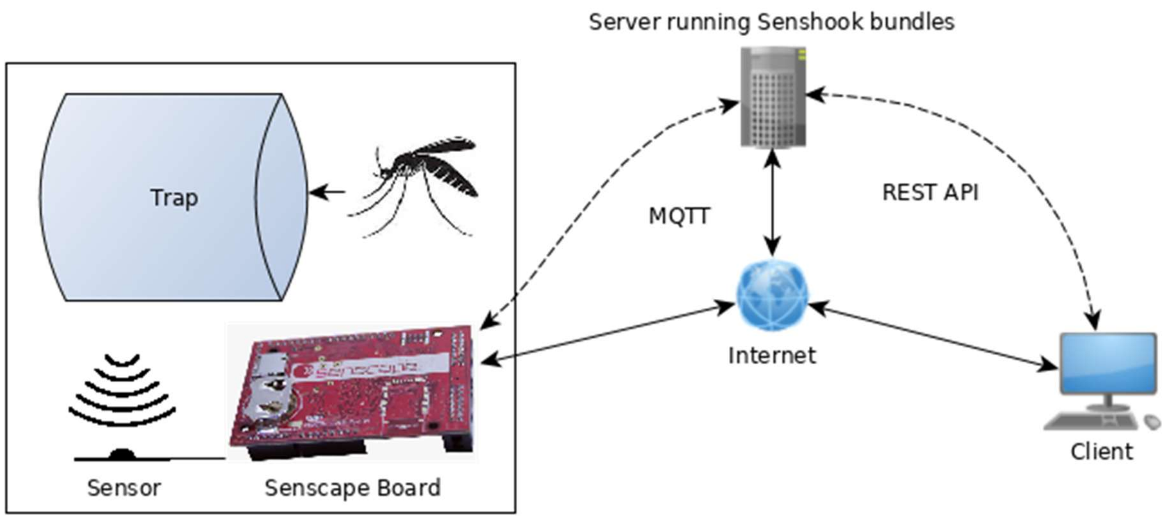


Figure 104: Mosquitto trap system architecture.

Objectives of the project

The specific objectives of the project are to:

- Perform a technical feasibility assessment of the Senshook solutions as part of the INTER-IoT project
- Implement Senshook according to INTER-IoT requirements
- Carry out a series of tests/pilots to evaluate the performance and benefits of the tool.

Collaboration approach

Irideon will contribute to the INTER-IoT project by providing a new open tool for the INTER-LAYER building block, which will allow the evolution of products based on INTER-IoT, but at the same time will allow us to evolve our products in order to add new interoperability features.

By contributing to the development of INTER-IoT, Irideon will be able to address new IoT scenarios in which different IoT platforms, apart from those based on Senscape, are involved, and also in those in which more than one application domain is addressed.

Senshook Arquitecture Overview

The system consists of the Senscape hardware and a D2D virtual gateway which provides connection to the middleware platform.

Following is a description of the different components of the gateway.

Dispatcher

The central part of the virtual gateway is the dispatcher, it consists of a communication layer which connects to the hardware via MQTT and a service layer which implements the IEEE 1451 standard.

IEEE 1451 is a set of smart transducer interface standards developed by the Institute of Electrical and Electronics Engineers (IEEE).

Irideon has developed a specific lightweight implementation of this standard for the Senscape hardware devices.

This implementation includes communication protocols, transducer electronic data sheet (TEDS) and common functions.

The dispatcher is implemented in two OSGi bundles:

- Dispatcher API bundle: This bundle defines and exports the interface.
- Dispatcher provider bundle: This bundle implements the service layer, which includes the IEEE 1451 standard and the communication layer with the MQTT connector. Furthermore, it registers a service with the dispatcher API.

Middleware Controller

The middleware controller consists of the three bundles:

- Middleware controller API Bundle: This bundle defines the OSGi interface for the bundle and exports it as a package.
- Middleware Controller Provider Bundle: This bundle implements the middleware controller and provides it as an OSGi service with the middleware controller API.
- Middleware Controller Application Bundle: This bundle consumes the service published by the provider and implements an application that exposes a REST API.

Measure Storage

The measure storage has two bundles:

- Measure storage API Bundle: This bundle defines the interface for the bundle and exports it as a package.

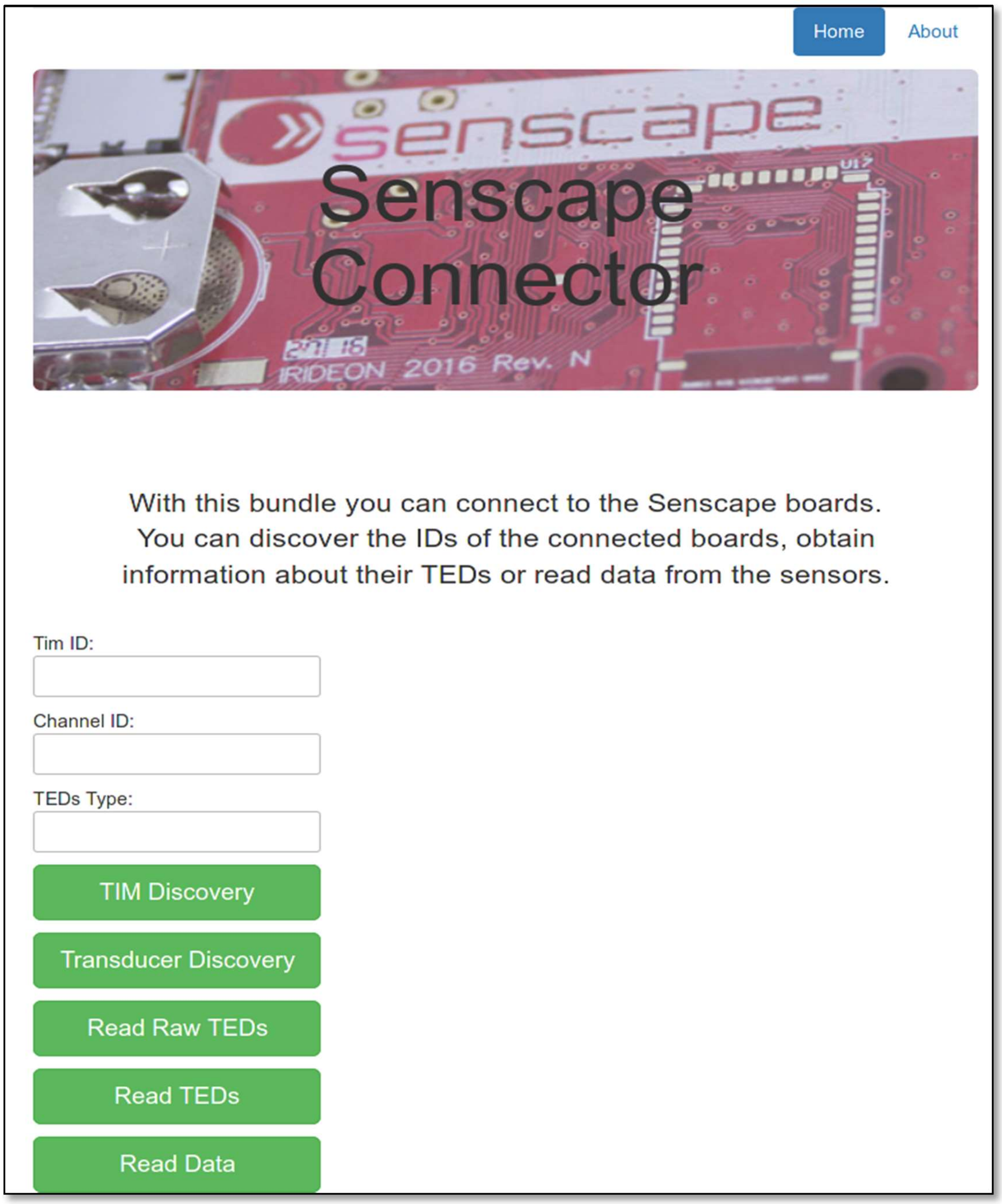
- Measure storage provider bundle: This bundle provides the connection to a local database to store and retrieve the values sent and requested by the dispatcher. This bundle implements the interface defined by the API bundle and registers it as a service.

<u>API / Web Application</u>

This component provides a REST API to the dispatcher and also includes a web application which serves as a front-end to interact with the Senscape hardware.

It consists of one bundle which is an OSGi application that consumes the service offered by the dispatcher.

In Figure 105 you can see a screenshot of the web front-end:



With this bundle you can connect to the Senscape boards. You can discover the IDs of the connected boards, obtain information about their TEDs or read data from the sensors.

Tim ID:

Channel ID:

TEDs Type:

Figure 105: Web Application of Senscape Connector.

And in Figure 106, in shown a diagram of the virtual gateway with their components and respective bundles:

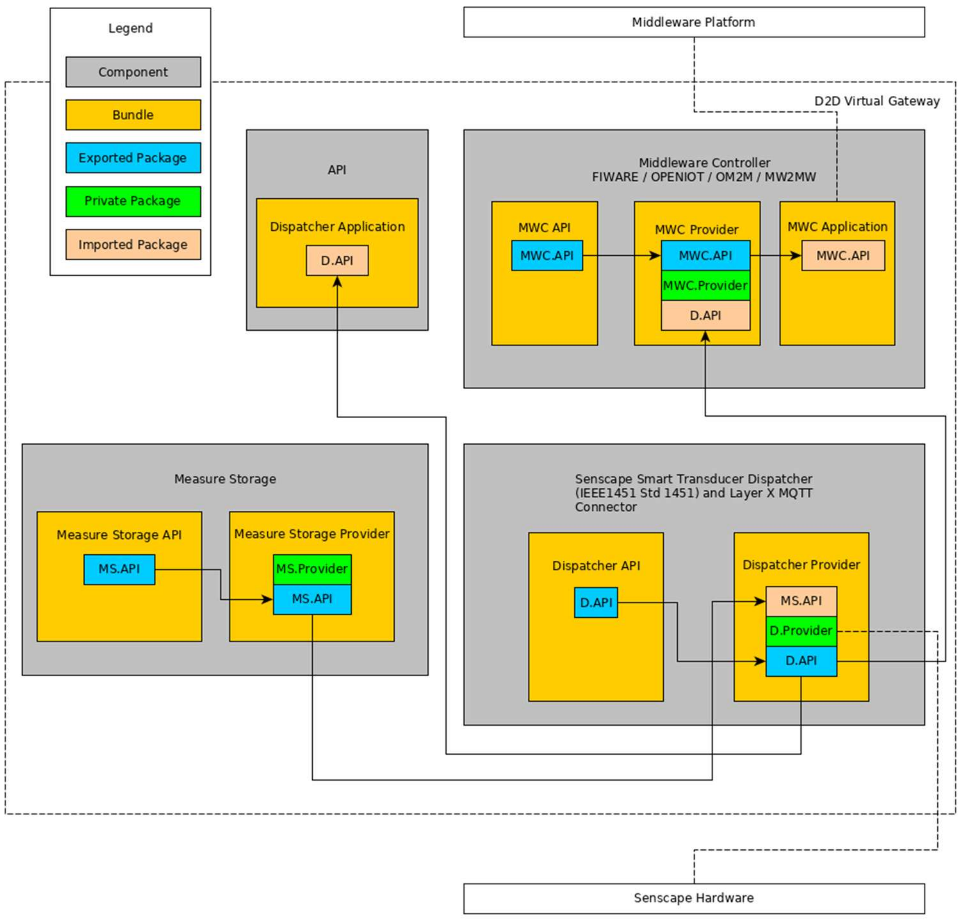


Figure 106: Virtual gateway bundles architecture.

System interfaces

The virtual gateway offers interfaces to two of its components, the middleware controller and the dispatcher.

<u>Dispatcher</u>

The dispatcher comes with four interfaces:

- Terminal interface: This interface features an Apache Felix Gogo Shell. It is thought for debugging.
- OSGi interface: The component registers an OSGi service API which can be used to incorporate the dispatcher in a modular OSGi application.
- REST API: The dispatcher also offers a REST API, so it is possible to communicate with it over the Internet making it possible to use it in a web application.
- Web front-end: Serves for testing and demonstration.

Middleware Controller

The middleware controller comes with four interfaces:

- Terminal interface: This interface features a gogo shell. It is thought for debugging.

- OSGi interface: The component registers a OSGi service API which can be used to incorporate the middleware in a modular OSGi application.
- REST API: The middleware controller also offers a REST API, so it is possible to communicate with the component over the Internet making it possible to use it in a web application.

3.3.6.1 Deliverables and version overview

The following table contains a deliverable list which need to be signed of before FAT testing commences.

ID	Description	Check
Documents		
1	FAT Document	
Hardware		
2	Senscape Mosquito Trap	
3	Computer with Internet access	
4	Server running the Senshook bundles	
Tools		
5	Eclipse	
6	JUnit	
7	Postman	

Table 78: Deliverable checklist

The following table shows the software components and version of which the system release version SensHook consists of.

ID	Description	Version	Check
IoT Virtual Gateway			
1	Dispatcher API Bundle	V1.0.0	
2	Dispatcher Provider Bundle	V1.0.0	
3	Dispatcher Application Bundle	V1.0.0	
4	Measure Storage API Bundle	V1.0.0	
5	Measure Storage Provider Bundle	V1.0.0	

Table 79: Component version overview

3.3.6.2 Requirements, scenarios and use cases

The following table provides an overview of the relation between the requirements and the test(s) that validate their implementation.

ID	Description	Covered by
Communications		
14	Platform independent	T15.1.2, T15.1.3, T15.2.2, T15.2.3, T15.3.2, T15.3.3, T15.5.2
15	Common IoT communication protocols must be supported.	T15.1.1, T15.1.2, T15.1.3, T15.2.1, T15.2.2, T15.2.3, T15.3.1, T15.3.2

39	Gateway capabilities	T15.1.2, T15.1.3, T15.2.2, T15.2.3, T15.3.2, T15.3.3, T15.5.2,T15.5.2
153	System cache in gateways and upper levels	T15.5.1, T15.5.2
Functionality		
21	Real time output	T15.1.1, T15.1.2, T15.1.3, T15.2.1, T15.2.2, T15.2.3, T15.3.1, T15.3.2, T15.3.3
26	Remote device control	T15.1.2, T15.1.3, T15.2.2, T15.2.3, T15.3.2, T15.3.3
API		
243	Gateway access API	T15.1.1, T15.1.2,T15.1.3, T15.2.1, T15.2.2, T15.2.3, T15.3.1, T15.5.1
Interoperability		
93	Standard protocol for the device communications	T15.1.1, T15.1.2, T15.1.3, T15.2.1, T15.2.2, T15.2.3, T15.3.1, T15.3.2, T15.3.3
226	API for network services	T15.1.2, T15.2.2, T15.2.3, T15.3.2, T15.5.2
Middleware		
234	Provide connectors to middleware standards	*Not implemented because we do not have information about the middleware layer*
Operational		
57	Device monitoring and self-awareness of the system	T15.1.1, T15.2.1, T15.3.1
Virtualization		
244	Gateway virtualization	T15.1.2, T15.1.3, T15.2.2, T15.2.3, T15.3.2, T15.3.3, T15.5.2

Table 80: Requirements vs test mapping

The following table provides an overview of the relation between the scenarios and the test(s) that validate their implementation.

ID	Scenario name	Covered by
15	Surveillance systems for prevention programs	T15.5.1, T15.5.2

Table 81: Scenario vs test mapping

3.3.6.3 Test environment

Introduction

To test the functionality of the SensHook in combination with the IoT framework a representative test system is needed. The test system needs to approach the “real world” as much as possible. The pilot setups must be recreated and proven. This chapter will describe this environment and the used hardware, software, tools and platforms.

Test environment

This paragraph describes the test environment and the complete system setup used during this FAT.

Test setups, tools, hooks and probes

This paragraph describes the test setups, tools, hooks and probes used in the by this document defined tests.

A test setup defines a setup used for a test, and describe the used system parts, interface etc.

Tools define the used COTS tooling that will be used during testing e.g. Logic analyzers, packet sniffer like e.g. Whireshark, developed scripts, etc.

Hooks are used to inject data or control parts of the system used to get the system to handle a needed scenario. This can be hidden, debug or service interfaces which contain the needed functionality.

Probes are used to gather logging from parts of the system under test to provide system feedback and to prove that the system is handling the scenario as it should.

The following paragraphs will define the used setups, tools, hooks and probes which will be referred to by the test descriptions.

TS_01 Unit Test Setup

The unit testing used for testing the Java/OSGI API is done with the IDE Eclipse and the framework JUnit.

TS_02 REST API Test setup

For REST API testing we use the following setup. On one hand there is a client which executes the tests, on the other is the Senscape device and in between there is a server running the bundle. The connection between client and server uses HTTP and between server and the Senscape board it is established via MQTT.

TS_03 Web Interface Front-End testing

To test the web interface, we need a computer with a browser with access to the web interface.

In Figure 107 you can see a diagram of the test setup:

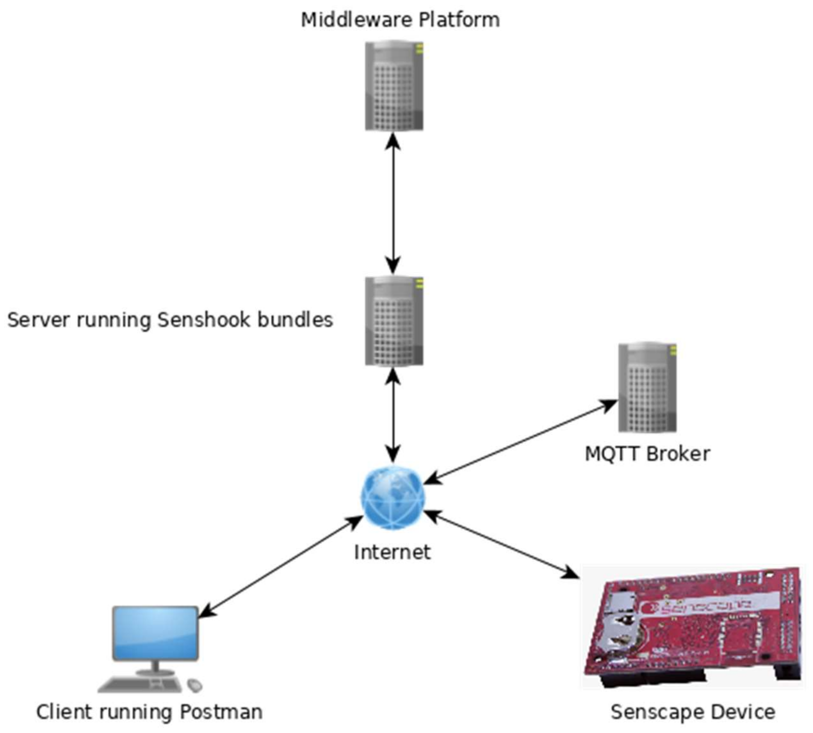


Figure 107: Diagram of the Web Interface Front-end testing.

TT_01 JUnit

JUnit is a framework for unit testing in Java, it will be used for the testing of the Java/OSGI API tests.

TT_02 Eclipse

Eclipse is an IDE widely used for developing in Java.

TT_03 Postman

Postman is a free API development environment. It offers the option to write complete tests for a REST API, so we use it to do status code validation, data type validation, etc.

TH_01 JUnit test script

For the unit tests the JUnit test script is responsible for setting up the context of the different tests.

TH_02 Postman test script

For the REST API tests the postman script is responsible for setting up the context of the differents tests

TP_01 JUnit test script

For the unit tests the JUnit test script is responsible for writing the logs of the test results.

3.3.6.4 Test description

3.3.6.4.1 Scenario 15 Surveillance systems for prevention programs

Non-native species cost the EU €12 billion per year in damage and control costs. In the last decades several species of disease carrying mosquitoes have invaded Europe through the transport of goods, increasing international travel and climate change.

SensHook will reduce inspection costs and improves surveillance programs. With our new electronic trap, we will be the first in the world to combine human mimicking with automatic pest information in their value proposition. This allows a whole new population of consumers to establish surveillance programs that were only accessible to those with significant resources.

The demonstration of the Senshook architecture will be based on JAVA bundles that makes up the Virtual Gateway of SensHook. The virtual gateway will be designed to enable bi-directional communication between the middleware platform and the Virtual Gateway and between the Virtual Gateway and Traps. The Virtual Gateway will be developed for receiving data from the traps in the field, and database architecture will be defined for the storage of data gathered by the traps. This software module will be able to read data received from the field and store it, after pre-processing it, in the system database. This module will also incorporate the ability to perform system checks, detect failures in nodes and sensors, troubleshooting guide, auto-calibration and to incorporate “plug and work” devices.

3.3.6.4.2 Connecting and detecting devices

Connecting and detecting Senscape devices. Currently there are three interfaces from which it is possible to achieve this, a Java/OSGI API, a REST API and a web interface.

T15.1.1 Device identification over Dispatcher API

ID	T15.1.1
Test	Device Identification over Dispatcher API
Type	System Testing
Setup	TT_01, TT_02, TS_01, TH_01, TP_01
Start	Device is not yet connected.
Req.	[243], [57], [21], [93], [15]
Input	• Connect device • Perform a call to the Dispatcher API method ‘discoverTims’
Output	String containing the device ID
Logs	log/T15-1-1.log
Outcome	Pass / Fail

T15.1.2 Device identification over REST API

ID	T15.1.2
Test	Device Identification over REST API
Type	System Testing
Setup	TT_03, TS_02, TH_02
Start	Device is not yet connected.
Req.	[21], [26], [243], [93], [226], [14], [15], [39], [244]
Input	• Connect device

	• Perform a call to the REST API method 'discoverTims'
Output	String containing the device ID
Logs	-
Outcome	Pass / Fail

T15.1.3 Device identification over Web Interface

ID	T15.1.3
Test	Device Identification over Web Interface
Type	System Testing
Setup	<Describe the needed setup, tools, hooks and probes needed for this test>
Start	TS_03
Req.	[21], [26], [243], [93], [226], [14], [15], [39], [244]
Input	• Connect device • Click 'TIM Discovery' Button
Output	String containing the device ID
Logs	-
Outcome	Pass / Fail

3.3.6.4.3 Obtain information about connected sensors

Information about the connected sensors to the different Senscape devices is retrieved. Senscape implements the IEEE 1451.4 Transducer Electronic Data Sheets (TEDS) standard. Besides the ability to retrieve detailed information about the connected sensors it also provides plug and play functionality for sensors.

Currently there are three interfaces from which it is possible to read the TEDS, a Java/OSGI API, a REST API and a web interface.

T15.2.1 Read TEDs over Dispatcher API

ID	T15.2.1
Test	Read TEDs over Dispatcher API
Type	System Testing
Setup	TT_01, TT_02, TS_01, TH_01, TP_01
Start	Device connected
Req.	[243], [57], [21], [93], [15]
Input	• Perform a call to the Dispatcher API method 'readTeds
Output	String containing the TEDs
Logs	log/T15-2-1.log
Outcome	Pass / Fail

T15.2.2 Read TEDs over REST API

ID	T15.2.2
Test	Read TEDs over REST API
Type	System Testing
Setup	TT_03, TS_02, TH_02
Start	Device connected
Req.	[21], [26], [243], [93], [226], [14], [15], [39], [244]

Input	• Perform a call to the REST API method ‘readTeds’
Output	String containing the TEDs
Logs	-
Outcome	Pass / Fail

T15.2.3 Read TEDs over Web Interface

ID	**T15.2.3**
Test	Read TEDs over Web Interface
Type	System Testing
Setup	TS_03
Start	Device connected
Req.	[21], [26], [243], [93], [226], [14], [15], [39], [244]
Input	• Fill out fields: TIM Id, Channel Id, TEDs type • Click on ‘Read TEDs’ button.
Output	String containing the TEDs
Logs	-
Outcome	Pass / Fail

3.3.6.4.4 Read sensor

The current value of a connected sensor is read. Currently there are three interfaces from which it is possible to read the sensors, a Java/OSGI API, a REST API and a web interface.

T15.3.1 Read sensor data over Dispatcher API

ID	**T15.3.1**
Test	Read sensor data over Dispatcher API
Type	System Testing
Setup	TT_01, TT_02, TS_01, TH_01, TP_01
Start	Device connected
Req.	[243], [57], [21], [93], [15]
Input	• Perform a call to the Dispatcher API method ‘readData’
Output	String containing the current value of the sensor
Logs	log/15-3-1.log
Outcome	Pass / Fail

T15.3.2 Read sensor data over REST API

ID	**T15.3.2**
Test	Read sensor data over REST API
Type	System Testing
Setup	TT_03, TS_02, TH_02
Start	[21], [26], [243], [93], [226], [14], [15], [39], [244]
Req.	<Define the requirements involved in [x], format>
Input	• Perform a call to the REST API method ‘data’String containing the current value of the sensor
Output	String containing the current value of the sensor.
Logs	-
Outcome	Pass / Fail

T15.3.3 Read sensor data over Web Interface

ID	T15.3.3
Test	Read sensor data over Web Interface
Type	System Testing
Setup	TS_03
Start	Device connected
Req.	[21], [26], [243], [93], [226], [14], [15], [39], [244]
Input	• Fill out TIM Id and Channel Id field. • Click on 'Read Data' button.
Output	String containing the current value of the sensor.
Logs	-
Outcome	Pass / Fail

3.3.6.4.5 Send mosquito flight data to Data Storage

A mosquito entered the trap and the flight is detected and sent to the data storage. This can be done via the Java/OSGI API.

T15.4.1 Send mosquito flight data over Dispatcher API

ID	T15.4.1
Test	TT_01, TT_02, TS_01, TH_01, TP_01
Type	System Testing
Setup	<Describe the needed setup, tools, hooks and probes needed for this test>
Start	• Device connected • Flight data extracted
Req.	[15], [153], [243], [93]
Input	• Perform a call to the Dispatcher API method 'writeDataDb'
Output	-
Logs	log/15-4-1.log
Outcome	Pass / Fail

3.3.6.4.6 Retrieve flight data from the data base

The mosquito flight data is retreived from the data base for further use. This can be done via the Java/OSGI API and the REST API.

T15.5.1 Retrieve flight data from the data base over Dispatcher API

ID	T15.5.1
Test	Retrieve flight data from the data base over Dispatcher API
Type	System Testing
Setup	TT_01, TT_02, TS_01, TH_01, TP_01
Start	Device connected
Req.	[153], [243]
Input	• Perform a call to the Dispatcher API method 'readLastValuesDb'
Output	Flight data results
Logs	Log/15-5-1.log
Outcome	Pass / Fail

T15.5.2 Retrieve flight data from the data base over REST API

ID	T15.5.2
Test	Retrieve flight data from the data base over REST API
Type	System Testing
Setup	TT_03, TS_02, TH_02
Start	Device connected
Req.	[14], [39], [153], [226], [244], [39], [244]
Input	• Perform a call to the REST API method 'readLastValuesDb'Flight data results
Output	Flight data results
Logs	-
Outcome	Pass / Fail

3.3.6.5 Test outcome overview

Test outcome

The following table will provide an overview of the test result of all the performed tests in this FAT.

Test	Description	Outcome
T15.1.1	Device identification over Dispatcher API	Pass / Fail
T15.1.2	Device identification over REST API	Pass / Fail
T15.1.3	Device identification over Web Interface	Pass / Fail
T15.2.1	Read TEDs over Dispatcher API	Pass / Fail
T15.2.2	Read TEDs over REST API	Pass / Fail
T15.2.3	Read TEDs over Web Interface	Pass / Fail
T15.3.1	Read sensor data over Dispatcher API	Pass / Fail
T15.3.2	Read sensor data over REST API	Pass / Fail
T15.3.3	Read sensor data over Web Interface	Pass / Fail
T15.4.1	Send mosquito flight data over Dispatcher API	Pass / Fail
T15.5.1	Retrieve flight data from the data base over Dispatcher API	Pass / Fail
T15.5.2	Retrieve flight data from the data base over REST API	Pass / Fail
FAT Outcome		**Pass / Fail**

Table 82: Test outcome overview

3.3.6.6 Integration ethics and security

Introduction

During the Ethical Review it was requested that ethics had to be included within the project scope. This section addresses recommendations and proposed solutions to the ethical and security points of the integration.

For each pilot the ethics is discussed in paragraphs 8.2 until 8.5. The security aspects of each layer is discussed in paragraph 8.7 and 8.8.

The information for the pilots for both ethics and security comes from the partners and may be included in other documents as well.

Smart Mosquito Trap

There are at least 3,528 species of mosquitoes. The majority are harmless to humans, but a few dozen species transmit diseases. SensHook addresses the major problem of disease-carrying Invasive Mosquito Species (IMS) that invade Europe due to climate change. The most threatening is the Asian Tiger Mosquito (Aedes albopictus), a vector that can transmit several serious and life threatening diseases.

3.3.7 Third Party: SOFOS

The imminent arrival of the Internet of Things (IoT), which consists of a vast variety of devices with heterogeneous characteristics, means that future networks need a new architecture to accommodate end-to-end IoT networking, dealing with: i) the expected increase in data generation, ii) the problems related to the end-to-end IP networking of the resource-constrained IoT devices, iii) the capacity mismatch between devices, and iv) the rapid interaction between services and infrastructure.

Software defined networking (SDN) and network function virtualization (NFV) are two technologies that promise to cost-effectively provide the scale and versatility necessary for IoT services in order to address efficiently the aforementioned challenges. Moreover, given that SDN and NFV are considered a fundamental component in the 5G landscape, since it is widely recognized that 5G networks will be software-driven and most components of future heterogeneous 5G architectures should be capable to support software-network technologies, both SDN and NFV are promising candidate technologies for a Software Defined Approach of end-to-end IoT Networking.

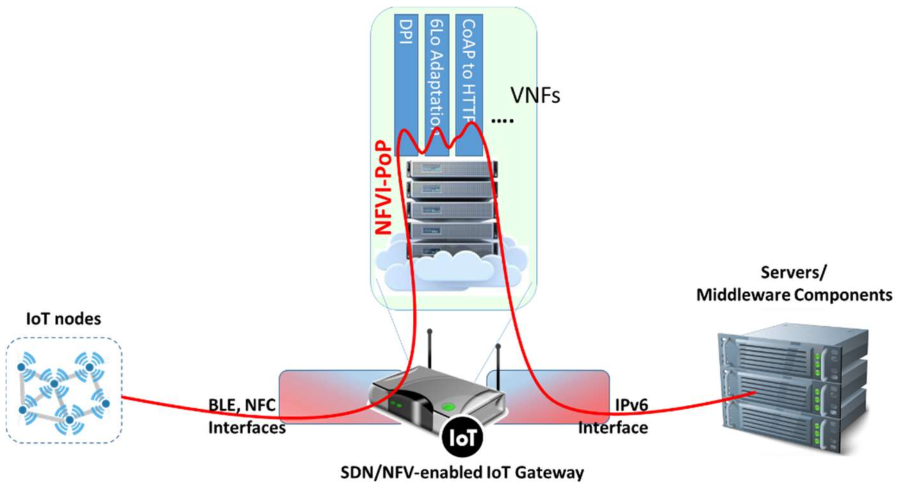


Figure 108: The proposed SDN/NFV end-to-end IoT Gateway overview

SOFOS aims at advancing the existing INTER-IoT framework with SDN and NFV functionalities towards a Software-defined end-to-end IoT infrastructure with IoT service chaining support. The main objective of the proposed SDN/NFV-enabled framework is to enhance the interoperability of the INTER-IoT framework in order to facilitate the interoperable management of a large number of diverse smart objects that currently operate utilizing a variety of different IoT protocols.

In this framework, specific objectives of the proposal include:

- To add SDN/NFV Automation and Verification in IoT Infrastructure
- To relocate various IoT functions from HW appliances to Virtual Machines (VMs) (i.e. Virtual Network Functions - VNFs).
- To enhance the interoperability support of the INTER-IoT platform by deploying VNFs that map IoT protocols (such as CoAP, MQTT) to standard IP networking
- To connect and chain the software-defined IoT functions (i.e. VNFs) together.
- To abstract the IoT's control plane by exploiting the SDN concept and advances.
- Inter-IoT Infrastructure with the proposed advances can be enhanced by means of NFV with integration of SDN, making it more agile and introducing a high degree of automation in service delivery and operation—from dynamic IoT service parameter exposure and negotiation to resource allocation, service fulfillment, and assurance.

3.3.7.1 Integration of IoT framework

SOFOS experiment considers that INFOLYSiS will deploy on top of INTER-IoT vGW modules that provide SDN/NFV Automation in IoT Infrastructure, such as the INFOLYSiS SDN/NFV Network Manager. By applying appropriate OPENFLOW commands, INFOLYSiS add-on will steer the data traffic from the INTER-IoT vGW to the various VNFs that will have been deployed in order to enhance the interoperability functions of INTER-IoT, allowing to the application layer to represent the received data in a unified way.

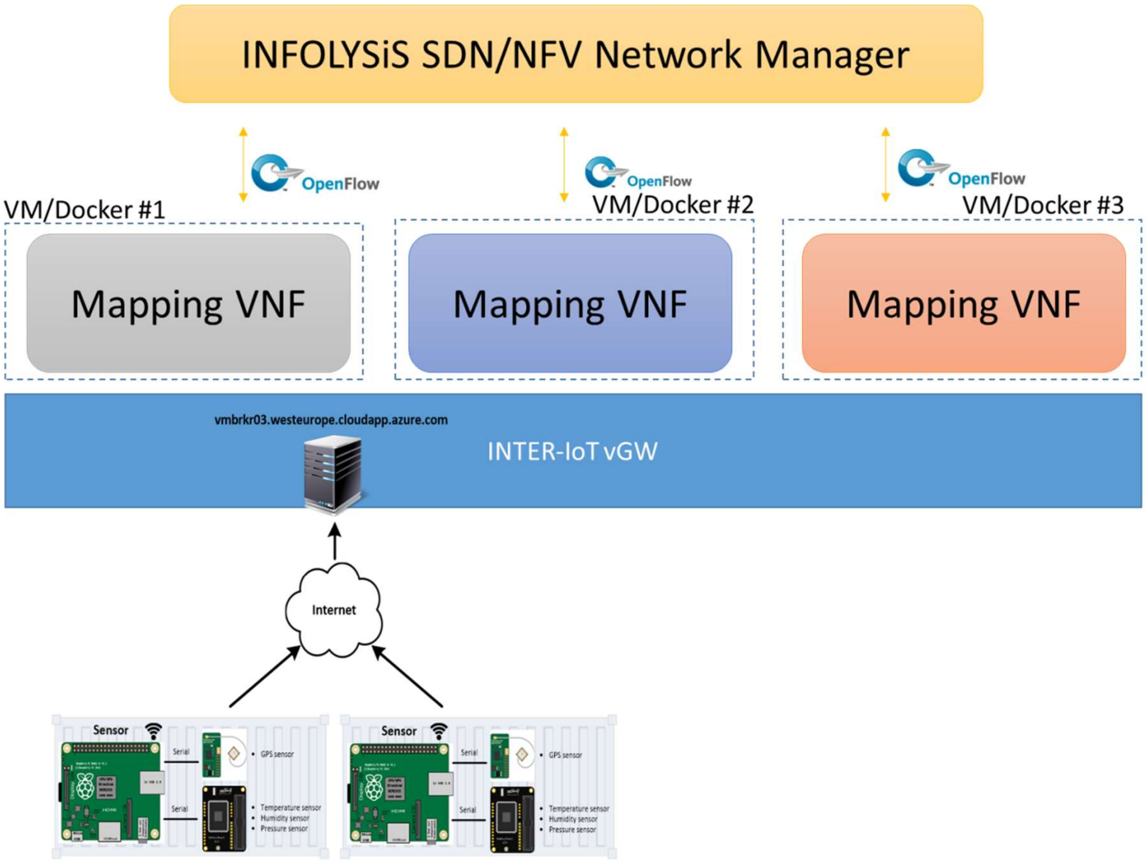


Figure 109: SOFOS Integration and Factory test setup overview.

Thus, with the mapping VNFs provided by INFOLYSiS and the support of the SDN/NFV techniques, the data provided by Raspberry and panStamp are mapped to a common protocol (e.g. HTTP). For the instantiation of the virtual functions, an SDN-compatible (i.e. OpenFlow compliant) cloud computing platform is considered at the MW layer for enterprise users, such as the Docker approach of the INTER-IoT framework.

The container-based topology of the Factory test setup overview, is depicted in the following figure, where each box represents a container on top of the docker-based virtualization.

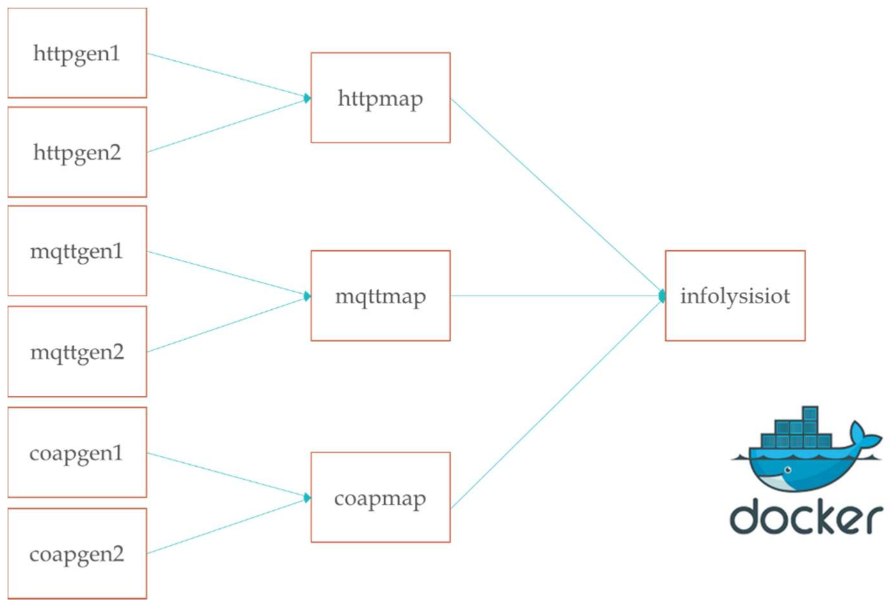


Figure 110: SOFOS Integration and Factory test setup logical topology.

The IoT node generators for the factory setup (i.e. httpgen1/2, mqttgen1/2, and coapgen1/2) will be based on data provided by Raspberry and panStamp units of the INTER-IoT platform or other HW-based IoT nodes that will be available by INTER-IoT system during the early integration phase.

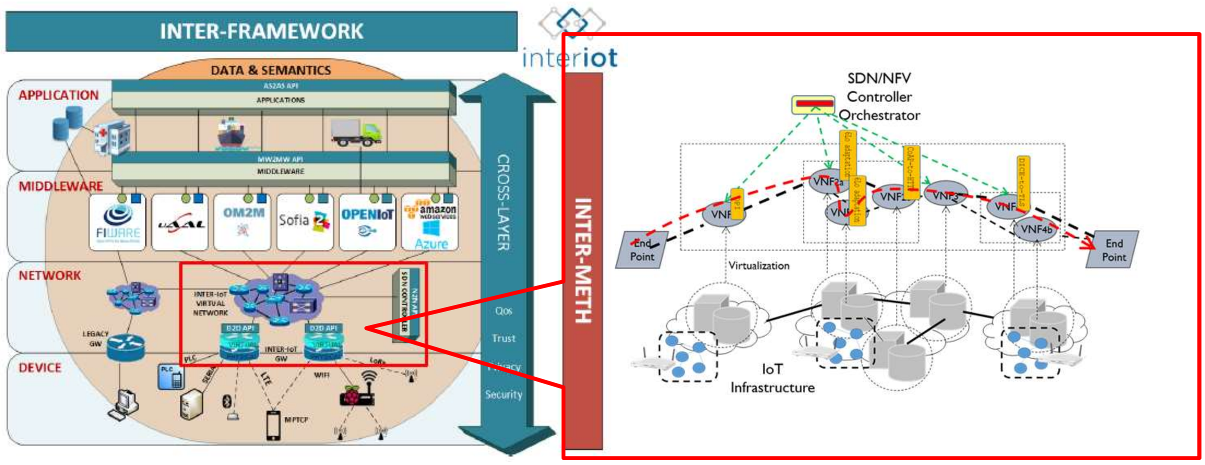


Figure 111: Collaboration approach of the proposed SDN/NFV end-to-end IoT infrastructure within the INTER-IoT architecture.

SOFOS experiment considers that INFOLYSiS will deploy the necessary mapping VNFs (i.e. proxies) on the top of the relevant virtual INTER-IoT GWs. The necessity for the SDN management on top of each testbed it is depicted in the Figure 2, where it is shown that without the proposed SOFOS SDN/NFV-based IoT system, the IoT nodes forwards the data traffic directly to the respective IoT GW, which is not capable to understand the different protocols and therefore communication is not achieved. With SOFOS SDN/NFV-based IoT system, by applying appropriate OPENFLOW commands via the SDN Controller, the data traffic from the IoT nodes will be routed (1) to the SDN-node/switch, then (2) will be diverted/routed/steered by the SDN switch of the testbed (due to appropriate Openflow commands/programming) towards the mapping function in order to be translated to the “interoperable” protocol (e.g. HTTP in the example) and (3) then (once it has been translated) will be further forwarded/routed back to the SDN switch and finally (4) from there to the original destination i.e. the IoT vGW.

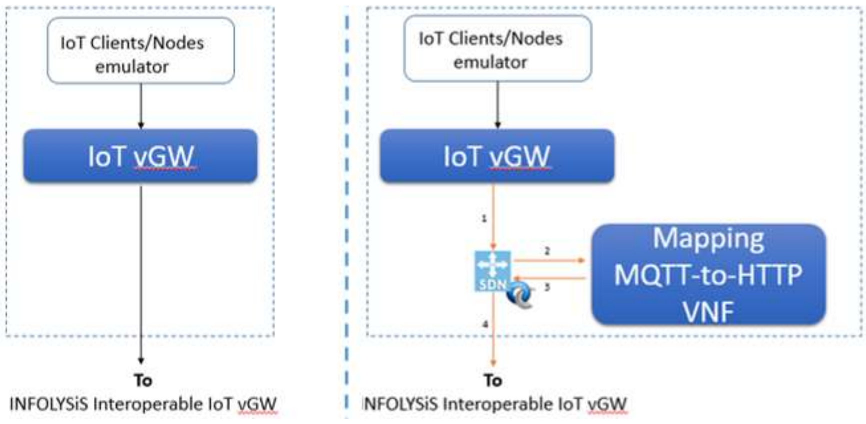

IoT Clients/Nodes emulator
IoT vGW
To
INFOLYSiS Interoperable IoT vGW
IoT Clients/Nodes emulator
IoT vGW
1
2
3
4
SDN
Mapping MQTT-to-HTTP VNF
To
INFOLYSiS Interoperable IoT vGW

Figure 112: Detailed approach of SDN applicability in SOFOS experiment on top of the INTER-IoT testbed.

3.3.7.2 Deliverables and version overview

The following table contains a deliverable list which need to be signed of before FAT testing commences.

ID	**Description**	Check
Documents		
1	Validation and Test reports of the IoT system components	
2	Validation and Test reports of the Pilot system components	
Hardware		
4	Raspberry and panStamp units or other IoT sensors that are available by the INTER-IoT platform	
Tools		
7	OpenVPN	

Table 83: Deliverable checklist

The following table shows the software components and version of which the system release version consists of.

ID	**Description**	**Version**	**Check**
IoT Physical Gateway			
1	AN Controller	V1.0.3	
IoT Virtual Gateway			
4	Fiware	V4.2.3	
5	Docker		
6	Ubuntu		
Universaal container			
7	UniversAAL REST API	V3.2.1	

Table 84: Component version overview

3.3.7.3 Requirements, scenarios and use cases

The following table provides an overview of the relation between the requirements and the test(s) that validate their implementation.

ID	Description	Covered by
15	Support of common IoT communication protocols	INFOLYSiS mapping functions
21	Real time output	INFOLYSiS vGW
70	Easy-to-use user interface	INFOLYSiS vGW
78	Automatic and dynamic selection of communication protocol	INFOLYSiS vGW
229	SDN capabilities	INFOLYSiS vGW
231	Network function virtualization	INFOLYSiS vGW

Table 85: Requirements vs test mapping

The following table provides an overview of the relation between the scenarios and the test(s) that validate their implementation.

ID	Scenario name	Covered by
9	Accident at the port area	
10	Health monitoring system with passengers aboard a ferry	
30	IoT access control, traffic and operational assistance	

Table 86: Scenario vs test mapping

3.3.7.4 Test environment

Introduction

To test the functionality of the SOFOS in combination with the IoT framework a representative test system is needed. The test system needs to approach the "real world" as much as possible. The pilot setups must be recreated and proven. This chapter will describe this environment and the used hardware, software, tools and platforms.

Test environment

This paragraph describes the test environment and the complete system setup used during this FAT.

3.3.7.5 Test setups, tools, hooks and probes

This paragraph describes the test setups, tools, hooks and probes used in the by this document defined tests.

A test setup defines a setup used for a test, and describe the used system parts, interface etc.

Tools define the used COTS tooling that will be used during testing e.g. Logic analyzers, packet sniffer like e.g. Whireshark, developed scripts, etc.

Hooks are used to inject data or control parts of the system used to get the system to handle a needed scenario. This can be hidden, debug or service interfaces which contain the needed functionality.

Probes are used to gather logging from parts of the system under test to provide system feedback and to prove that the system is handling the scenario as it should.

The following paragraphs will define the used setups, tools, hooks and probes which will be referred to by the test descriptions.

TS_01 Test setup x

The following topology refers to the FAT test setup that will be used for verifying the SOFOS solution. The IP addresses of the figure refer to the ones that have been already used for testing at INFOLYSiS testbed envorionemnt. Appropriate ones will be used, while the IoT nodes/emulators that are currently used for testing will be replaced by actual HW-based IoT nodes provided by INTER-IoT ecosystem.

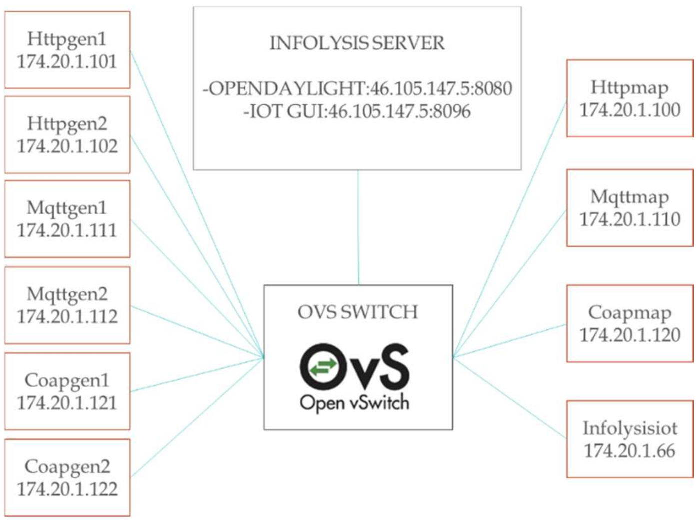


Figure 113: FAT test setup of SOFOS solution.

TT_01 Test tool x

Packet sniffers can be used for verifying the IoT protocols that are entering to the SOFOS systems and the ones that are translated when are existing the system.

TH_01 Test hook x

For testing purposes SOFOS systems utilizes IoT nodes emulator in order data to feed the system. Appropriate hooks that produce MQTT/COAP and HTTP traffic will be used by the SOFOS system in order to demonstrate the SDN-based IoT interoperability provision.

TH_01 Test probe x

INFOLYSiS IoT vGW provides a detailed monitoring interfaces, which shows in real time the data that are translated by the mapping functions to generic UDP streams, proving that the system is handling the scenario as it should.

3.3.7.6 Test description

3.3.7.6.1 S1 – Accident at the port area: Health monitoring system with passengers aboard a ferry

This scenario considers an emergency situation where vessels with casualties are approaching the port where the health units/rescue teams should be prioritized/coordinated depending on the health condition of each casualty. The implementation and use of the SDN paradigm by SOFOS will speed up IoT connections, provide interoperability among different IoT health devices and centralize the management between the vessels domains and the port domain. Moreover, the SDN applicability will allow the prioritization of IoT data flows using traffic engineering, achieving a general overview of the whole network at any time. SOFOS pilot will provide at the first responders' commander, who will coordinate the rescue teams, a unified view of IoT data visualisation. More specifically, the proposed SDN/NFV-enabled IoT GW will be used to provide interoperability between eHealth IoT systems on the different vessels with the coordination center at the port with scope to provide a common unified view of the patients/casualties and the location of the available rescue teams. For this purpose, a virtual mapping function that implements an existing interoperability standards commonly used in healthcare information systems will be deployed by the SDN/NFV orchestrator, offering interoperable and continuous data transmission, allowing to the coordinator to allocate at each available rescue unit the appropriate casualty.

Interoperability in this scenario is required to connect IoT health devices.

The resulting service will be obtained by the integration of:

- IoT wearable health sensor platform
- SOFOS SDN-based interoperable vGW

If IoT wearable health sensors are not available by INTER-IoT project, the described scenario will be adapted to containers with IoT sensors and prioritizing their disembarking based on the data received.

Agile IoT protocol translation/mapping

Design, implementation and integration of interoperable networking layer components (in the form of VNFs) for INTER-FW. The proposed SDN/NFV extension facilitates the deployment of virtual mapping functions and other networking layer components (such as virtual SDN switches), which are based on different standards higher-level communication standards (e.g. TCP/IP, HTTP, CoAP, etc).

This use case involves these requirements: [15], [21], [70], [78], [227] and [231].

Provision of interoperability over heterogeneous IoT units based on SDN/NFV techniques

Test	**Provision of interoperability over heterogeneous IoT units based on SDN/NFV techniques**
Type	System testing
Setup	Integration of SOFOS with INTERIOT via a VPN

Start	IoT sensors produce data
Req.	[15], [21], [70], [78], [227], [231].
Input	Enable the sensor within range of the physical gateway
Output	• no # of heterogeneous IoT nodes that are connected in an interoperable way no # of heterogeneous IoT nodes that provide data to be visualized in a unified way • no # of VNF deployments and SDN rules of heterogeneous IoT nodes that are connected in an interoperable way
Outcome	Pass / Fail

3.3.7.7 Test outcome overview

Test outcome

The following table will provide an overview of the test result of all the performed tests in this FAT.

Test	Description	Outcome
T1	MQTT mapping to UDP-based raw data	Pass / Fail
T2	CoAP mapping to UDP-based raw data	Pass / Fail
T3	HTTP mapping to UDP-based raw data	Pass / Fail
FAT Outcome		**Pass / Fail**

Table 87: Test outcome overview

3.3.7.8 Integration ethics and security

Introduction

SOFOS will give prompt attention to any ethical issues that may arise as a result of project activities, and will address them in a professional way following established and upcoming EU regulations and the corresponding national laws about data protection, digital and property rights issues and protection of minors very closely.

SOFOS

In case that ehealth IoT sensors are available by INTER-IoT project, then the execution of SOFOS experiment involves human participants. If such sensors are not available and IoR container sensors will be used, then human participants are not involved and therefore Ethics/Privacy aspects are not considered.

In the case of ehealth sensors, the human participants for these activities will be recruited by INTERIOT consortium and not the third party for the needs of the execution of the experiments of the Open Call. Established procedures will be followed that respect all pertinent laws (Directive 95/46/EC and General Data Protection Regulation) and ethics standards, in particular related to contacting individuals, providing comprehensive and clear information about the objectives of the conducted research and the use of the collected data, obtaining their consent and not sharing any of the collected personal data with third parties.

The precise documentation that will be communicated to users participating in the pilot studies will be made available before the execution of the experiment and will be included as appendices to the final deliverable.

3.3.8 Third Party: ACHILLES

Access control and endpoint authentication in the IoT is a challenging problem. Things are usually small devices with limited storage capacity, power, energy, and processing capabilities, in order to be inexpensive and practical. In many cases Things are "exposed" to tampering, whereas in many application scenarios, after Things are deployed, it is not easy to access them. Things usually are not able to perform "heavy" tasks, such as complex cryptographic operations. Storing user credentials or any other sensitive information in a Thing creates security risks, adds storage overhead, and makes security management an impossible task. When it comes to interoperable applications, Things (or even gateways) cannot interpret complex business roles and processes. Moreover, companies are not willing to share sensitive information about their users with a Thing (or a gateway), even if this information is required by an access control mechanism, neither do they want to invest in yet another security system.

The ACHILLES project overcomes these limitations by allowing the delegation of security operations to a third party, referred to as the Access Control Provider (ACP), which can be implemented by a trusted separate entity, or even the service provider itself. The ACHILLES concept is depicted in Figure 114.

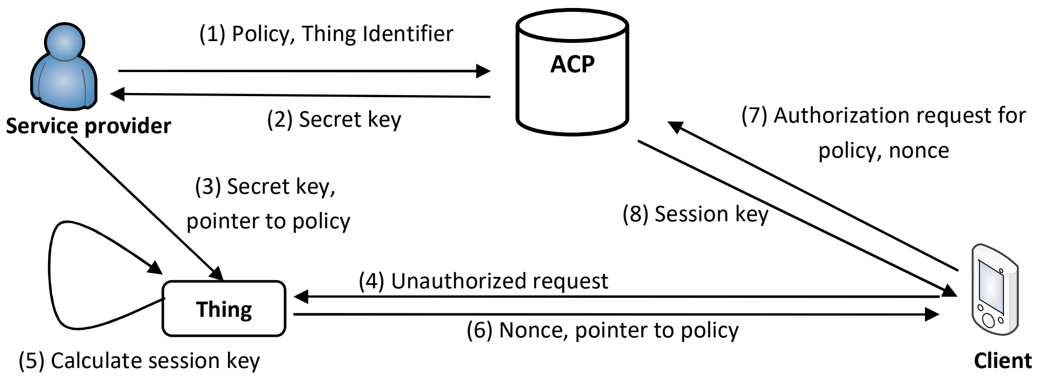

(1) Policy, Thing Identifier
ACP
Service provider
(2) Secret key
(7) Authorization request for policy, nonce
(3) Secret key, pointer to policy
(8) Session key
(4) Unauthorized request
Thing
(6) Nonce, pointer to policy
(5) Calculate session key
Client

Figure 114: The concept of the ACHILLES project.

The main idea of the ACHILLES concept is that IoT service providers store access control policies in ACPs and in return ACPs generate secret keys which are stored in Things (steps 1-2). These keys are generated, during a setup phase, using a secure hash with input the Thing identifier. Additionally, Things are configured with pointers (e.g., a URL that points to an ACP and a particular file) to the access control policies that protect sensitive resources (step 3). Every time a client requests access to a protected resource (step 4) the Thing uses a secure hash function to generate a session key (step 5). The secret key used by that function is the key generated by the ACP and the hash inputs are: (a) the pointer to the policy that protects the resource and (b) a random nonce. The Thing transmits the nonce and the pointer to the client (step 6), which in return requests authorization from the appropriate ACP (over a secure channel) (step 7). The ACP has all the necessary information required to calculate the session key: if the client is authorized, the ACP calculates the session key and transmits it back to the client (step 8). Providing that: (i) the Thing has not lied about its identity and (ii) the messages exchanged between the client and the Thing have not been modified, the Thing and the client end up sharing a secret key. This key can be used for securing subsequent communications (e.g., by using DTLS).

3.3.8.1 Testing system

Our testing system is illustrated in Figure 115.

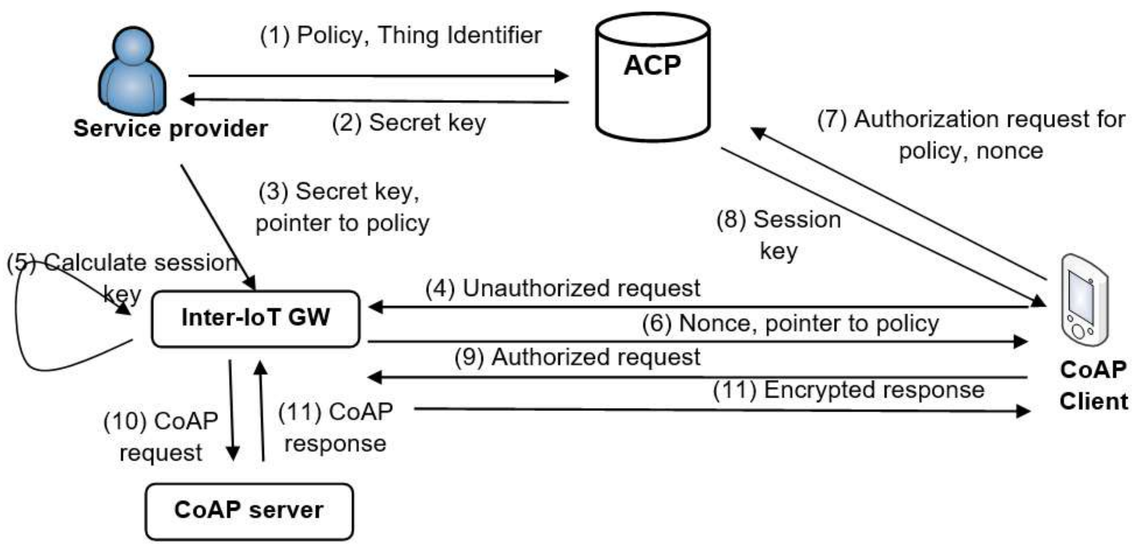


Figure 115: Testing System.

Our testing system is composed of the following components: a CoAP server running in a RIOT-based device, a Java-based CoAP Client, Java-based access control providers (ACP) and a Java-based extension bundle running in an Inter-IoT GW implementing the ACHILLES protocol. In the device, a CoAP resource is provided only to authorized clients. The resource access is regulated by a username-password based access control policy. The access control policy is stored in the ACP and in essence maps the username of a client to a Boolean output (true, false). When the output of an access control policy invocation is true (i.e., an appropriate username-password pair was provided), the client is considered authorized.

For testing purposes we assume an out-of-band, secured, setup phase. It is assumed that the ACP has generated a Master Secret Key (MSK) and a Uniform Resource Identifier (URI) for the policy (steps 1-2). Moreover, the Inter-IoT GW is equipped with a secret key generated by the ACP using as input the MSK and the resource URI (step 3). The gateway maintains an Access Table that contains a tuple of the form [Resource,Policy,SK] where Resource is the URI of the protected resource, Policy is the URI of the policy that protects the resource, and SK is the secret key that has been generated by the ACP. The client initially sends an unauthorized CoAP request to the Inter-IoT GW (step 4). Upon receiving this request, the GW retrieves the Policy URI and the SK from the Access Table and generates a token. The token is a (public) variable unique among all sessions of that specific GW. Then, the GW uses SK to calculate a session key. All tokens and session keys are stored in a Token Table (step 5). Finally, the GW sends the URI and the Token back to the client (step 6). Upon receiving the GW response, the client sends an authentication request to the ACP, including her username and password (step 7). The ACP authenticates the client, examines if the username and the password are correct, calculates a secret key and transmits it back to the client (step 8). The keys calculated by the ACP and the device are identical if the following conditions hold: (i) the GW is a legitimate device, (ii) messages have not been tampered with by an attacker. Finally, the client performs an authorized request (step 9) which is forwarded to the CoAP server (step 9). The CoAP server responds to the GW (step 10), the GW encrypts the response using the session key and sends it back to the client (step 11).

A sequence diagram of our testing system follows.

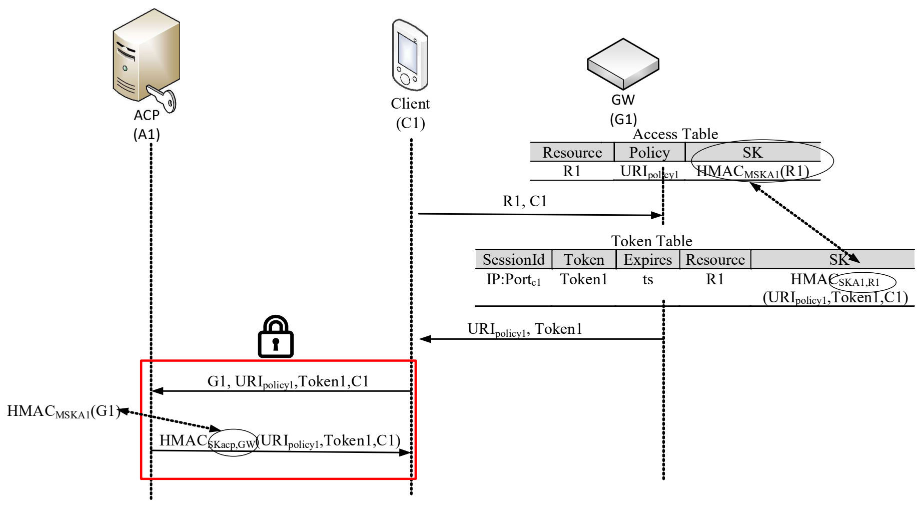

ACP
(A1)
Client
(C1)
GW
(G1)
Access Table
Resource | Policy | SK
R1 | URI$_{policy1}$ | HMAC$_{MSKA1}$(R1)
R1, C1
Token Table
SessionId | Token | Expires | Resource | SK
IP:Port$_{c1}$ | Token1 | ts | R1 | HMAC$_{SKA1,R1}$(URI$_{policy1}$,Token1,C1)
URI$_{policy1}$, Token1
G1, URI$_{policy1}$,Token1,C1
HMAC$_{MSKA1}$(G1)
HMAC$_{SKacp,GW}$(URI$_{policy1}$,Token1,C1)

Figure 116: Message sequence diagram of the core functions of the testing system.

3.3.8.2 Integration of IoT framework

ACHILLES will interact with the following D2D Gateway Components:

- **Logging**: This component will be used for creating test logs.
- **Protocol Controller**: ACHILLES will extent the protocol controller with a new component specific to the ACHILLES protocol. We refer in the following to this component as Achilles GW module.
- **CoAP Protocol Module**: The Achilles GW module will utilize the CoAP GW protocol module in order to dispatch CoAP requests to the appropriate devices.
- **Gateway Configuration**: The gateway configuration component will be extended to hold ACHILLES specific configuration.
- **API**: ACHILLES will extend the appropriate D2D GW APIs in order to enable Access Table modifications.

The following table contains a list of D2D gateway components that will be used during FAT tests.

ID	Component	Tests
1	Logging	All
2	Protocol Controller	All
3	CoAP Protocol Module	T1.2.2, T2.2.2
4	Gateway Configuration	T1.1.1, T2.1.1, T2.1.2
5	API	T1.1.1, T2.1.1, T2.1.2

Table 88: List of GW components that will be used during the Tests

3.3.8.3 Deliverables and version overview

The following table contains a deliverable list which need to be signed of before FAT testing commences.

ID	Description	Check
Documents		
1	Validation and Test reports of the IoT system components	
2	Validation and Test reports of the Pilot system components	
3		
Tools		
7	Wireshark	

Table 89: Deliverable checklist

The following table shows the software components and version of which the system release version 1.0 consists of.

ID	Description	Version	Check
IoT Physical Gateway			
1	Logging	VX.X.X	
3	Protocol Controller	VX.X.X	
3	CoAP	VX.X.X	
4	Gateway Configuration	VX.X.X	
ACHILLES			
4	ACP	V1.0.0	
5	ACHILLES CoAP Client	V1.0.0	
6	GW Module	V1.0.0	
7	CoAP Server	V1.0.0	

Table 90: Component version overview.

3.3.8.4 Requirements, scenarios and use cases

The following table provides an overview of the relation between the requirements and the test(s) that validate their implementation.

ID	Description	Covered by
Architecture		
2	Scalability. Design	T1.1.1, T2.1.1
6	Efficiency of the processing of information	T1.1.1, T2.1.1
Communications		
14	Platform independent	T1.2.2, T2.2.2
15	Common IoT communication protocols must be supported.	T1.2.2, T2.2.2
Functionality		
11	Addressability and reachability	T1.1.1, T2.1.1
22	Unique identifier (this is the new 256 requirement)	T1.1.1, T2.1.1
API		
243	Gateway access API	T1.1.1, T2.1.1
Interoperability		
13	Extensibility	T2.1.2
Legality		
76	Interoperability between things from different administrative/management domains	T2.2.2

Performance		
72	Communication should be done using protocols that are efficient in terms of amount of exchanged information over message size	T1.2.2, T2.2.2
Security		
27	System security	T 1.2.1, T 2.2.1, T3.1.1, T3.2.1, T3.2.2
28	System privacy	T3.1.1, T3.2.1, T3.2.2
95	Robustness, resilience and availability	T3.1.1, T3.2.1, T3.2.2
98	Data provenance	T1.1.1, T2.1.1

Table 91: Requirements vs test mapping.

The following table provides an overview of the relation between the scenarios and the test(s) that validate their implementation.

ID	Scenario name	Covered by
1	IoT Data sharing	T1.1.1, T1.2.1, T1.2.2
2	B2B Services	T2.1.1, T2.2.1, T2.2.2
3	System under attack	T3.1.1, T3.2.1, T3.2.2

Table 92: Scenario vs. test mapping.

3.3.8.5 Test environment

Introduction

To test the functionality of the ACHILLES project in combination with the IoT framework a representative test system is needed. The test system needs to approach the “real world” as much as possible. The pilot setups must be recreated and proven. This chapter will describe this environment and the used hardware, software, tools and platforms.

Test **environment**

Our testing environment is composed of the following components: a CoAP server running in a RIOT-based device, a Java-based CoAP Client, Java-based access control providers (ACP) and a Java-based extension bundle running in an Inter-IoT GW implementing the ACHILLES protocol. In the device, a CoAP resource is provided only to authorized clients. The CoAP-Client and the ACP server will be executed in isolated virtual machines running in a single laptop. The hosting laptop will be in the same LAN as the Inter-IoT GW. The CoAP server will accept CoAP GET, as well as CoAP PUT requests. The latter type of requests will be used for controlling the usage of an (emulated, probably) actuator.

3.3.8.6 Test setups, tools, hooks and probes

TS_01 Test setup 1

This setup is used by all tests that involve a single ACP. During this setup the ACP is configured with username-password pairs. Moreover, the ACP generates and securely stores a Master Secret Key (MSK). In addition, access control policies are created and stored in ACPs, and a Uniform Resource Identifier (URI) for each policy is generated. The ACP uses a secure HMAC and the MSK to calculate a secret key as follows: $SK_{acp,GW} = HMAC_{MSK}(ID_{GW})$. The calculated SK is then configured to the GW. Finally, each CoAP client is configured with an ACP username and password as well as with the URI of the desired resources.

TS_02 Test tool 2

This setup is used by all tests that involve multiple ACPs. This setup is in essence TS_01 repeated for each ACP. Each ACP is responsible for protecting different resources. Moreover, each CoAP client is configured with a single ACP.

TT_01 Packet sniffer

In order to visualize the exchanged messages, we will use the Wireshark network sniffer.

TH_01 Test configurator

This hook is used for configuring the ACP with the appropriate parameters, as well as for injecting into the Inter-IoT GW the generated secret keys and proper configuration files. The configuration files contain entries related to the available CoAP resources.

TH_02 Session re-player

This hook is used for replaying requests from authorized users with or without modifications and it is used for testing the security properties of the system.

TP_01 GW Dumper

This probe will output all state related to ACHILLES maintained by the Inter-IoT GW. This includes configuration files, access control tables, as well as Token Tables.

TP_03 Actuator output

This probe is used for outputting the state of the actuator.

3.3.8.7 Test description

3.3.8.7.1 S1 - IoT data sharing

The objective of this scenario is to enable a resource owner to share measurement data with other authorized users. In this scenario resource owners have Things they own connected to an Inter-IoT GW. These Things perform various measurements. Measurements are grouped based on the Thing location and can be accessed in real-time using the appropriate CoAP resource URIs (e.g., coap://window.bedroom.user1/temperature). Resource owners define access control policies in the ACP (e.g., “Friends”, “Family”) and define in the GW the access control policy that protects each group of measurements (e.g., “window.bedroom.user1 can be accessed by Friends”).

3.3.8.7.2 U1: New measurement group creation

The resource owner creates a new group of measurements and registers them in the GW, providing at the same time a pointer to the access control policy that protects them.

T1.1.1 New measurement group creation

ID	T1.1.1
Test	Registration of a new group of measurements
Type	System testing
Setup	Needs setup TS_01

Start	Access Table in GW is empty
Req.	[2],[6],[11],[22],[243],[98]
Input	Resource owner invokes the resource registration API call
Output	Access Table is updated
Logs	Folder “`T1_Output`”, prefix “`T1.1.1_achilles`” >
Outcome	Pass / Fail

3.3.8.7.3 U2- User request

A user is interested in a receiving a measurement protected under a specific access control policy. The user performs an initial request (an unauthorized request) to learn all information required for authorization. Then, it authenticates himself in the appropriate ACP and obtains an authorization token. The latter is used for performing an authorized request.

T1.2.1 Unauthorized request

ID	**T1.2.1**
Test	Request from an unauthorized user for a protected resource
Type	System Testing
Setup	Needs setup TS_01
Start	Access Table contains some entries
Req.	[27]
Input	A CoAP request from an unauthorized user
Output	● Check if the resource is included in the Access Table ● Generate session key ● Generate token ● Respond to the user with the ACP URI and the token
Logs	Folder “`T1_Output`”, prefix “`T1.2.1_achilles`” >
Outcome	Pass / Fail

T1.2.2 Authorized request

ID	**T1.2.2**
Test	Request from an authorized user for a protected resource
Type	System Testing
Setup	Needs setup TS_01
Start	Access Table and Token Table contains some entries

Req.	[14],[15],[72],[76]
Input	A CoAP request from an authorized user
Output	● Check if the resource is included in the Access Table ● Check if the Token is included in the Token Table and it is still valid ● Perform a CoAP request to the appropriate Thing ● Encrypt the response and send it back to the user
Logs	Folder “`T1_Output`”, prefix “`T1.2.2_achilles`” >
Outcome	Pass / Fail

3.3.8.7.4 S2 - B2B services

The objective of this scenario is to enable protected resources for multiple groups of authorized users belonging to diverse administrative domains. In this a scenario, a resource owner owns actuators connected to an Inter-IoT GW. These actuators can accept commands as CoAP PUT requests. Various stakeholders define access control policies in their corresponding ACP (e.g., “Employees”, “Managers”). Moreover, the resource owner defines in the GW the access control policies that protect each operation (e.g., “switch1 can be turned on by the “Employees” of the company that has business relationships with ACP A, or the “Employees” of the company that has business relationships with ACP B).

3.3.8.7.5 U1: New operation creation and management

The resource owner defines an operation that can be performed on an actuator and provides pointers to the policies that protect this operation. Moreover, later on, the resource owner can modify the list of the pointers to policies by adding or removing a pointer.

T2.1.1 New operation registration

ID	**T2.1.1**
Test	Registration of a new resource protected by multiple policies
Type	System testing
Setup	Needs setup TS_02
Start	Access Table in GW is empty
Req.	[2],[6],[11],[22],[243],[98]
Input	Resource owner invokes the resource registration API call
Output	Access Table is updated
Logs	Folder “`T1_Output`”, prefix “`T2.1.1_achilles`” >
Outcome	Pass / Fail

T2.1.2 List of policies modification

ID	T2.1.2
Test	Add or remove a pointer to an access control policy
Type	System testing
Setup	Needs setup TS_02
Start	Access Table in GW has some entries
Req.	[13]
Input	Resource owner invokes the resource registration API call
Output	Access Table is modified
Logs	Folder "`T1_Output`", prefix "`T2.1.2_achilles`" >
Outcome	Pass / Fail

3.3.8.7.6 U2- User request

A user is interested in triggering an actuator protected by some access control policies. The user performs an initial request (an unauthorized request) to learn all information required for authorization. Then it authenticates himself in the appropriate ACP and obtains an authorization token. The latter is used for performing an authorized request.

T2.2.1 Unauthorized request

ID	T2.2.1
Test	Request from an unauthorized user for a protected actuator
Type	System Testing
Setup	Needs setup TS_02
Start	Access Table contains some entries
Req.	[27]
Input	A CoAP request from an unauthorized user
Output	● Check if the resource is included in the Access Table ● Generate session key ● Generate token ● Respond to the user with the ACP URIs and the token
Logs	Folder "`T1_Output`", prefix "`T2.2.1_achilles`" >
Outcome	Pass / Fail

T2.2.2 Authorized request

ID	T2.2.2
Test	Request from an authorized user for a protected resource
Type	System Testing
Setup	Needs setup TS_02
Start	Access Table and Token Table contains some entries
Req.	[14],[15],[72],[76]
Input	A CoAP request from an authorized user
Output	● Check if the resource is included in the Access Table ● Check if the Token is included in the Token Table and it is still valid ● Perform a CoAP request to the appropriate Thing ● Encrypt the response and send it back to the user
Logs	Folder "`T1_Output`", prefix "`T2.2.1_achilles`" >
Outcome	Pass / Fail

3.3.8.7.7 S3 - System under attack

The objective of this scenario is to evaluate the security of the integrated platform in the presence of malicious users.

3.3.8.7.8 U1-New sessions

An attacker is able to capture and record successful sessions. He then replays the messages in order to gain access to a protected resource.

Replay attack

ID	T3.1.1
Test	Emulate an attacker that repeats captured sessions
Type	Security Test
Setup	Needs setup TS_01 or TS_02 and Test hook 1
Start	Access Table and Token Table contains some entries
Req.	[27],[28],[95]
Input	A CoAP request that appears to be from an authorized user
Output	● Check if the resource is included in the Access Table ● Check if the Token is included in the Token Table and it is still valid ● Reply with an error
Logs	Folder "`T1_Output`", prefix "`T3.1.1_achilles`" >>
Outcome	Pass / Fail

3.3.8.8 U2-Tampering with existing sessions

An attacker is able to intercept the communication between an authorized user and a Thing. His goal is to modify the transmitted packets in way that will give him access to protected resources.

Packet modification attack

ID	T3.2.1
Test	Emulate an attackers that modifies transmitted packets
Type	Security Test
Setup	Needs setup TS_01 or TS_02 and Test hook 1
Start	Access Table and Token Table contains some entries
Req.	[27],[28],[95]
Input	A CoAP request that appears to be from an authorized user
Output	● Check if the resource is included in the Access Table ● Check if the Token is included in the Token Table and it is still valid ● Reply with an error
Logs	Folder "`T1_Output`", prefix "`T3.2.1_achilles`" >>
Outcome	Pass / Fail

T3.2.2 Man-in-the-middle attack

ID	T3.2.2
Test	Emulate an attackers that perform man-in-the-middle attack
Type	Security Test
Setup	Needs setup TS_01 or TS_02 and Test hook 1
Start	Access Table and Token Table contains some entries
Req.	<Define the requirements involved in [x], format>
Input	A CoAP request that appears to be from an authorized user
Output	● Check if the resource is included in the Access Table ● Check if the Token is included in the Token Table and it is still valid ● Reply with an error
Logs	Folder "`T1_Output`", prefix "`T3.2.2_achilles`" >>
Outcome	Pass / Fail

3.3.8.9 Test outcome overview

Test outcome

The following table will provide an overview of the test result of all the performed tests in this FAT.

Test	**Description**	**Outcome**
T1.1.1	Container registration	Pass / Fail
T1.2.1	Unauthorized request	Pass / Fail
T1.2.2	Authorized request	Pass / Fail
T2.1.1	New operation registration	Pass / Fail
T2.1.2	List of policies modification	Pass / Fail
T2.2.1	Unauthorized request	Pass / Fail
T2.2.2	Authorized request	Pass / Fail
T3.1.1	Replay attack	Pass / Fail
T3.2.1	Packet modification attack	Pass / Fail
T3.2.2	Man-in-the-middle attack	Pass / Fail
FAT Outcome		**Pass / Fail**

Table 93: Test outcome overview.

3.3.8.10 Integration ethics and security

Introduction

During the Ethical Review it was requested that ethics be included within the project scope. This section addresses recommendations and proposed solutions to the ethical and security points of the integration.

Achilles

The FAT plan presented in this document does not raise any ethical issue, since the data that will be used during the tests will be artificially created and will not affect any real entity. Moreover, all experiments will be conducted in a closed system that does not interact with the external world, hence no concerns should be raised. When it comes to the actual deployment of ACHILLES, various risks should be taken into consideration. ACHILLES ACPs may have access to sensitive user information, including user names, passwords, and access control policies. Moreover, ACHILLES ACPs may have access to the “seed” used by the Things to generate session secret keys. For these reasons, a security incident handling plan should be considered.

3.3.9 Third Party: Inter-HINC

The following figure describes the current design of INTER-HINC, which fits well into the design of the Inter-IoT framework. Two main components that will interact with other IoT platforms and services outside the Inter-IoT framework are:

- LocalManagementService: instances of Local Management Service are used to interface to IoT providers.
- Global Management Service: instances of Global Management Server are used to the application and other middleware to use INTER-HINC to control IoT devices, networks and services and acquire IoT data.

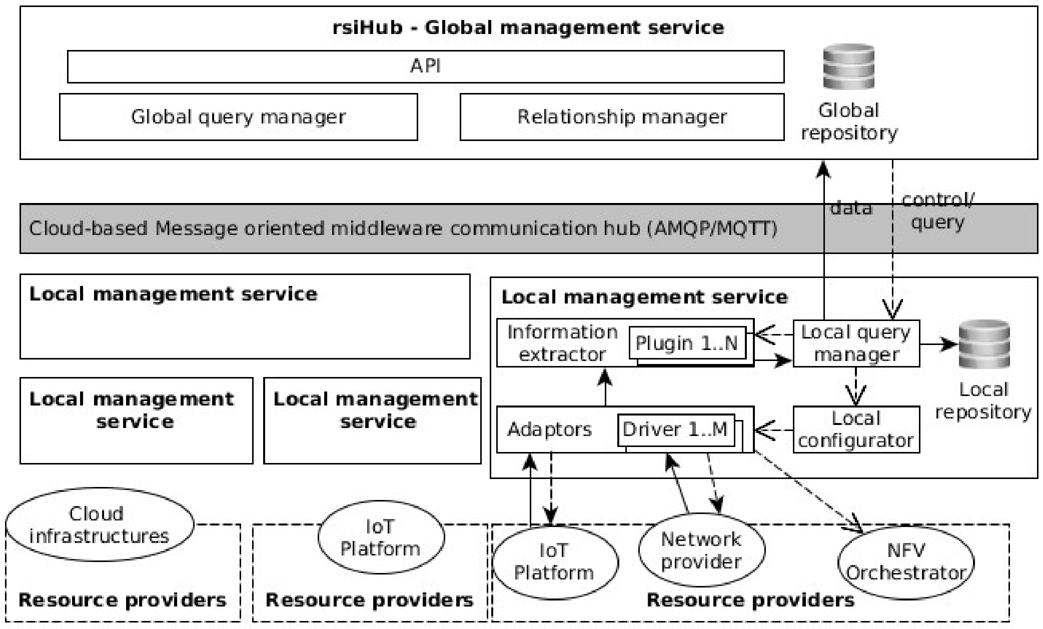


Figure 117: INTER-HINC Architecture overview.

Both Local Management Service and Global Management Service provide

- REST APIs: standard REST APIs for any clients to use our services.
- Client libraries: for applications to program calls to Service.

3.3.9.1 Requirements, scenarios and use cases

The following table provides an overview of the relation between the requirements and the test(s) that validate their implementation.

ID	Description	Covered by
Application		
239	Support Service choreography and Service Orchestration	T1.1.2, T2.1.5
240	Support Mash-up	T1.2.1, T1.2.2, T2.1.5
241	Native support services	T2.2.1, T2.2.2
Architecture		
2	Scalability. Design	T1.1.2, T1.2.1, T2.1.2, T2.1.3, T2.1.4, T2.1.5, T2.1.6, T.2.1.7, T2.1.8,

		T2.1.9, T2.2.1, T3.1.1, T3.1.2, T3.2.1, T3.2.2
6	Efficiency of the processing of information	T1.1.1, T1.1.2, T1.2.1, T2.1.3,
9	Multi-level data processing support	T1.1.1, T1.1.2, T1.2.1, T1.2.2
Communications		
7	Support of opportunistic communications to avoid data loss	T2.1.3, T2.1.4, T2.1.8, T2.2.1, T2.2.2
14	Platform independent	T1.2.1, T3.1.1, T3.1.2
15	Common IoT communication protocols must be supported.	T1.2.1, T2.1.1, T2.1.9
17	Dynamic network support	T2.1.1, T2.1.9
18	Roaming across networks	T2.1.1, T2.1.9
39	Gateway capabilities	T1.1.2, T2.1.1, T2.1.2
45	Connectivity not based on HW identifiers	T2.1.1, T2.1.2, T2.1.9
78	Automatic and dynamic selection of communication protocol	T1.1.1, T1.1.2, T2.1.2
231	Network function virtualization	T2.1.1, T2.1.2, T2.1.9,
232	Fault tolerance	T1.1.2, T2.1.3, T2.1.4, T2.1.8, T2.1.9, T3.1.2, T3.2.2,
233	Flow control and network information tracking	T1.1.2, T1.2.1, T2.1.2, T2.2.1, T2.2.2,
Functionality		
11	Addressability and reachability	T1.1.2
179	IoT Platform Semantic Mediator supports platform to platform communication and communication between platforms and an external actor	T1.1.1, T1.1.2,
Interoperability		
4	Alignment with other IoT architectures, especially with AIOTI	T2.1.1
13	Extensibility	T2.1.1
Middleware		
234	Provide connectors to middleware standards	T2.1.1, T2.1.2, T2.1.9
Security		
95	Robustness, resilience and availability	T2.1.3, T2.1.4, T2.1.8, T3.1.2, T3.2.2,
Virtualization		
242	Object/Device virtualization	T2.1.1,

Table 94: Requirements vs test mapping

The following table provides an overview of the relation between the scenarios and the test(s) that validate their implementation.

ID	Scenario name	Covered by
2	IoT support for transport planning and execution	Scenarios 1,2, 3
5	Monitoring of containers carrying sensitive goods	Scenarios 1, 2, 3
9	Accident at the port area	Scenarios 1, 2, 3
13	IoT interoperability for Vessel Arrivals	Scenarios 1, 2, 3
19	Transport on truck breaks down or is hijacked	Scenarios 1, 2, 3
29	Reliable control of robotic cranes and trucks in port terminals	Scenarios 1, 2, 3
30	IoT access control, traffic and operational assistance	Scenarios 1, 2, 3

Table 95: Scenario vs test mapping

3.3.9.2 Test Environment

Introduction

This document outlines a very first overview of our testing environment for INTER-HINC within Inter-IoT. Since INTER-HINC is still just at the beginning of its design (mainly about "Models and Approaches for Slice Interoperability" activities at the writing of this document), the report here mostly sets the overview of the test environment and scenarios. Detailed and concrete FAT can be done only after having detailed design, implementation and integration.

3.3.9.3 Test environment

For testing environment, it is important to understand components of INTER-HINC. Figure 117 gives the overview of INTER-HINC. The key principle of INTER-HINC is to make resources for clients into manageable slices; a slice consists of IoT resources, network function resources and cloud resources. Important layers to be tested are:

- INTER-HINC Local Services and Resource (IoT, Network Functions, and Cloud) Providers: they are instances of Local Services, distributed in the cloud or the edge, interfacing to specific Resource Providers.
- INTER-HINC Global Service: it serves clients by communicating with Local Services via the Cloud Message Broker.
- Resource Slice Management: this includes utilities for clients to manage resources within slices.

To facilitate services developed in INTER-HINC, we provide Programming API for clients to query and control resources in slices.

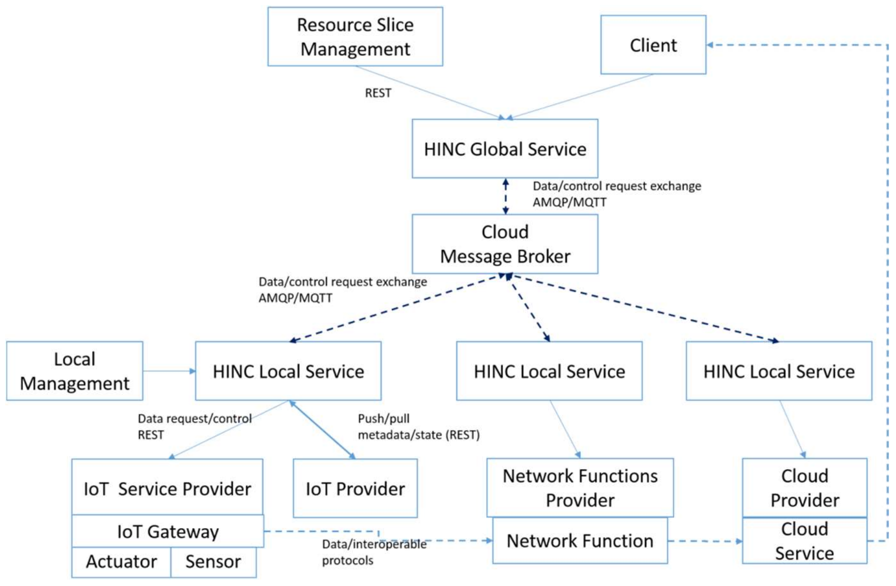


Figure 118: Overview of components in INTER-HINC.

In this view, the system under test (SUT) for Factory Acceptance Testing will consist of

- Infrastructures: several Virtual Machines and containers for hosting INTER-HINC services (such as for LocalService, Broker, and Global Service). The infrastructures will be based on typical cloud virtual machines (VM)/dockers (using resources from public providers, like Google cloud), lightweighted VMs/containers emulated Raspberry

PI, and physical Raspberry PI. The infrastructures also include global cloud brokers based on existing cloud services (e.g., using cloudamqp.com and cloudmqtt.com).

- INTER-HINC Services: include Local Services, Global Services and other components to be deployed atop the above-mentioned infrastructures.

The testing environment will be based on data-centered cloud computing infrastructures, combined with edge resources, reflecting the view of slice resources. An example of the testing environment is given in Figure 119, described in our current working INTER-HINC paper[35]. In principle, we will have similar structures of testing environments with different configurations.

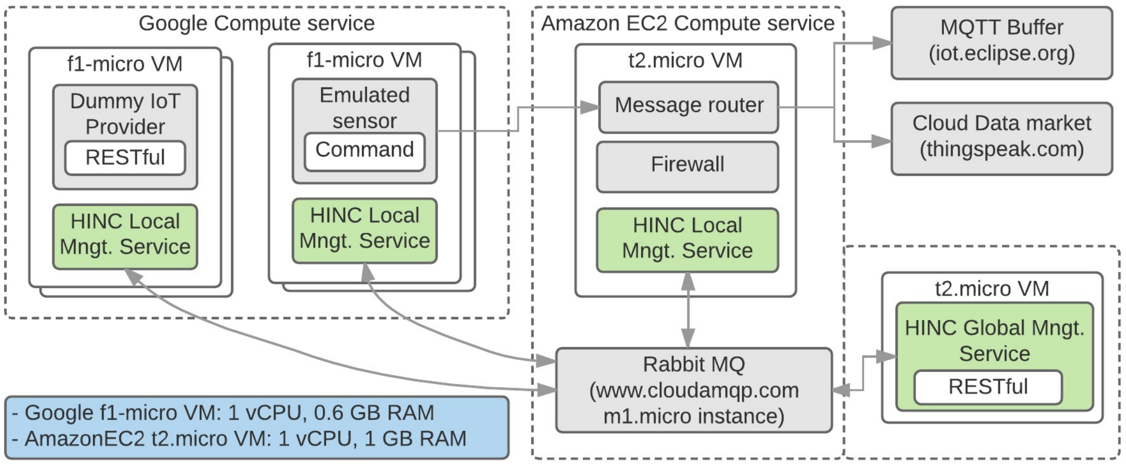

Google Compute service
f1-micro VM
Dummy IoT Provider
RESTful
HINC Local Mngt. Service
f1-micro VM
Emulated sensor
Command
HINC Local Mngt. Service
Amazon EC2 Compute service
t2.micro VM
Message router
Firewall
HINC Local Mngt. Service
MQTT Buffer (iot.eclipse.org)
Cloud Data market (thingspeak.com)
t2.micro VM
HINC Global Mngt. Service
RESTful
Rabbit MQ (www.cloudamqp.com m1.micro instance)
- Google f1-micro VM: 1 vCPU, 0.6 GB RAM
- AmazonEC2 t2.micro VM: 1 vCPU, 1 GB RAM

Figure 119: An example of infrastructure configuration for INTER-HINC as the system under test (SUT)

3.3.9.4 Test setups, tools, hooks and probes

Test setups will require the deployment of SUT described previously. For testing tools, currently, in our plan we will use tools developed by the INTER-HINC team[36] which includes both model-based testing tools[37] for modeling and generating test cases, and a runtime testing framework[38]. Especially, we will focus on our current developing tool – T4SINC – and other testing tools for

- Easy to deploy various components of SUT into virtual environments of VMs/dockers
- Run test cases for FAT
- Collect test data from various places and sources
- Test analytics (including machine learning based techniques)

One important aspect is to create/obtain suitable IoT dataset for testing. In this case, we will work with other partners to obtain datasets and components from Inter-IoT use cases.

TS_01 Test setup Mixed Local and Cloud Resources with Real IoT providers

[35] Hong-Linh Truong, Duc-Hung Le, Nanjangud Narendra, Provisioning and Managing Interoperable Resource Slice, Across IoT, Network Functions and Clouds, Nov 2017. Working paper.

[36] http://github.com/rdsea
[37] https://github.com/rdsea/T4UME
[38] https://github.com/rdsea/T4SINC

In this TS_01 setup, we will use existing IoT providers (and their infrastructures) from Inter-IoT for IoT providers, we use cloud resources for INTER-HINC Local Services and Global Services, for Network functions Providers, and Cloud Providers. This test setup is used to test real providers.

Infrastructures: The real IoT providers from Inter-IoT are still being determined and the infrastructures of these providers depend the existing ones. For Local Services we will use containers and lightweighted VMs (with similar configurations like Raspberry Pi). For the Global Services and other components, we will use typical VMs and containers.

Scale: In this setup, we will only scale the Global Services and number of clients as the IoT providers will be fixed (based on real deployment of Inter-IoT).

Data and software for IoT providers: this is dependent on the existing providers (to be identified)

Testing goal: we will test performance, data compatibility, conformance, and installation

TS_02 Test setup Mixed Local and Cloud Resources

In this TS_02 setup, in overall, we will use Local/Edge infrastructures and cloud infrastructures

Infrastructures: Various local/edge resources/services running in RaspberryPI and private clouds will be used for HINC Local Services, IoT Providers and Network Function Providers whereas global and other resources running in public clouds are used for the Global Service, Cloud Services and possible Network Function Providers. We will use our own Raspberry PI and private cloud resources as well as existing IoT Providers offered by Inter-IoT partners. We will use Google Cloud and Amazon EC resources.

Scale: In this setup, we will change number of VMs/containers, number of providers, size of data to be tested, number of local/edge sites, individual services/units, subsystems and the entire system.

Data and Software for IoT: In this setup, we will use datasets and software from Inter-IoT and other examples for the test. Currently, we are still investigating available datasets and software from Inter-IoT. We already have datasets and (emulated) software for electricity, alarms, camera, and GPS from various sources representing different types of IoT providers.

Testing goal: we will test performance, data compatibility, conformance, and installation

TS_03 Test setup Cloud Only

In this setup, we will use entirely cloud resources for testing using public and private Cloud infrastructures (local/edge resources are emulated by cloud resources).

Infrastructures: Various local resources/services running in the cloud using different data centers; global services running in the public cloud

Scale: We will scale number of VMs/containers, number of providers, size of data to be tested, individual services/units, subsystems and the entire system

Data and Software for IoT: In this setup, we will use datasets and software from Inter-IoT and other examples for the test as described in TS_02.

Testing goals: Performance, data compatibility, conformance, installation.

TS_04 Test setup Cloud Federation

In this setup, we will use entirely cloud resources for testing using public and private cloud infrastructures (local/edge resources are emulated by cloud resources), but differently from the previous setup, herein we will adjust a setup from different cloud providers, in order to setup a cloud federation.

Infrastructures: Various local resources/services running in the federated cloud using different data centers; global services running in the public cloud federation.

Scale: We will scale number of VMs/containers, number of providers, size of data to be tested, individual services/units, subsystems and the entire system.

Data and Software for IoT: In this setup, we will use datasets and software from Inter-IoT and other examples for the test as described in previous test setups.

Testing goals: availability, reliability, resource scheduling, resource efficiency, data compatibility, conformance, installation.

TS_05 Test setup Simulation

In this setup, we will adjust an entire simulation environment, where the existing providers, infrastructures, resources and services are completely controlled, by having a certain [min, max] interval. This is a crucial point of the test phase, because it enables a clear comparison between an entirely simulated environment with partial or total real environments.

Scale: Due to the simulation environment facilities, the scaling can be done on almost all the parameters from the previous test setups, including global services, client number, IoT providers, infrastructure type, resource type, data size, services/units, subsystem or entire system test.

Data and Software for IoT: In this setup, we will use fictional datasets and software, which is comparable to the Inter-IoT and real life traces.

Testing goals: availability, reliability, resource scheduling, resource efficiency, data compatibility and conformance.

3.3.9.5 Test description

In the following, we present different testing scenarios. The list is not exhaustive as we are still current at the design phase of the INTER-HINC. The details of FAT plan will be provided in the next reporting period (2018), when we move into the implementation and integration phase. Note that even before performing FAT, we will also have to detail different unit, integration and smoke tests for our software components (which are not reported in this deliverable). Here we just outline possible main scenarios for FAT.

3.3.9.5.1 Scenario 1: Testing Metadata Extraction and Query Interoperability from different IoT providers

Use case metadata extraction

We have data pipelines to obtain and extract metadata from IoT providers to the INTER-HINC. In our design, to be scalable and support interoperability, we use the data pipeline concept to perform metadata extraction. In the pipeline, there will be resource driver, information validator, capability extractor, etc. for extracting IoT provider data to our resource slice information. For this test case, we will need to test the pipeline with different types of IoT providers. The test will examine the interoperability based on APIs of IoT providers and their metadata. Key testing types for Factory Acceptance Testing (Operational acceptance testing) are:

T1.1.1 Metadata Exaction Successful Rate

ID	T1.1.1
Test	Conformance testing w.r.t. metadata extraction
Type	Successful rate
Setup	TS_01, TS_02
Start	IoT services running, all other services running
Req.	
Input	Various metadata about IoT devices capabilities
Output	Events about metadata extraction
Logs	All logs will be stored in Google Storage
Outcome	Successful Rate

T1.1.2 Software performance testing

ID	T1.1.2
Test	Software performance testing w.r.t. the execution of the pipeline
Type	Performance test
Setup	TS_01, TS_02, TS_03
Start	IoT services running, all other services running
Req.	
Input	Various metadata about IoT devices capabilities
Output	Performance/time
Logs	All logs will be stored in Google Storage
Outcome	Execution time

Use case Query Metadata

Clients will query metadata about IoT data provided by different providers before obtaining the data. In this test, we will test the interoperability of metadata about different types of data to make sure that the data delivery is feasible. This is involved in checking delivery protocols, data delivery granularity, interoperability of data broker platforms, data contracts, etc. Key testing types for Factory Acceptance Testing (Operational acceptance testing) are:

T1.2.1 Performance test

ID	T1.2.1

Test	Performance testing w.r.t. time for querying data
Type	Performance test
Setup	TS_01, TS_02, TS_03
Start	IoT services running, all other services running
Req.	
Input	Various metadata about IoT devices capabilities
Output	Performance/time
sLogs	All logs will be stored in Google Storage
Outcome	Execution time

T1.2.1 Compatibility Test

ID	T1.2.2
Test	Compatibility testing w.r.t. data delivery metadata compatibility
Type	Compatibility testing
Setup	TS_01, TS_02
Start	IoT services running, all other services running
Req.	
Input	Various metadata about IoT devices capabilities
Output	Compatibility degree
Logs	All logs will be stored in Google Storage
Outcome	Ratio

3.3.9.5.2 Scenario 2: Testing Middleware and Infrastructure

Use case Infrastructure Deployment and Configuration

INTER-HINC and its services are running in virtual environments. IoT providers interfacing INTER-HINC are also running in distributed sites. One of important tests is to test the deployment of infrastructures, especially for dealing with metadata and data to be delivered by IoT providers through INTER-HINC. Furthermore, the configuration of services will be tested to make sure that the commands for configuration of slides will be reliable and met the expected performance. In this test, we will test time, failure, loss of messages, etc., for deployment and configuration. Key testing types for Factory Acceptance Testing (Operational acceptance testing) are:

T2.1.1 Installation Test

ID	T2.1.1
Test	Installation testing w.r.t. deployment of services
Type	Installation testing
Setup	TS_01, TS_02, TS_03
Start	all INTER-HINC services running, cloud services running
Req.	
Input	Required services to be deployed
Output	Number of services deployed, location, deployment time, successful/failed events
Logs	All logs will be stored in Google Storage
Outcome	Successful rate

T2.1.2 Performance Test

ID	T2.1.2
Test	Performance testing w.r.t. deployment and configuration time
Type	Performance testing
Setup	TS_01, TS_02, TS_03
Start	all INTER-HINC services running, cloud services running
Req.	
Input	Required services to be deployed
Output	Number of services deployed, location, deployment time
Logs	All logs will be stored in Google Storage
Outcome	Performance

T2.1.3 Availability Test

ID	T2.1.3
Test	Availability testing w.r.t. deployment infrastructure
Type	Availability testing

Setup	TS_01, TS_02, TS_03, TS_04, TS_05,
Start	All INTER-HINC services running, cloud services running
Req.	
Input	Required services to be deployed
Output	Number of services deployed, location, deployment time
Logs	All logs will be stored in Google Storage
Outcome	Availability improvement

T2.1.4 Reliability Test

ID	T2.1.4
Test	Reliability testing w.r.t. deployment infrastructure
Type	Reliability testing
Setup	TS_01, TS_02, TS_03, TS_04, TS_05,
Start	All INTER-HINC services running, cloud services running
Req.	
Input	Required services to be deployed
Output	Number of services deployed, location, deployment time
Logs	All logs will be stored in Google Storage
Outcome	Reliability improvement

T2.1.5 Resource scheduling Test

ID	T2.1.5
Test	Resource scheduling testing w.r.t. deployment infrastructure
Type	Resource scheduling testing
Setup	TS_01, TS_02, TS_03, TS_04, TS_05,
Start	All INTER-HINC services running, cloud services running
Req.	
Input	Required services to be deployed
Output	Number of services deployed, location, deployment time
Logs	All logs will be stored in Google Storage

Outcome	Resource scheduling improvement

T2.1.6 Resource efficiency Test

ID	T2.1.6
Test	Resource efficiency testing w.r.t. deployment infrastructure
Type	Resource efficiency testing
Setup	TS_01, TS_02, TS_03, TS_04, TS_05,
Start	All INTER-HINC services running, cloud services running
Req.	
Input	Required services to be deployed
Output	Number of services deployed, location, deployment time
Logs	All logs will be stored in Google Storage
Outcome	Resource efficiency improvement

T2.1.7 Energy efficiency Test

ID	T2.1.7
Test	Energy efficiency testing w.r.t. deployment infrastructure
Type	Energy efficiency testing
Setup	TS_01, TS_02, TS_03, TS_04, TS_05,
Start	All INTER-HINC services running, cloud services running
Req.	
Input	Required services to be deployed
Output	Number of services deployed, location, deployment time
Logs	All logs will be stored in Google Storage
Outcome	Energy efficiency improvement

T2.1.8 Single point of failure[39] Test

ID	T2.1.8

[39] This means, we will test all the services, daemons, parts of the infrastructure that represent a potential single point of failure within a system under test (SUT).

Test	Single point of failure testing w.r.t. deployment infrastructure
Type	Single point of failure testing
Setup	TS_01, TS_02, TS_03, TS_04, TS_05,
Start	All INTER-HINC services running, cloud services running
Req.	
Input	Required services to be deployed
Output	Number of services deployed, location, deployment time
Logs	All logs will be stored in Google Storage
Outcome	Single point of failure improvement

T2.1.9 Maintainability Test

ID	T2.1.9
Test	Maintainability testing w.r.t. Installation testing and deployment infrastructure
Type	Maintainability testing
Setup	TS_01, TS_02, TS_03, TS_04, TS_05,
Start	All INTER-HINC services running, cloud services running
Req.	
Input	Required services to be deployed
Output	Number of services deployed, location, deployment time
Logs	All logs will be stored in Google Storage
Outcome	Maintainability improvement

Use case Data Delivery Testing

IoT providers will deliver data directly to the client or via data brokering services configured by INTER-HINC. In this test, we will test the performance and interoperability of data delivery for the clients. Key testing types for Factory Acceptance Testing (Operational acceptance testing) are:

T2.2.1 Performance Test

ID	T2.2.1
Test	Performance testing w.r.t. data delivery
Type	Performance testing

Setup	TS_01, TS_02, TS_03
Start	all INTER-HINC services running, cloud services running, data services running
Req.	
Input	Types of IoT data to be delivered, time windows
Output	Amount of data, time
Logs	All logs will be stored in Google Storage
Outcome	Transfer rate

T2.2.2 Conformance Test

ID	T2.2.2
Test	Conformance testing w.r.t. data to be obtained
Type	Conformance testing
Setup	TS_01, TS_02
Start	all INTER-HINC services running, cloud services running, data services running
Req.	
Input	Types of IoT data to be delivered, time windows
Output	Events about data received
Logs	All logs will be stored in Google Storage
Outcome	Compatibility degrees (w.r.t. delivery rate, format)

3.3.9.5.3 Scenario 3: Testing Programming APIs

Use case Slice Creation

In this test, we will test programming APIs for clients to test the slice creation. We will consider different parameters for slices and check if the creation is successful or not. Key testing types for Factory Acceptance Testing (Operational acceptance testing) are:

T3.1.1 Performance Test

ID	T3.1.1
Test	Performance testing w.r.t. slice creation
Type	Performance testing
Setup	TS_01, TS_02, TS_03
Start	all INTER-HINC services running, cloud services running

Req.	
Input	Slice specification
Output	Events about slice creation
Logs	All logs will be stored in Google Storage
Outcome	Performance

T3.1.2 Performance Test

ID	**T3.1.2**
Test	Success/Failure testing w.r.t slice creation parameters
Type	Performance testing
Setup	TS_01, TS_02, TS_03
Start	all INTER-HINC services running, cloud services running
Req.	
Input	Different slice specifications
Output	Events about slice creation
Logs	All logs will be stored in Google Storage
Outcome	Successful Rate

Use case Slice Reconfiguration

We will test reconfiguration features of slices. During the test, we will change configuration of INTER-HINC services and will measure the success and performance of reconfiguration. Key testing types for Factory Acceptance Testing (Operational acceptance testing) are:

T3.2.1 Performance Test

ID	**T3.2.1**
Test	Performance testing w.r.t. slice configuration
Type	Performance testing
Setup	TS_01, TS_02, TS_03
Start	all INTER-HINC services running, cloud services running, a slice exists
Req.	
Input	A configuration for an existing slice
Output	Events about slice configuration

Logs	All logs will be stored in Google Storage
Outcome	Time to finish the configuration

T3.2.2 Performance Test

ID	**T3.2.2**
Test	Success/Failure testing w.r.t slice configuration parameters
Type	Performance testing
Setup	TS_01, TS_02, TS_03
Start	all INTER-HINC services running, cloud services running, slices exist
Req.	
Input	Different slice configuration parameters
Output	Events about slice reconfiguration
Logs	All logs will be stored in Google Storage
Outcome	Successful Rate

3.3.9.6 Integration ethics and security

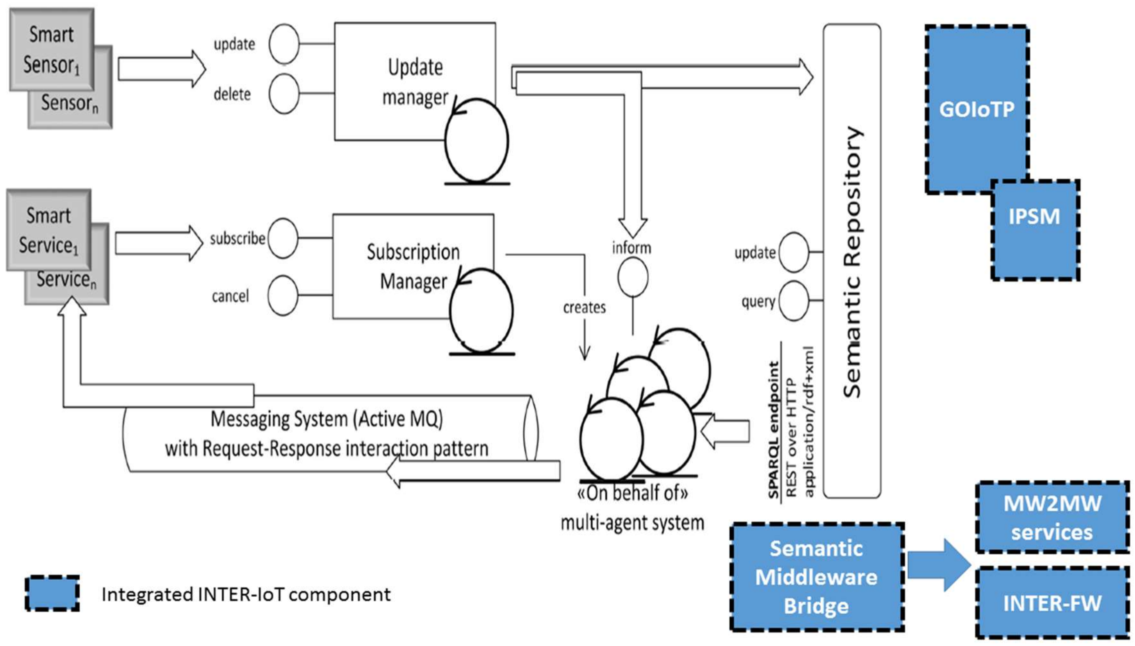
INTER-HINC

Currently we do not have any specific ethics and security.

3.3.10 Third Party: Semantic Middleware

Figure 120 depicts the overall architecture of the Semantic Middleware, focusing on its semantic information, integration and dispatching capabilities. The diagram outlines the components in charge of supporting it: Update Manager (UM) and Semantic Broker (SB), on its turn made up by the Subscription Manager (SM) and the Messaging System (MS) supported by a multi-agent System. Each sensor, after having gathered the information which oversees, affects the knowledge base (**GOIoTP**) hosted on the shared semantic repository (**RDF store**) by updating or deleting semantic assertions. This is done through a web service exposed by UM. On the other hand, information consumers (smart services and sensors) subscribes to the SB, providing their profile of interest.

SM is the component which is always listening on the queue that manages the new subscriptions, leveraging the Apache ActiveMQ (ActiveMQ) messaging system. Whenever SM receives a subscription request from a network client, it activates server-side an agent (the ClientAgent) which takes care of the client interests. Namely, SM records this interest activating an agent in charge of signaling emerging new information to the consumer. If such agent already exists, SM simply notifies the new consumers' interest. In each moment, the consumer can unsubscribe by cancelling the request. Each time a sensor authors new knowledge, the UM informs the SB of the occurred event so that, in a continuous query processing fashion, the SB can evaluate emerging information and notifies it to the consumer through the MS.



Smart Sensor$_1$
Sensor$_n$
update
delete
Update manager
Smart Service$_1$
Service$_n$
subscribe
cancel
Subscription Manager
inform
creates
update
query
Messaging System (Active MQ) with Request-Response interaction pattern
«On behalf of» multi-agent system
SPARQL endpoint REST over HTTP application/rdf+xml
Semantic Repository
GOIoTP
IPSM
MW2MW services
Semantic Middleware Bridge
INTER-FW
Integrated INTER-IoT component

Figure 120: Overall architecture and its interaction with INTER.IoT.

The client starts the interaction with the server through a subscription message containing the description of the information of interest (specified in a query) with the minimum refresh rate in milliseconds, together with a unique identifier of the request (req-id) and a reference (pointer, address, etc.) of the client (client-ref.) to which forward the discovered information.

The query transmitted from the client is expressed through the SPARQL 1.1 syntax and may be a SELECT, ASK or CONSTRUCT that refers to semantic model contained in the repository. The server processes the subscription request and decides whether to accept it. If it is rejected, the repository sends to the client the rejection condition ending the interaction. If it is accepted, at each interval of minimum refresh rate, the server updates the evaluation of the subscribed queries and transmits an information message (inform-result to the client) containing the result of the executed query (query-result) if the result is not empty (SELECT or CONSTRUCT query type) or positive (ASK query type), according to the chosen response format.

The server continues to broadcast type messages (inform-result) as long as one of the following conditions happen:

1. the client deletes the subscription request by the cancellation request (see next section);
2. An error occurs for which the server is no longer able to communicate with the client or to process queries.

All interactions are identified by a unique identifier other than zero (req-id) assigned by the initiator of the protocol and valid for it (client-ref). This allows stakeholders to manage their communication strategies and activities. Moreover, since it can be important to preserve the sequence of the messages, the transport layer has to preserve the order of the messages (reliable transport layer). Thanks to the oneness of the req-id, each client can participate in multiple signaling at the same time. Figure 121 reports the overall workflow of the subscription process.

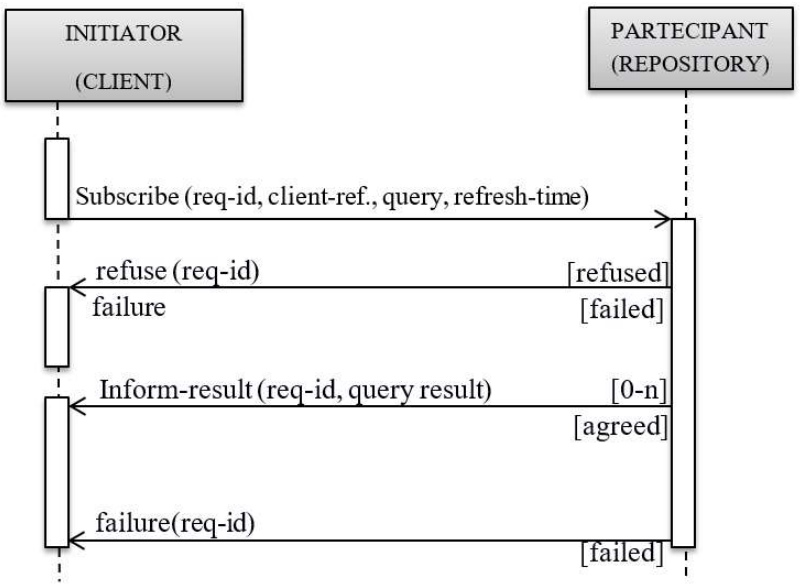


Figure 121: Subscription and notification workflow.

At any time, the client may cancel a subscription request by transmitting a cancel request to the server. In such a request, the parameters *req-id-ref* and *client* identify the interaction to be stopped (Figure 122). The server informs then the client if the interruption succeeded (done) or that it was not possible to break the interaction due to an error (failure).

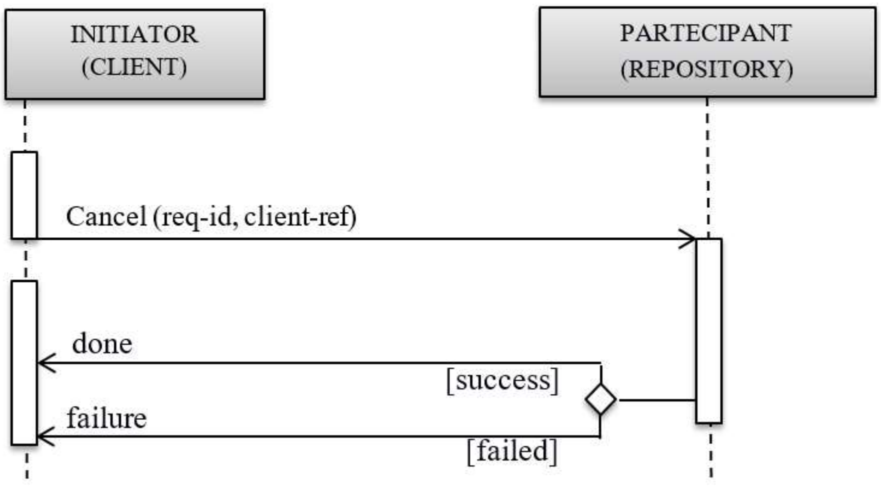


Figure 122: Workflow of the cancellation of the subscription.

3.3.10.1 Integration of IoT framework

This section explores the integration of the *Semantic Middleware* with INTER-IoT components. The idea is implementig a "**Semantic Middleware Bridge**" which handles the connection of our *Semantic Middleware* with all the underlying platforms, with the **MW2MW** services and with the **INTER-FW**. Under these conditions, the Semantic Middleware can be considered as a new platform. In this way, *Semantic Middleware* can ask information requests to the the other underlying platforms. The requests can concern information about specific values. They are expressed composing a message queue as specified in D3.1 (using Kafka protocol). As soon as the Semantic Middleware receives the answer, it elaborates the received the received information. The realization of the platform will be based on "the generic interface which provides a structured template to easily develop new bridges." (D3.1). In addition, the *Semantic Middleware* will expose its functionalities according to the structure provided by the "generic interface".

In addition to **Bridge**, another INTER-IoT components that *Semantic Middleware* will use is **GOIoTP**. Indeed, *Semantic Middleware* is agnostic to the meta-model (TBOX) of the IoT platform ontology, i.e. the behavior of the *Semantic Middleware* does not depend on a specific semantic structure of the ontology. Under these conditions, the *Semantic Middleware* uses the **GOIoTP**, which is taken as global common semantic model that all the devices share. **GOIoTP** will be used to represent general concepts of our scenario and thus a link between our application ontology and **GOIoTP** will be studied and implemented. The ontology model will be stored into the RDF store provided by INTER-IoT infrastructure through the SPARQL engine also provided by the same infrastructure.

In addition, since it is essential to reconcile and mediate the domain ontologies handled by applications and devices and the core ontology shared within the whole platform, it should be evaluated the integration of *Semantic Middleware* with **Inter Platform Semantic Mediator (IPSM)** component proposed in INTER-IoT.In particular, the latter allows the alignment of the commonalities (overlapping concepts) between the domain ontology (RDF model) used in a specific IoT platform and the core ontology (RDF model) defined within the INTER-IoT. It will be necessary to define the rules of the mapping for each domain ontology that has to be aligned with the core ontology. Thus, **IPSM** will be exploited to translate and link the meta-model (TBOX) of our application ontology with **GOIoTP**.

Regarding the middleware protocol used to implement *Semantic Middleware*, the idea is leveraging **MQTT** among the protocols already supported by INTER-IoT; a proper connection with the **Dispatcher** will be investigated.

Components	Interface overview of the used IoT components	Tests
Bridge	The generic common interface as a structured template to easily develop new bridges (D3.1)	
GOIoTP	• Various entities defined in the Semantic Model	• All
IPSM	• The integration of this component is still under analysis	• All
MQTT \ Dispatcher	• The integration of this component is still under analysis	• All

Table 96. Table containing the components and interface overview of the used IoT components and the tests that will test these IoT components

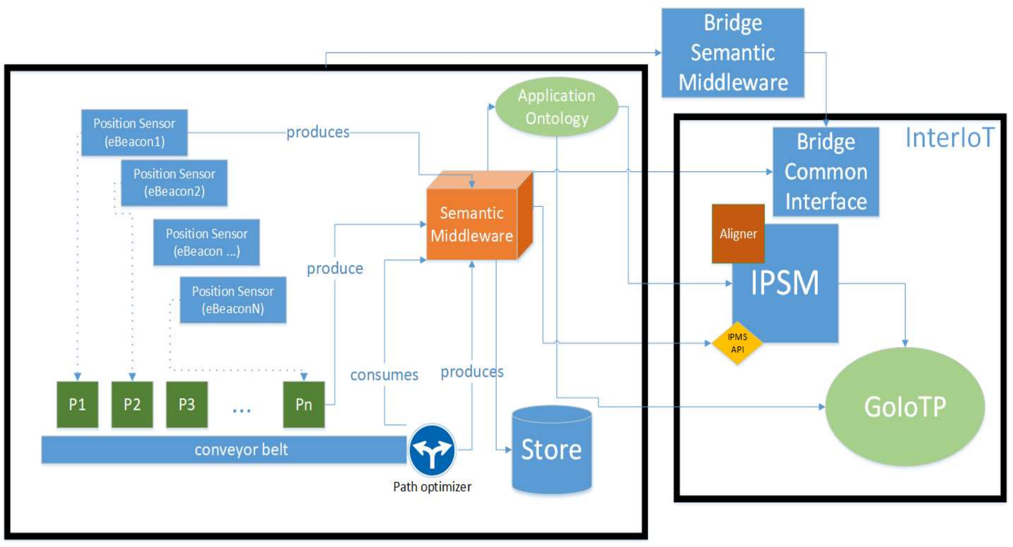


Figure 123. **Integration of the Semantic Middleware with the IoT framework**

3.3.10.2 Deliverables and version overview

The following table contains a deliverable list which need to be signed of before FAT testing commences.

ID	Description	Check
Documents		
1	Validation and Test reports of the Semantic Middleware components	
2	Validation and Test reports of the Pilot system components	
Hardware		

4	Workstation server \ Cloud (server which hosts Semantic Middleware server + database)	
5	PC client (PC which hosts Semantic Middleware client)	
Tools		
7	Semantic Middleware	

Table 97: Deliverable checklist

The following table shows the software components and version of which the system release version consists of.

ID	Description	Version	Check
IoT Physical Gateway			
1	MQTT ***(integration is under analysis)***		
IoT Virtual Gateway			
2	Bridge		
3	GOIoTP		
4	IPSM		
5	Platform Request Manager (Deliverable D3.1)		
6	Data Flow Manager (Deliverable D3.1)		
Semantic Middleware			
7	Semantic Broker		
8	Update Manager		
9	Client Semantic Middleware Library		
10	RDF store (Stardog[40] free version)		
11	Publish-Subscribe middleware (ActiveMQ[41])		
12	Semantic data model		

Table 98: Component version overview

3.3.10.3 Requirements, scenarios and use cases

The following table provides an overview of the relation between the requirements and the test(s) that validate their implementation.

ID	Description	Covered by
42	Heterogeneous information representation	T34.62.7
75	The interaction between IoT endpoints may follow M2M concept	T34.62.1, T34.62.2, T34.62.3, T34.62.4
96	Enable (automated or semi-automated) linking of relevant data models	T34.62.7
163	***Design support for semantic interoperability (this requirement has been deleted in the new template)***	T34.62.1, T34.62.2, T34.62.3, T34.62.4
178	Inter Platform Semantic Mediator provides data and semantic interoperability functionality	T34.62.6
179	Inter Platform Semantic Mediator supports platform communication	T34.62.6
180	Syntactic and semantics interoperability - Data format and semantics translation	T34.62.6
237	API Middleware for interoperability between different platforms	T34.62.5

[40] https://www.stardog.com/
[41] http://activemq.apache.org/

270	API allows subscription to data streams/queues	T34.62.1, T34.62.2, T34.62.3, T34.62.4
282	Map publish/subscription between platforms	T34.62.5

Table 99: Requirements vs test mapping

The following table provides an overview of the relation between the scenarios and the test(s) that validate their implementation.

ID	Scenario name	Covered by
32	Third party developer using INTER-FW to access data from two different platforms	T34.62.5 T34.62.6 T34.62.7
33	Heterogeneous Platforms Methodology-driven Integration	T34.62.5 T34.62.6 T34.62.7
34	Position and Optimization of the pallets ***(New scenario)***	T34.62.1, T34.62.2, T34.62.3, T34.62.4

Table 100: Scenario vs test mapping

3.3.10.4 Test environment

Introduction

To test the functionality of the **Pallet scenario within CPPS LAB** in combination with the IoT framework a representative test system is needed. The test system needs to approach the "real world" as much as possible. The pilot setups must be recreated and proven. This chapter will describe this environment and the used hardware, software, tools and platforms.

Test environment

A valid business scenario for the pilot of the Semantic Middleware is offered by the Cyber Physical System (CPS) Lab located at the ITIA-CNR's headquarters in Milan, where recently a cutting-edge CPS system has been developed in order to monitor and optimize the position of various pallets along a conveyer belt within an industrial scenario (Figure 124). The idea behind this system is that processes in a production facility can be optimized with the aid of indoor localization and route analysis. In addition, an asset tracking solution will make it possible to retrieve location and nearest available services provided by the servitization of the factory. The pallets contain hardware components on which operations have to be performed at different working stations following a specific order. Moreover, the working stations can perform different operations (e.g. drilling, milling, etc.), and the path of the pallet is optimized by a simulation application (Optimizer) in order to send it to the closest available working station.

3.3.10.5 Test setups, tools, hooks and probes

This paragraph describes the test setups, tools, hooks and probes used in the by this document defined tests.

A test setup defines a setup used for a test, and describe the used system parts, interface etc.

Tools define the used COTS tooling that will be used during testing e.g. Logic analyzers, packet sniffer like e.g. Whireshark, developed scripts, etc.

Hooks are used to inject data or control parts of the system used to get the system to handle a needed scenario. This can be hidden, debug or service interfaces which contain the needed functionality.

Probes are used to gather logging from parts of the system under test to provide system feedback and to prove that the system is handling the scenario as it should.

The following paragraphs will define the used setups, tools, hooks and probes which will be referred to by the test descriptions.

TS_01 SemanticMiddlewareComponents

The server components, which are Update Manager (UM) and Semantic Broker (SB), are deployed on a server. The client component (Client Semantic Middleware Library) is deployed on the client application (typically included in its setup package).

TS_02 IPSM

The IoT IPSM is up and running.

TS_03 GOIoTP

The GOIoTP ontology.

TS_04 MW2MW

The MW2MW layer.

TT_01 RDF store

An RDF store (we propose the version trial of Stardog[42], but any other kind of RDF store can be used).

TT_02 ActiveMQ

A message oriented middleware. We propose ActiveMQ[43].

TH_01 Semantic Model

A Semantic Model, reported as owl file, which can be imported by the RDF store. This model represents the knowledge concerning the scenario environments.

The Semantic model is paired with a list of SPARQL queries to subscribe a change into the knowledge base and a list of SPARQL queries that updates the knowledge base in correspondence of the subscription.

In Section 3.3.10.6.1 it is reported some hints about the semantic model used in these FAT and about involved SPARQL queries (e.g. Query 1, etc.**)**.

TP_01 FeedbackWithinGUI

Feedback of the tests will be shown within the GUI of the client applications.

TP_02 MockUpApplicationCallingBridge

This application sumulates the call to the Semantic Broker Bridge.

TP_03 LogFile

A log file is provided which reports the auditing of some operations.

3.3.10.6 Test description

Test output log file (TP_01 LofFile): Folder "`LOG`", prefix TestName_

[42] http://www.stardog.com/

[43] http://activemq.apache.org/

3.3.10.6.1 S34: Position and Optimization of the pallets

The sensors monitoring the pallet position will play the role of publisher as they will send the information concerning the pallet position through the middleware (Step 1); this information is expressed under the form of a SPARQL UPDATE. Also the working stations will publish their availability status (Step 2). This information will be then consumed by the simulation tool (Optimizer) which has previously subscribed to the changes applied to the pallet position (Step 3) and the availability status of the working stations (using a proper SPARQL query) with the goal to identify the optimized pallet route. In addition, the information concerning the route is then published (Step 4) and in its turn consumed by the IoT actuators which allow to change the route of the pallets along the conveyor belt (Step 5).

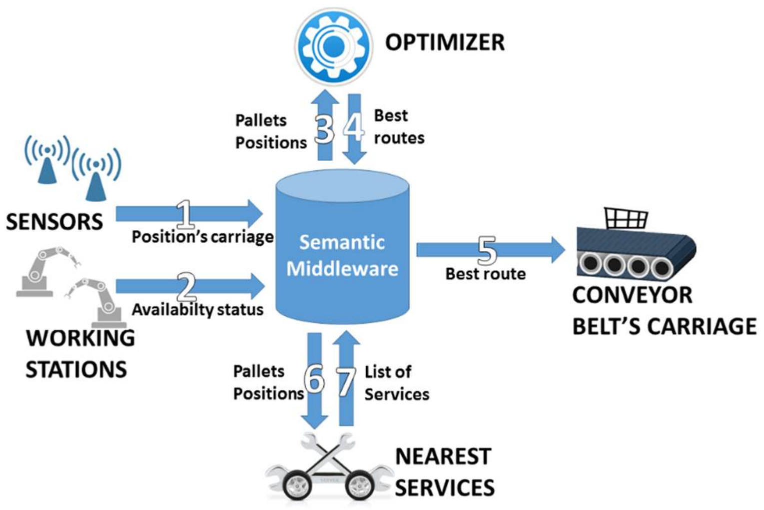


Figure 124. Workflow of the scenario

The use cases reported in the following sections involve various components integrated with the *Semantic Middleware*. The components are the following:

- The Semantic Model. The Semantic model used in these FATs represents knowledge concerning sensors and their corresponding measures, pallets and their real and optimized positions.
- An application (*Virtual sensor*) that mocks and simulates the behavior of an Ebeacon sensor tracking the position of the pallets. These sensors will publish, through an UPDATE SPARQL (**Query 1**), the position of the pallets P1 and P2 within the knowledge base, according to a proper domain ontology.
  To support the FAT, *Virtual Sensor* provides a GUI that allows to set a position for a specific pallet, thus generating the corresponding UPDATE SPARQL query.
- *Optimizer*, a simulation tool which uses the pallet position and the availability status of the working stations with the goal to identify the optimized pallet route. It consumes data published by the various tracking sensors, while it publishes optimized routes for the pallets.
  To support the FAT, *Optimizer* is virtualized and provides a GUI that allows to start to listen and consume information concerning position of a specific pallet, thus generating the corresponding SELECT SPARQL query (**Query 2**). In addition, *Virtual Optimizer* allows to set the new best route for the pallet, simulating the behavior of the real application.
- *Virtual carriages* (simulating the real carriages) each one transporting a pallet. They consume data published by the simulator in order to follow a specific route.

To support the FAT, *Virtual carriage* provides a GUI that allows to start to listen and consume information concerning route for a specific pallet, thus generating the corresponding UPDATE SPARQL query (**Query 3**).

3.3.10.6.2 U62 – Device (sensor) triggers information

A device, typically a sensor, triggers an event sending determined information to the gateway in order to be stored on the platform Cloud or server or in order to generate a response for an actuator (being handled by the rules engine).

This use case involves these requirements: [75], [163] and [270].

T34.62.1 Information published by Virtual Sensor are persisted

Test	**Virtual Sensor publishes information and this information is persisted into the RDF store**
Type	System testing
Setup	Need test setup TS_01 SemanticMiddlewareComponents (Query 1, etc.) Need test tool TT_01 RDF store Need test tool TT_02 ActiveMQ Need test hook TH_01 Semantic Model (Query 1, etc.)
Start	Information to be published are not yet persisted
Req.	[75], [163], [270]
Input	Enable the sensor within range of the physical gateway
Output	• The result of a SPARQL query on the RDF store
Outcome	Pass / Fail

Test output:

- Access the RDF store and verify through a SPARQL query (SELECT) if the information has been stored

T34.62.2 Information updated by Virtual Sensor are received by the subscribed clients

Test	**Information updates are received by the subscribed clients**
Type	System testing
Setup	Need test setup TS_01 SemanticMiddlewareComponents Need test tool TT_01 RDF store Need test tool TT_02 ActiveMQ Need test hook TH_01 Semantic Model (Query 1, etc.)

Start	Information to be published are not yet persisted A client (Optimizer) is subscribed to the updated information
Req.	[75], [163], [270]
Input	Information published by Virtual Sensor. It concerns the position of the pallet.
Output	• Check if the subscriber (Optimizer) receives the information updates (postion of the pallet). It has to receive the information updated by the Virtual Sensor. • Check if other subscribers (Virtual Carriage), which are not subscribed to the updated information, does not receive the updated information.
Outcome	Pass / Fail

Test output:

- Feedback of the tests will be shown within the GUI of the client applications (TP_01 FeedbackWithinGUI).
- Also a log file can be provided (TP_03 LofFile)

T34.62.3 Information updated by Optimizer are received by Virtual Carriage

Test	**Information updates are received by the subscribed clients**
Type	System testing
Setup	Need test setup TS_01 SemanticMiddlewareComponents Need test tool TT_01 RDF store Need test tool TT_02 ActiveMQ Need test hook TH_01 Semantic Model (Query 2 and Query 3)
Start	Information to be published are not yet persisted A client (Virtual carriage) is subscribed to the updated information
Req.	[75], [163], [270]
Input	Information published by Optimizer (new route of the pallet).
Output	• Check if the subscriber (Virtual carriage) receives the information updates. The have to receive the updates triggered by the Optimizer changes.
Outcome	Pass / Fail

Test output:

- Feedback of the tests will be shown within the GUI of the client applications (TP_01 FeedbackWithinGUI).
- Also a log file can be provided (TP_03 LofFile)

T34.62.4 Updates concerning information on which no client is subscribed

Test	Updates concerning information on which no client is subscribed
Type	System testing
Setup	Need test setup TS_01 SemanticMiddlewareComponents Need test tool TT_01 RDF store Need test tool TT_02 ActiveMQ Need test hook TH_01 Semantic Model (Query 1, etc.)
Start	Information to be published are not yet persisted A couple of clients (Virtual carriage and Optimizer) are subscribed to various information. The latter are not linked with the updated information
Req.	[75], [163], [270]
Input	Information published by Optimizer
Output	• Check if the subscribers (Virtual carriage) receives or not the information updates: They do not have to receive the updates.
Outcome	Pass / Fail

Test output:

- Feedback of the tests will be shown within the GUI of the client applications (TP_01 FeedbackWithinGUI).
- Also a log file can be provided (TP_03 LofFile)

T34.62.5 The connection with the Semantic Middleware Bridge

This is the TEST of INTEGRATION with IoT components.

Test	Connection with the Platform Request Manager
Type	TEST of INTEGRATION with IoT components
Setup	Need TS_04 MW2MW Need TP_02 MockUpApplicationCallingBridge
Start	A request of information to the IoT underlying platforms is sent.
Req.	[237], [282]
Input	The request to
Output	Check if the request is succeful
Outcome	• Pass / Fail

Test output:

- Feedback of the tests will be shown within the GUI of the client applications (TP_02 MockUpApplicationCallingBridge).

T34.62.6 The ontology alignment test through IPSM

This is the TEST of INTEGRATION with IoT components.

Test	**The ontology alignment test between our Application Ontology (AO) and the GOIoTP ontology will be performed according to the general structure of the INTER-IoT alignment format (also called IPSM alignment format)[44]. The alignment element describes a uni-directional set of translation rules comprised of independent mapping cells, each of which has an "input" and "output" entity descriptions. Elements <onto1> and <onto2> describe respectively the AO ontology and the GOIoTP ontology of the alignment, by giving their URIs and specifying the formalism used for their definition (in our case AO adopt the OWL Lite formalism). The following listing shows an example of the alignment passed to the IPSM Aligner service within the InterIoT platform:**
	`<Alignment name="align_name" version=" align_version" creator="align_creator" description="align_desc">`
	`    <onto1> {AO Ontology info} </onto1>`
	`    <onto2> { GOIoTP ontology info } </onto2>`
	`    <steps>`
	`        <step order="1" cell="cell_1"/>`
	`        <step order="2" cell="cell_2"/>`
	`        …`
	`        <step order="N" cell="cell_N"/>`
	`    </steps>`
	`    <map>`
	`        <Cell id="cell_id">`
	`            <entity1> { AO RDF pattern } </entity1>`
	`            <entity2> { GOIoTP RDF pattern } </entity2>`
	`            <transformation>`
	`                { functional constraints }`
	`            </transformation>`
	`            <filters> { datatype constraints } </filters>`
	`            <typings> { typing info } </typings>`
	`        </Cell>`
	`        {…}`
	`    </map>`
	`</Alignment>`
	Each cell will represent a match from <entity1> within our Application ontology into <entity2> from GOIoTP. Both entities are valid RDF graphs (presented in the RDF/XML serialization). We do not plan to use any transformation and function, as we perform simple syntactical and conceptual matching between the entities.

Type	TEST of INTEGRATION with IoT components

[44] Maria Ganzha et al., *"Alignment-based semantic translation of geospatial data"*

Setup	Need test setup TS_02 IPSM Need test hook TH_01 Semantic Model (a subset of this model must be aligned)
Start	Invoking the proper function of IPMS Aligner and passing it the mapping in the form presented above
Req.	[178], [179], [180]
Input	An alignment element

Output	We expect to obtain a result similar to the following from IPMS in order to accept the alignment test:
	`<map><Cell>`
	`<entity1 rdf:resource="http://www.opengis.net/gml/Point"/>`
	`<entity2 rdf:resource="http://www.w3.org/2003/01/geo/wgs84_pos#Point"/>`
	`<measure rdf:datatype="http://www.w3.org/2001/XMLSchema#float">0.99</measure>`
	`<relation>=</relation>`
	`</Cell></map>`
	The correct response returns the URI of the entities that have been aligned and two fundamental information: the logical relation existing between them (<relation>) and the relative confidence of such relation (<measure>).

Outcome	Pass / Fail

Test output:

- The result is reported in the GUI of the proper API provided by IPSM

T34.62.7 Ontology import Text

This is the TEST of INTEGRATION with IoT components.

Test	**GOIoTP ontology are imported within the Application Ontology used to represent knowledge for the scenario S34.**
Type	TEST of INTEGRATION with IoT components
Setup	Need test setup TS_03 GOIoTP Need test tool TT_01 RDF store Need test hook TH_01 Semantic Model
Start	The following subsequent steps will be carried out: Add the import directive in the AO ontology directed to the GOIoTP ontology, so that all the statements of the latter are imported in the former ontology; After the concept alignment has been executed between AO and GoIoTP, the alignment results are used in order to create logical relation axioms within the integrated ontology (e.g., equivalentClass axioms, subClassOf, etc.).
Req.	[42], [96]

Input	Instantiate an ontological individual as an instance of a specific class of the AO ontology, which is equivalent to a class imported from the GoIoTP ontology.
Output	• Test if ontological individual inherits the features of the equivalent class.
Outcome	Pass / Fail

3.3.10.7 Test outcome overview

Test outcome

The following table will provide an overview of the test result of all the performed tests in this FAT.

Test	Description	Outcome
T31.62.1	Information published by Virtual Sensor are persisted	Pass / Fail
T31.62.2	Information updated by Virtual Sensor are received by the subscribed clients	Pass / Fail
T31.62.3	Information updated by Optimizer are received by Virtual Carriage	Pass / Fail
T31.62.4	Updates concerning information on which no client is subscribed	Pass / Fail
T31.62.5	(TEST of INTEGRATION with IoT components) Connection with the Bridge	Pass / Fail
T31.62.6	(TEST of INTEGRATION with IoT components) The ontology alignment test through IPSM	Pass / Fail
T31.62.7	(TEST of INTEGRATION with IoT components) Ontology import Text	Pass / Fail
FAT Outcome		**Pass / Fail**

Table 101: Test outcome overview

3.3.10.8 Integration ethics and security

Introduction

During the Ethical Review it was requested that ethics had to be included within the project scope. This section addresses recommendations and proposed solutions to the ethical and security points of the integration.

For each pilot the ethics is discussed in paragraphs 8.2 until 8.5. The security aspects of each layer is discussed in paragraph 8.7 and 8.8.

The information for the pilots for both ethics and security comes from the partners and may be included in other documents as well.

Semantic Middleware

The various steps needed for the execution of these FAT are not connected with major ethical concerns. In particular, the components that make up the proposed semantic middleware and data sets managed in these tests do not deal with sensitive personal data. Therefore, just a limited attention should be given to the drawn of a code of ethics.

Under a generic perspective, Semantic Middleware allows a new way of combining and integrating different data streams. As analyzing and acting on insights from these data can introduce new classes of risks of unethical or even illegal use of insights, it is necessary to tackle ethical challenges that can emerge, analyzing a proper risk mitigation.The idea is creating a code of data ethics leveraging the Universal principles for data ethics—Guidelines introduced in [45]

Concerning the security, no particular attention is required. RDF STORE and Semantic Middleware expose an authenticated access and for this reason test credentials will be provided to access RDF store and to access Semantic Middleware.

[45] Accenture Labs: OnLine "Building digital trust: The role of data ethics in the digital age",Available at: https://www.accenture.com/t20160613T024441Z__w__/us-en/_acnmedia/PDF-22/Accenture-Data-Ethics-POV-WEB.pdf

3.3.11 Third Party: SecurIoTy

Concerns about information security are one of the main reason for companies and private individuals not to adopt cloud services and to be sceptical about IoT systems. On top of concerns regarding physical- and cyber-attacks, international corporations additionally carry legal attacks in their threat model. However, cloud services are essential in improving efficiency and cost structures.

SecurIoTy adresses that gap. SecurIoTy combines a number of security mechanisms to protect data and to address all relevant security dimension such as confidentiality, integrity and availability. We use CloudRAID, fragmentation and encryption. CloudRAID means that data is fragmented, the fragments are encrypted and the encrypted fragments are redundantly distributed to multiple independent storages. SecurIoTy is storage agnostic, i.e. the data fragments may be distributed across multiple jurisdictions adding additional security.

SecurIoTy solves security and compliance issues when sending and sharing data via public networks like the internet and when storing data in cloud services, thus enabling companies to use cloud based IoT services, which they would not use without SecurIoTy protection.

SecurIoTy is crypto proxy technology and as such won't interfere with the user experience or with processes. SecurIoTy users will get to keep their established usage pattern and processes and also keep their legacy infrastructure. SecurIoTy aims to integrate seamlessly. SecurIoTy can be operated off-premise, on-premise or hybrid.

SecurIoTy is based on DocRAID® - a storage system which offers distributed, secured storage and data transfer. To operate SecurIoTy we maintain a geo-redundant high availability computing cluster. We operate storage capacities spread among different European computing centres, among others in Germany and France. SecurIoTy provides highly secure storage based on the principles of fragmentation and encryption. This means no one (1) storage knows all the information to recompile a document.

Within the INTER-IoT framework we identify

(1) technical,
(2) legal and
(3) organizational

challenges. In the technical category, we can further identify challenges at the

(1) networking,
(2) middleware,
(3) application,
(4) interoperability and
(5) security issues.

In the current version of SecurIoTy we focus on and test requirements from these categories:

Category		Tested in current version
Technical	networking	Yes
	middleware	No
	application	Yes

	interoperability	No
	security	Yes

Table 102: Test categories.

SecurIoTy offers HTTP(S) interfaces and offers interfaces to connect to cloud storage services which are operated by AvailabilityPlus (DocRAID® CloudRAID).

From the architecture point of view, we have established and tested these components:

Category		Tested in current version
Distributed storage	Storage 1	Yes
	Storage 2	Yes
	Storage 3	Yes
Key Storage	Local storage	Yes
	HSM – hardware security module	No
Controller	One controller at one site	Yes
	Multiple distributed controllers at multiple sites	No
Frontend	HTML	No
Secure Gateway	Gate Keeper	Yes
	Load Balancer	No
	Firewall	No
Frontend access	HTTPS	Yes
	WebDAV	No

Table 103: Tested components.

The following paragraph gives a brief introduction of the implemented security measures and potential vulnerabilities.

Security Level	Use Case	Measures / Best Practice	Potential Vulnerability
Low to very high	All	Files are encrypted by AES256	weak password, security is directly related to the strength of the password
Very high	All	Two different random number generators are used. The	It was reported that the random number generator

		random number generator is modularized; upon request customer specific modules can be used.	Dual_EC_DRBG contains a potential backdoor.
Medium	Single User	Strong user generated password and Keyfiles	weak password, keyfiles stored on local machine
High	Single User	User generated strong password and Keyfiles; Keyfiles stored on external device	weak password, social engineering
Low to very high	Multi User	Admin generated password and Keyfiles	weak password, keyfiles stored on local machine
Low to very high	Multi User	Master Key File must be stored in safe place	Master Key File in an unsecure place
Medium	Multi User	Secret keys to access files are stored by default in the Windows key container	Windows key container can potentially be hacked or contain backdoors
Very high	Multi User	Secret keys to access files are stored on an external protected device, e.g. crypto stick	social engineering
Very high	Multi User	Secret keys to access files in a workspace are exchanged based on the Diffi-Hellman key exchange, i.e. perfect forward secrecy	Currently no backdoor known to hack Diffi-Hellman
Medium	Multi User	User is activated by admin after request. For very high security user verification is required.	Workspace file could be intercepted, e.g. if sent by email. User verification by digital handshake supports user verification.
Very high	Multi User	Digital handshake for user verification.	A man-in-the-middle attack would generate an additional request visible to the admin

Table 104: Security measures.

3.3.11.1 Integration of IoT framework

The proposed approach will complement the INTER-IoT Architecture and will provide industry standard interfaces to integrate security as necessitated by the respective application. A high degree of interoperability is achieved by adhering to standard protocols (HTTPS(S), WebDAV, REST, TCP) and by integration of widely used cloud services. In contrast to current approaches to IoT security which mainly focus on single aspects of IoT security, SecurIoTy provides a single framework to cover scalable security from the device level to the application level and which covers all dimensions of security such as confidentiality, integrity and availability (CIA). Put to work in the logistics use case (INTER-logP), SecurIoTy will push the envelope of IoT security well beyond the state of the art.

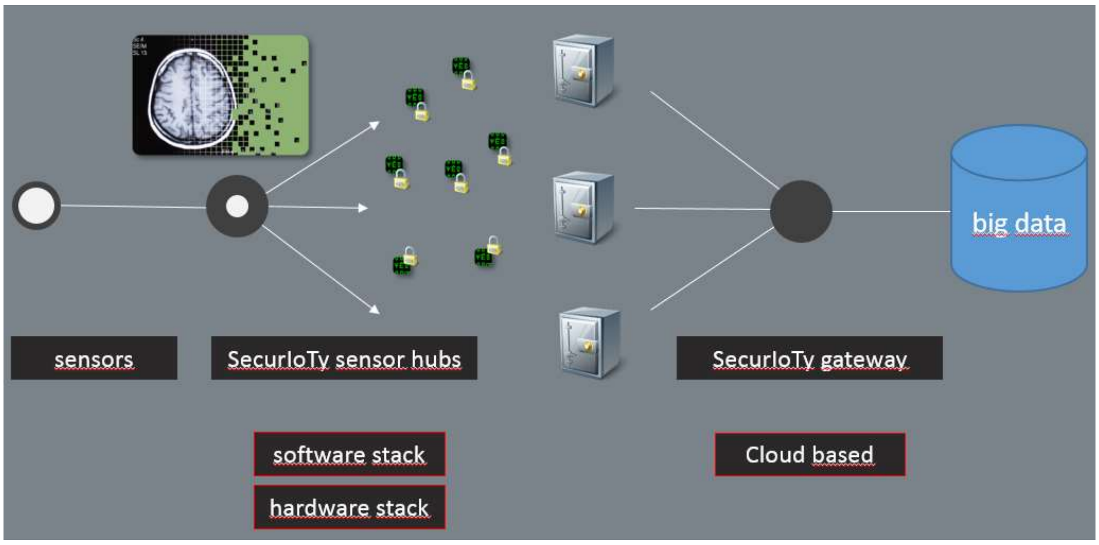


Figure 125: SecurIoTy overview.

The DocRAID® CloudRAID (seeFigure 125) proxy can be deployed as a sensor hub, collecting data from sensors directly or alternatively can be set up as a gateway receiving data from sensor hubs and deliver that data to a data storage. In this project we will assume that DocRAID will be set up as a gateway.

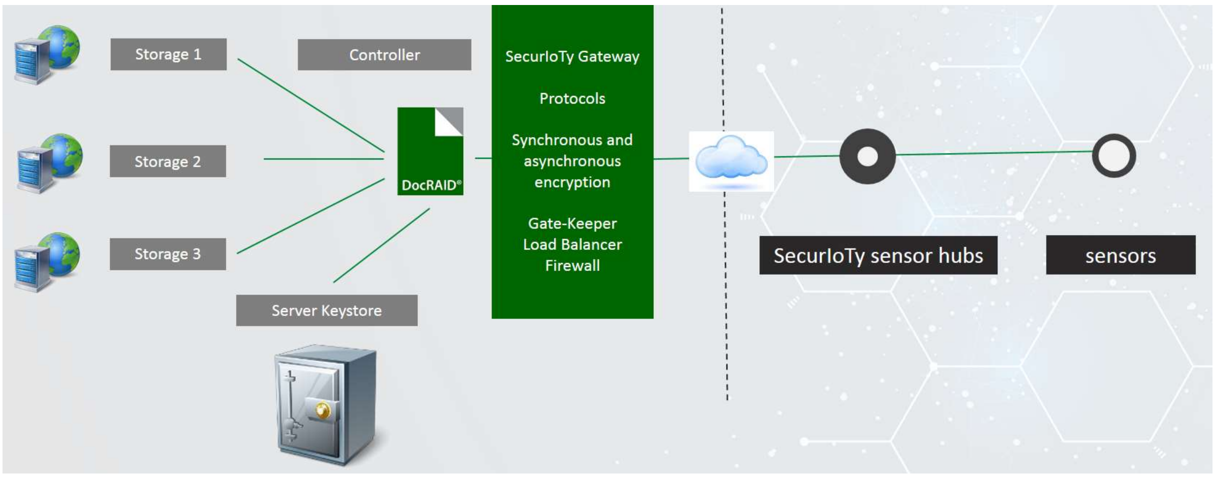


Figure 126: SecurIoTy architecture with the DocRAID crypto proxy

The DocRAID crypto proxy works in three phases:

1. Fragmentation

Data is sent through a shredder and fragmented to pieces.

2. Encryption

Each fragment is encrypted using AES256. Key exchange optionally via Diffi-Hellman.

3. RAID distribution

Encrypted fragments are redundantly distributed by the DocRAID® algorithm, no one (1) storage knows all fragments. Distribution across geographies and jurisdictions, keep legacy infrastructure.

For discussion: at this stage of the project it is to determine what components will communicate with SecurIoTy and when.

3.3.11.2 Deliverables and version overview

The following table contains a deliverable list which need to be signed of before FAT testing commences.

ID	Description	Check
Documents		
1	Validation and Test reports of the IoT system components	
2	Validation and Test reports of the Pilot system components	
Tools		
7	Burp Suite	

Table 105: Deliverable checklist

The following table shows the software components and version of which the system release version SecurIoTy 17.11.0201 consists of.

ID	Description	Version	Check
Distributed storage			
10	Storage Provider 1	17.09.1102	
20	Storage Provider 2	17.09.1102	
30	Storage Provider 3	17.09.1102	
Key Storage			
40	Local Storage Provider	17.09.1102	
Controller			
50	DocRAID controller	17.11.0201	
Secure Gateway			
60	Gate Keeper	17.01.3008	
Frontend provider			
70	HTTPS	15.04.269	
80	WebDAV	15.04.269	

Table 106: Component version overview

3.3.11.3 Requirements, scenarios and use cases

The following table provides an overview of the relation between the requirements and the test(s) that validate their implementation.

ID	Description	Covered by
API		
243	Gateway access API	TC200010, TC200020 TC200030, TC200040
264	API allows create/update/remove users	TC200090, TC200100
Interoperability		
56	Secure synchronization	TC300000, TC300010
Performance		

72	Communication should be done using protocols that are efficient in terms of amount of exchanged information over message size	tbd
Security		
27 30	System security	TC401010, TC401020 TC401030, TC401040 TC401050, TC401060 TC401070
28 37	System privacy	TC401000
95	Robustness, resilience and availability	TC403000, TC403010
98	Data provenance	
261	A user knows its permissions	TC409000, TC409005
263	Access to personal data needs to be previously authorized	TC409010, TC409020 TC409030, TC409040 TC409050, TC409060 TC409070, TC409080 TC409090, TC409100 TC409110
Non-functional requirements		
47	API for third-party developers	TC200050, TC200060
58	Auditability and Accountability	TC500010, TC409060 TC500030, TC500040
60	AutoLogin	TC200070, TC200080
63	Provision of authentication credentials	TC406000, TC406010 TC406020, TC406030 TC406040
68	Logging	TC500050
69	Confidentiality, Avoid data falsification or disclosure	TC404000, TC403000 TC401000
94	Supports multiplatform	TC300020
Architecture		
36	Scalability. Computing resources	TC100010, TC100020 TC100030, TC405000 TC405010, TC405020

Table 107:* *Requirements vs test mapping

3.3.11.4 Test environment

Introduction

To test the functionality of SecurIoTy in combination with the IoT framework, a representative test system is needed. The test system needs to approach the "real world" as much as possible. The pilot setups must be recreated and proven. This chapter will describe this environment and the used hardware, software, tools and platforms.

Test environment

This paragraph describes the test environment and the complete system setup used during this FAT. The test system contains these elements:

(1) DocRAID controller
(2) Secure gateway
(3) Keystore
(4) 3 storages

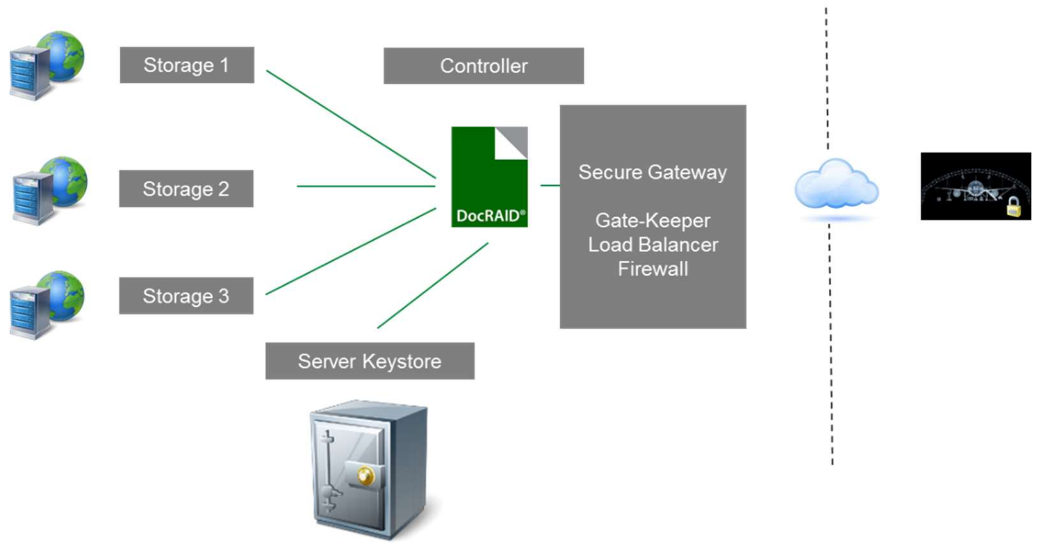


Figure 127: test architecture.

In the test scenarios the controller and the three (3) storages reside on one (1) physical server. The test environment is set up using these components: The secure gateway resides on a separate server.

The server side:

- CPU: 4x Intel Xeon E5-26xx (Sandy Bridge) @ 2.1 GHz
- RAM: 16 GB
- Ethernet: Red Hat VirtIO Ethernet Adapter
- Operating systems:
    - Windows 2012 R2 Server x64

The secure gateway:

- CPU: 4x Intel Xeon E5-26xx (Sandy Bridge) @ 2.1 GHz
- RAM: 16 GB
- Operating systems:
    - Debian 9.2
    - Nginx community edition

Libraries

- Microsoft .net environment 4.51

No additional libraries necessary.

The bug reporting process is defined in the following chart:

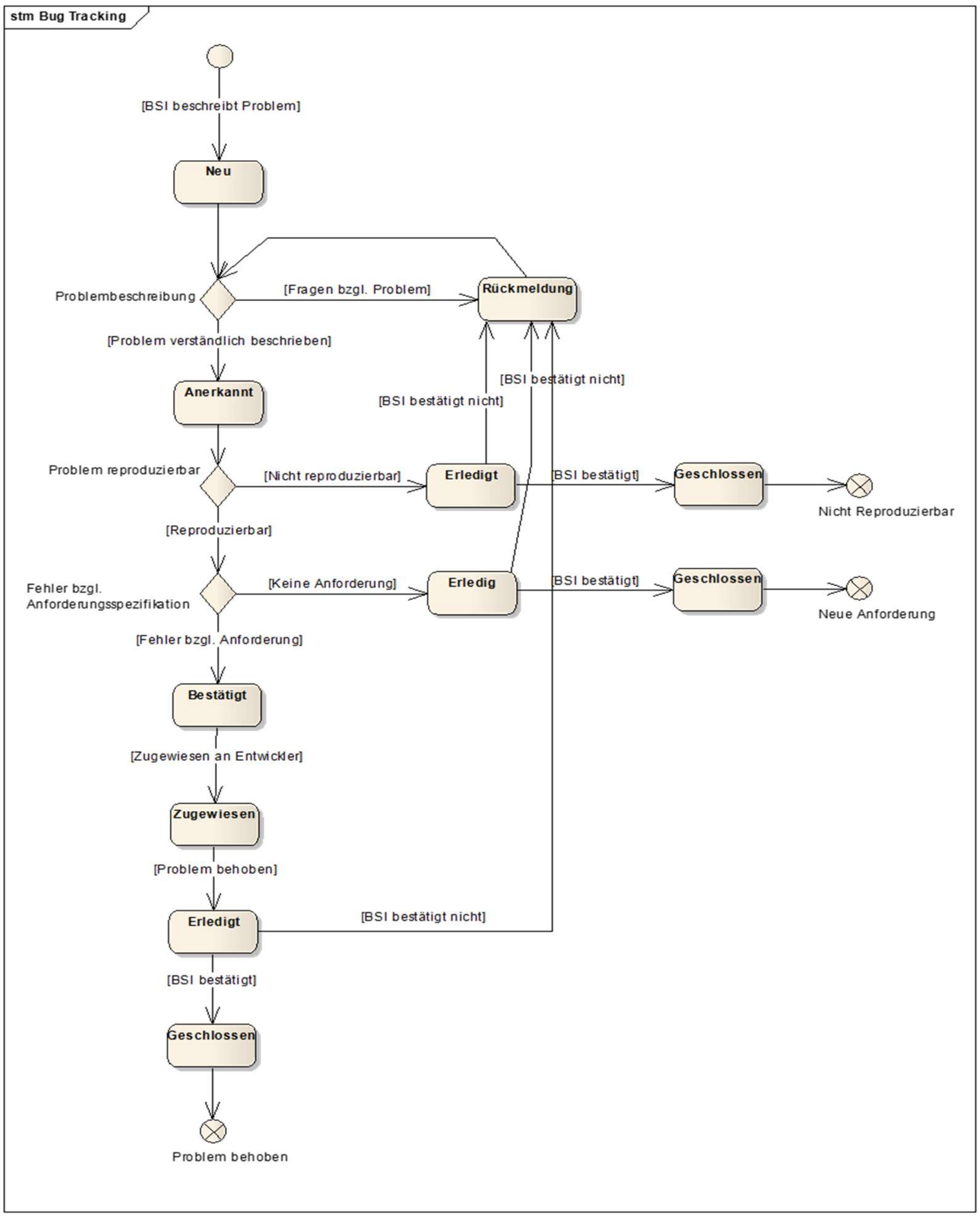


Figure 128: Bug reporting process

3.3.11.5 Test setups, tools, hooks and probes

This paragraph describes the test setups, tools, hooks and probes used in the by this document defined tests.

A test setup defines a setup used for a test, and describe the used system parts, interface etc.

Tools define the used COTS tooling that will be used during testing e.g. Logic analyzers, packet sniffer like e.g. Whireshark, developed scripts, etc.

Hooks are used to inject data or control parts of the system used to get the system to handle a needed scenario. This can be hidden, debug or service interfaces which contain the needed functionality.

Probes are used to gather logging from parts of the system under test to provide system feedback and to prove that the system is handling the scenario as it should.

The following paragraphs will define the used setups, tools, hooks and probes which will be referred to by the test descriptions.

TS_01 Test setup

SecurIoTy acceptance test includes sections covering

(1) Acceptance criteria
(2) Severity and priority of bugs
(3) Reporting process of bugs (covered in 7.2)
(4) Test Environment (covered in 7.2)
(5) Test cases (covered in 8.)

SecurIoTy relevant test cases are classified into these test type:

Test type	Acronym
Manual	MN
Automated	AU
To be done	tbd

Table 108: Requirements vs test mapping

We refer to the requirements given in ***D2.3_INTER-IoT_Requirements-and-business_v1.2***

Acceptance criteria

During this project we will deliver prototypical developer (dev) implementations. We define Alpha-, Beta-and release-Versions in the following paragraphs. The quality standards are defined as given in the following tables based on a maximum of allowable errors and severity levels.

Developer Version (Dev)

Developer versions are snap shots of the current developer process. A number of tests should have been passed and documented in test cases. However, there are no formal release criteria defined.

Alpha Version

An alpha version delivers the number of features required for this state. It is feature complete. In the following table the acceptance criteria are defined for the release.

Criticality	Maximum number of Test Cases
Critical	5
Important	10
Low	30
Trivial	60

Table 109: Alpha version quality criteria

Beta Version

A beta version delivers the number of features required for this state. It is feature complete. In the following table the acceptance criteria are defined for the release. Features will not be added anymore. Quality is in the focus now.

Criticality	Maximum number of Test Cases
Critical	0
Important	5
Low	20
Trivial	40

Table 110: Beta version quality criteria

Release Version

A release version delivers is fully featured and complies with the quality standards defined. Quality is defined in the following table.

Criticality	Maximum number of Test Cases
Critical	0
Important	0
Low	15
Trivial	30

Table 111: Release version quality criteria

The software is delivered if all acceptance criteria are met during the tests. Per iteration only those features will be test which are relevant for the current iteration and delivery plan.

3.3.11.6 Criticality and Priority

Bugs found during testing will be reported in the internal sprint-log (bug tracking). Bugs are classified into criticality and priority. The classification system is given in the following table.

Criticality	Description
Critical	The application, major parts of the application or major features are not available or will crash the system The testcases have not been met.
Important	Important parts of the application are not available or have not passed the tests. There is a workaround to provide the same/similar functionality.

Low	Some functions and features do not work according to the specification. There is a workaround. There is no major disadvantage in using existing workarounds.
Trivial	The application runs smoothly. Changes are made for improved efficiency or for cosmetic reasons.
Feature request	This not a bug rather a new request. Here requirements are reformulated, changed or added.

Table 112: Criticality description

Feature requests have been added to include a way to specify useful extensions. A decision has to be made if this is handled as a change request.

In addition to criticality a bug can be given a priority. Bugs with higher priority will be handled earlier.

Priority	Description
Urgent	Bug handling must be initiated immediately. A patch should be provided asap.
High	Bug handling must be pursued with high priority. The bug can be addressed during the next release.
Medium	Bug can be adressed in one oft he next releases.
Low	Bug can be adressed as soon as there are ressources available.

Table 113: Priority description

Nonfunctional test

Please note that nonfunctional tests are included for completeness. However, they do not manifest test cases but rather are descriptions of how to use the system and what to look for, e.g. security, usability, compatibility.

TT_01 Test tool Burp

For external security testing we use: *Burp Suite Scanner* | *PortSwigger*

<u>Test description</u>

Test descriptions are given in the following sections. Test results are based on beta release requirements.

Architecture

ID	Name	Description	Reference	Status
TC10xxxx	**Architecture**			
TC100010	Scalability. Computing resources	Use load generator to retrieve data (1) Test with 50 clients (response time < 1000 msec) (2) Test with 250 clients (response time < 1000 msec) (3) Test with 500 clients (response time < 2000 msec)	REQ3	**AU/PASS**
TC100020		Add storage to system / hot spare while system is operational.		**AU/PASS**
TC100030		Remove storage from system / hot spare while system is operational.		**AU/PASS**

API

ID	Name	Description	Reference	Status
TC20xxxx	**API**			
TC200010	Gateway access API	A list of exposed functions can be found in the technical documentations. All exposed functions can be accessed and are covered by the security gateway. Non-allowed, non-exposed and non-existing function calls are blocked	REQ243	**MN / PASS**
TC200020		Send unlisted commands to systems, gateway must not send data to controller and will show an error message		**MN / PASS**
TC200030		Send listed commands to systems, gateway must send data to controller and will show content		**MN / PASS**

TC200040		Send listed commands to system with non-plausible added parameters, gateway must not send data to controller and will show an error message		**MN / PASS**
TC200050	API for third-party developers	A list of exposed functions can be found in the technical documentations. HTML frontend is available.	REQ47	**MN / PASS**
TC200060		Have a third party integrate the API		**Tbd**
TC200070	AutoLogin	A list of exposed functions can be found in the technical documentations. Autologin is one of the exposed functions. Alternative implementations are available. User can assign rights and expiration dates.	REQ60	**MN / PASS**
TC200080		Have a third party integrate the API		**Tbd**
TC200090	API allows create/update/remove users	A list of exposed functions can be found in the technical documentations. Create/update/remove is part of the exposed functions.	REQ264	**MN / PASS**
TC200100		Have a third party integrate the API		**Tbd**

Interoperability

ID	Name	Description	Reference	Status
TC30xxxx	**Interoperability**			
TC300000	Secure synchronization	Synchronization is handled by publicly available time servers, here: ptbtime1.ptb.de	REQ56	**MN / PASS**
TC300010		Time server on OS level must be set, check in best practice		

TC300020	Supports multiplatform	Needs clarification. SecurIoTy runs natively on Windows Server 2012 R2. Virtualized it runs on any Host including Linux. Clients can be operated from any OS.	REQ94	**tbd**

Privacy/Security

ID	Name	Description	Reference	Status
TC40xxxx	**Privacy / Security (CIA – confidentiality, integrity, availability)**			
TC401000	Sensitive data is stored according to national and EU policies	Third parties can not access private data or unauthorized data within the SecurIoTy system. Data protection meets the national and European policies. Data security has been tested against German PersDat, §203 StGB, AO. Zero- knowledge data storage: admins and other personal with access to the physical storage have no access to clear text information. Separation of content and operations: admins and other personal with access to the physical storage have no access to clear text information. Fragmentation of content: content is fragmented and distributed across multiple storages. No one storage has all knowledge to recompile a document. Cryptography: fragments are encrypted by AES 256 Resilience: RAID 5-3 concepts protects against failure of a storage and against manipulation of content. Default is that 1 of 3 storages may fail.	REQ30 REQ27	**MN / PASS**

		House many clients on one system. Each client has got its own key material.		
TC401010		Read backend data from storage and verify it is encrypted		**MN / PASS**
TC401020		Send encrypted fragments though a decryption tool and test if it be broken		**Tbd**
TC401030		Remove storage while system is active, system must keep running		**MN / PASS**
TC401040		Add storage while system is active, system must add fragments and re-initiate the original state		**MN / PASS**
TC401050		Remove keys from keystore, system must not deliver any files anymore		**MN / PASS**
TC401060		Alter content of keyfiles in keystore, system must not deliver any files anymore		**MN / PASS**
TC401070		reinstall keyfiles in keystore, system must deliver files		**MN / PASS**
TC402000	Privacy	See TC401000	REQ37 REQ28	
TC403000	Robustness, resilience and availability	See TC401000 Failure of storage will be compensated by RAID principle; redundant storages will cover and provide fall back. No one (1) storage knows all the fragments to recover a document, if a storage is compromised, the attacker will not gain access to the content even if the attacker is capable of breaking the encryption.	REQ95	**AU / PASS**

		The architecture component “security gateway” will shield the controller from non-conform traffic. security gateway will filter requests and will only let exposed function-calls pass. Optionally to increase availability and performance controllers may be duplicated and spread across a controller farm. The secure gateway will act as a load distributor. If a controller fails or is attacked, spare controller can take over.		
TC403010		Attack the system with an automated attack tool which is fed with current vulnerabilities, result must not show any vulnerability		**AU / PASS**
TC404000	Confidentiality	TC401000 TC403000	REQ69	
TC405000	Avoid data falsification or disclosure	Detailed rights on a user level grant access to specific data only. Data manipulation at the backend will be detected by the DocRAID parity checks. Multitenant capable.	REQ36	
TC405010		Run multiple tenants on same system and verify that data from one tenant cannot be seen by another tenant from the frontend.		**MN / PASS**
TC405020		Run multiple tenants on same system and verify that data from one tenant is different than from another tenant. Use same date entry from front end and verify that backend fragments show different content.		**MN / PASS**

TC406000	Provision of authentication credentials	authentication credentials consisting of a User ID and an authentication device, e.g. Password must be given to gain access. Additionally, a second access code may be required, a second factor, this can be a SMS or an Email.	REQ63	
TC406010		Try to login with false credentials, system must deny access		**MN / PASS**
TC406020		Try to login with no credentials, system must deny access		**MN / PASS**
TC406030		Try to login with correct credentials, system must grant access		**MN / PASS**
TC406040		Leave session open but unattended for > timeframe set at controller, system must deny access, session has ended, user must logon again.		**MN / PASS**
TC409000	A user knows its permissions	User may retrieve permissions	REQ261	**MN / PASS**
TC409005		Login and retrieve user permissions at the user terminal		**MN / PASS**
TC409010	Access to personal data needs to be previously authorized	User may retrieve data by permissions only. Permission can be set to invalidate automatically after a certain time.	REQ263	
TC409020		Admin grants / denies management right, user can/cannot manage the system		**MN / PASS**
TC409030		Admin grants / denies file access rights, user can/cannot read/write files		**MN / PASS**
TC409040		Admin grants / denies file share rights, user can/cannot share files		**MN / PASS**
TC409050		Admin grants / denies history (version) access rights, user can/cannot read/write files		**MN / PASS**

TC409060		Admin grants / denies report access rights, user can/cannot access reports		**MN / PASS**
TC409070		Admin grants / denies WebDAV access rights, user can/cannot WebDAV		**MN / PASS**
TC409080		Share a file, recipient must be able to retrieve file without credentials		**MN / PASS**
TC409090		Share a directory, recipient must be able to retrieve directory without credentials		**MN / PASS**
TC409100		Share a file with an expired date, recipient must not be able to retrieve file, error message will be shown		**MN / PASS**
TC409110		Share a folder with an expired date, recipient must not be able to retrieve folder, error message will be shown		**MN / PASS**
TC410010	**Encryption**			
	Filename encryption	Do not encrypt filenames: The known-plaintext attack (KPA) is an attack model for cryptanalysis where the attacker has samples of both the plaintext (called a crib), and its encrypted version (ciphertext). These can be used to reveal further secret information such as secret keys and code books.		

Compliance (Usability)

ID	Name	Description	Reference	Status
TC50xxxx	**Compliance**			
TC500010	Auditability and Accountability	All user actions are logged into a logfile and can be retrieved with the appropriate rights. Log files can be set to autodelete after a certain period of time, e.g. to be compliant with national law.	REQ58	
TC500020		TC409060		
TC500030		Set autodelete to 1 (one) day, reports with age > 1 day must be deleted by the system		**MN / PASS**
TC500040		Set autodelete to 1 (one) week, reports with age > 1 week must be deleted by the system		**MN / PASS**
TC500050	Logging	See TC500010	REQ68	**MN / PASS**

3.3.11.7 Test outcome overview

Test outcome

The following table will provide an overview of the test result of all the performed tests in this FAT. Please note that this section provides redundant information given already in the previous Section 8.

Architecture

ID	Name	Status
TC10xxxx		
TC100010	Scalability. Computing resources	**AU/PASS**
TC100020		**AU/PASS**
TC100030		**AU/PASS**

API

ID	Name	Status
TC20xxxx		
TC200010	Gateway access API	**MN / PASS**
TC200020		**MN / PASS**
TC200030		**MN / PASS**
TC200040		**MN / PASS**
TC200050	API for third-party developers	**MN / PASS**
TC200060		**Tbd**
TC200070	AutoLogin	**MN / PASS**
TC200080		**Tbd**
TC200090	API allows create/update/remove users	**MN / PASS**
TC200100		**Tbd**

Interoperability

ID	Name	Status
TC30xxxx		
TC300000	Secure synchronization	**MN / PASS**
TC300010		
TC300020	Supports multiplatform	**tbd**

Privacy/Security

ID	Name	Status
TC40xxxx		
TC401000	Sensitive data is stored according to national and EU policies	**MN / PASS**
TC401010		**MN / PASS**
TC401020		**Tbd**
TC401030		**MN / PASS**
TC401040		**MN / PASS**
TC401050		**MN / PASS**
TC401060		**MN / PASS**
TC401070		**MN / PASS**
TC402000	Privacy	
TC403000	Robustness, resilience and availability	**AU / PASS**
TC403010		**AU / PASS**
TC404000	Confidentiality	
TC405000	Avoid data falsification or disclosure	
TC405010		**MN / PASS**
TC405020		**MN / PASS**
TC406000	Provision of authentication credentials	

TC406010		**MN / PASS**
TC406020		**MN / PASS**
TC406030		**MN / PASS**
TC406040		**MN / PASS**
TC409000	A user knows its permissions	**MN / PASS**
TC409005		**MN / PASS**
TC409010	Access to personal data needs to be previously authorized	
TC409020		**MN / PASS**
TC409030		**MN / PASS**
TC409040		**MN / PASS**
TC409050		**MN / PASS**
TC409060		**MN / PASS**
TC409070		**MN / PASS**
TC409080		**MN / PASS**
TC409090		**MN / PASS**
TC409100		**MN / PASS**
TC409110		**MN / PASS**

Compliance (Usability)

ID	Name	Status
TC50xxxx		
TC500010	Auditability and Accountability	
TC500020		
TC500030		**MN / PASS**
TC500040		**MN / PASS**
TC500050	Logging	**MN / PASS**

Table 114: Test outcome overview

3.3.11.8 Integration ethicas and security

Introduction

During the Ethical Review it was requested that ethics had to be included within the project scope. This section addresses recommendations and proposed solutions to the ethical and security points of the integration.

For each pilot the ethics is discussed in paragraphs 8.2 until 8.5. The security aspects of each layer is discussed in paragraph 8.7 and 8.8.

The information for the pilots for both ethics and security comes from the partners and may be included in other documents as well.

SecurIoTy

A major purpose of SecurIoTy is to mitigate ethical issues and security risks. For example, a typical requirement in such a scenario would be to separate the operations from the content. In a typical data storage and/or transport scenario this requirement can hardly be upheld. This problem is known as the “privileged account problem”, i.e. a privileged person, e.g. an administrator would typically have access to sensitive data stored and transported by a system. SecurIoTy solves this problem and would thus be a solution for potential ethical issues arising in the context of IoT.

3.3.12 Third Party: E3City

This project integrates E3Tcity devices with the Device Layer of INTER-IoT Inter Layer Platform. E3city has his own platform, this platform is in production stage that is being used in more than 20 towns in Spain. This development will provide INTER-IoT with a whole device/cloud/app vertical solution to be applied to the Smart Port pilot and any project in general.

These are main objectives:

- Connect E3Tcity devices to the Device layer of INTER-IoT, so that cross interaction can be used in further stages.
- Use E3Tcity devices to provide Smart Lighting features to Valencia Port, such as lighting control, power consumption, climatic sensors, movement detection and lightness level.

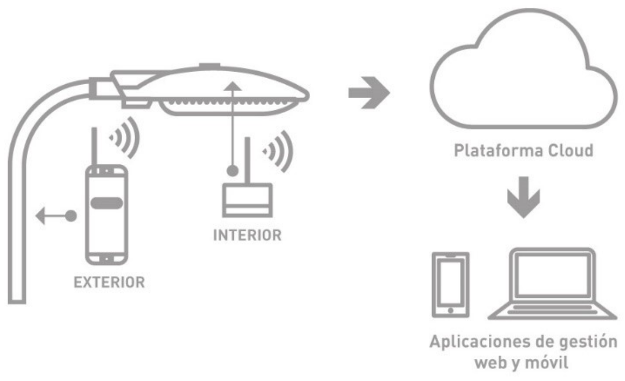


Figure 129: System e3tcity description.

The system consists of three elements:

- **E3Tcity drivers**
  All smart city controllers have a MAC address and they communicate with the cloud via Wi-Fi or cellular networks. They are also endowed with intelligence that allows to monitor continuously energy parameters. With this info they can also control their power consumption and detect faults and cable theft alarms.
- **Cloud platform**
  Drivers exchange information in real time with the cloud platform to provide information on their sensors measures and to be able to interact with the system manager. You can set the volume of data exchange Generated data are sent to the cloud and store for a time in the device memory, ensuring their integrity even in case of failure or sabotage.
- **Users applications**
  User applications are available for PC and Smartphone. They allow users to know the state of offered services in real time and from anywhere.

This system is in production phase and ready to implant.

3.3.12.1 Integration of IoT framework

The following table provides a description of the components of the IoT framework will be integrated in this pilot and witch interfaces are used.

Components	Interfaces	Test
LC_10 (This device will control the sensing part)	Protocol Modbus + INTER-Middleware Platform e3tcity + INTER-Middleware	• If possible, activate the AC OUT 110-230 V from the application? • The device executes the orders of regulation 0-10 V from the application? • The device connects to the router configured in the application? • The device connects correctly with e3tcity cloud?
LS_10 (This device will control the lights on and off)	Protocol Modbus + INTER-Middleware Platform e3tcity + INTER-Middleware	• If possible, activate the AC OUT 110-230 V from the application? • The device executes the orders of regulation 0-10 V from the application? • The device connects to the router configured in the application? • The device connects correctly with e3tcity cloud? • The six relay control outputs work by activating them from the application? • Are the TOTAL POWER ACTIVE (W) measurements monitored correctly from the application? • Are the TOTAL POWER REACTIVE (VAr) measurements monitored correctly from the application? • Are the Voltage (V) and Intensity(A) measurements monitored correctly from the application e3tapp.com?

Table 115: Description of the components.

3.3.12.2 Deliverables and version overview

The following table contains a deliverable list which need to be signed of before FAT testing commences.

ID	Description	Check
Documents		
1	Validation and Test reports of the IoT system components	
2	Validation and Test reports of the Pilot system components	
Hardware		
4	E3t city Devices	
Tools		
7	Electronic testing tools.	
8	E3t City Platform	

Table 116: Deliverable checklist

The following table shows the software components and version of which the system release version 1.0 consists of.

ID	Description	Version	Check

IoT Physical Gateway			
1	E3t city devices	V3.14	
IoT Virtual Gateway			
4	E3t city platform	V2.3.4	
Universaal container			
7	UniversAAL REST API	V3.2.1	

Table 117: Component version overview

3.3.12.3 Requirements, scenarios and use cases

The following table provides an overview of the relation between the requirements and the test(s) that validate their implementation.

ID	Description	Covered by
Functionality		
11	Addressability and reachability	T_03
20	Real time support	T_03
21	Real time output	T_03
25	Remote programming of devices	T_02
26	Remote device control	T_02
API		
243	Gateway access API	T_02
Interoperability		
226	API for network services	T_02
Operational		
96	Enable (automated or semi-automated) linking of relevant data sources	T_03
204	Support smart network resources allocation in heterogeneous wireless sensor networks	T_03
Security		
98	Data provenance	T_03
27	System security	T_01
95	Robustness, resilience and availability	T_01

Table 118: Requirements vs test mapping

3.3.12.4 Test environment

Introduction

In this report we describe the test environment and tools with which we check the correct functioning of our devices, here we detail the 3-main test carried out by e3tcity.

Test environment

To understand our test, it is necessary to understand how solution works.

We provide control and management through our devices. These devices are controlled by users through our platform implemented in the cloud, and they can control them and receive information through web or mobile apps.

Also, we have an API/REST to offer our clients, where they can implement their own platform or connect our devices to yours.

In the next figures we can see a diagram of how the e3tsolution works:

Our testing phase it consists of the three parts:

- Test power and electrical connectivity.
- Test connectivity to our platform.
- Test the correct reception of data and control our devices through the platform.

The tests are done in our laboratory and it is necessary that 2 technicians verify the device to validate it.

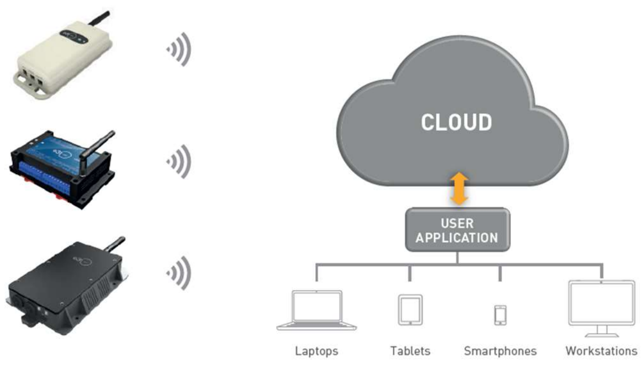


Figure 130: Diagram of solution e3t.

3.3.12.5 Test setups, tools, hooks and probes

TS_01 Test setup electrical

We must connect an antenna and a battery to the device to start the test.

TT_01 Test tool electrical

In relation with the tools necessary for the hardware test, we use the usual electronic testing tool, like Multimeter or oscilloscope.

TH_01 Test hook electrical

This test is to check the correct manufacture of the equipment. To do it, we connect an auxiliary battery on the device. We use the tools to check several points in the device.

For this test, we follow the next checklist.

1	Is the equipment powered correctly by the AC IN 110-230V?	☐ Yes ☐ No ☐ N/A
2	The 6 analog/digital inputs for the sensors works?	☐ Yes ☐ No ☐ N/A
3	If possible, connect and auxiliary battery in the auxiliary BAT input?	☐ Yes ☐ No ☐ N/A
4	The equipment has Modbus RS485 port for connect three-phase network analyzers and work?	☐ Yes ☐ No ☐ N/A

5	The inputs 3,3V and 12 V Dc works?	☐ Yes ☐ No ☐ N/A
6	The equipment has input for the clamp Amp meter and works?	☐ Yes ☐ No ☐ N/A

TH_01 Test probe electrical

The device is designed internally to self-test, once it has been fed and later when it has been connected to our platform, we will receive the data of this test.

TS_02 Test setup connectivity.

Here we test the connectivity between the devices with the cloud, we need an internet connection and the only thing that must be sure that the equipment is connected to the platform, for it the blue led must be fixed.

TT_02 Test tool connectivity.

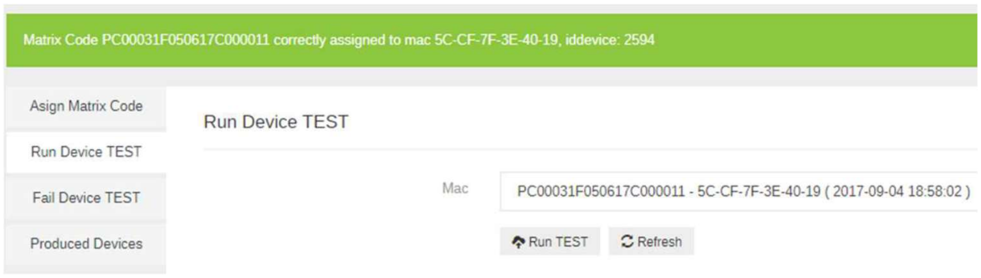
The only tool is our platform.

TH_02 Test hook connectivity.

In the second test we probe the connection with de cloud. Physically the device has two led with 2 different colors, white and blue. The led blue indicates the connection with the cloud, first the led blink and when the connection is established the light of led is fixed.

For continue the test we follow the next steps:

1. We go to the run device test tab and we have to push de button ***Run TEST***

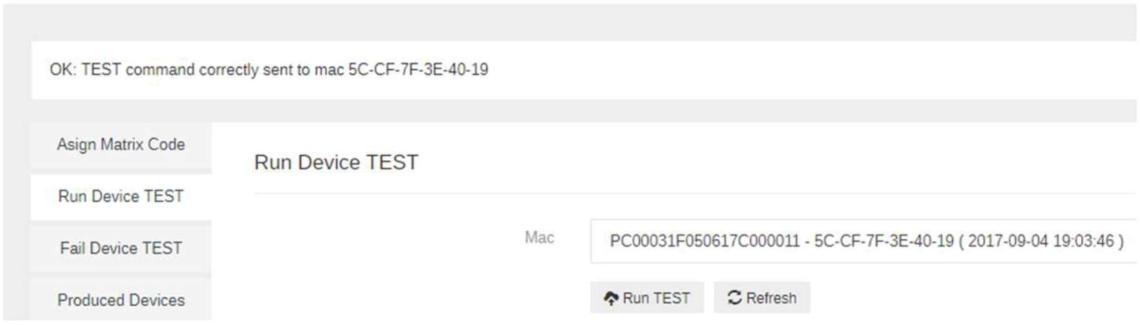


Matrix Code PC00031F050617C000011 correctly assigned to mac 5C-CF-7F-3E-40-19, iddevice: 2594

Run Device TEST

Mac: PC00031F050617C000011 - 5C-CF-7F-3E-40-19 (2017-09-04 18:58:02)

2. We verify that appear the message OK: TEST. This confirms that the message has been sent correctly.



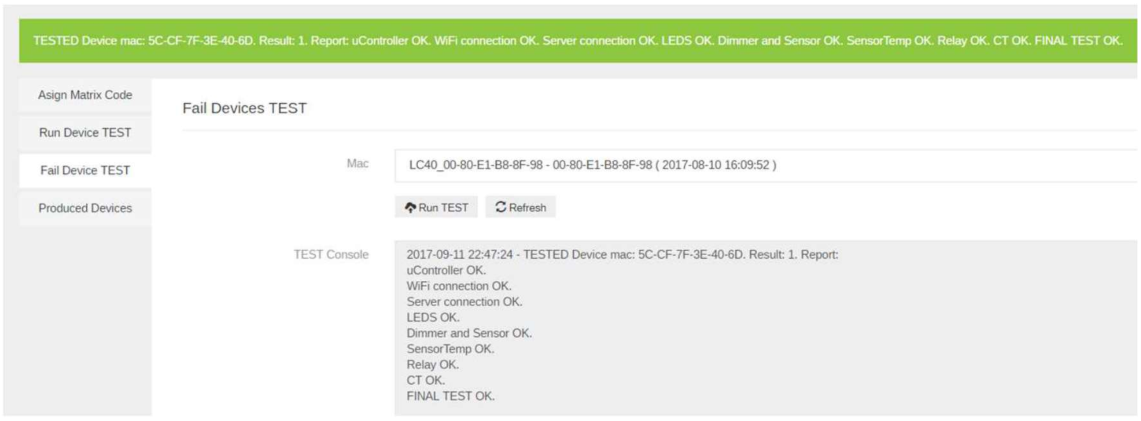
OK: TEST command correctly sent to mac 5C-CF-7F-3E-40-19

Run Device TEST

Mac: PC00031F050617C000011 - 5C-CF-7F-3E-40-19 (2017-09-04 19:03:46)

Six relay control outputs work by activating them from the app.

3. When the test finish correctly we go to the next tab ***Fail Device TEST***


TESTED Device mac: 5C-CF-7F-3E-40-6D. Result: 1. Report: uController OK. WiFi connection OK. Server connection OK. LEDS OK. Dimmer and Sensor OK. SensorTemp OK. Relay OK. CT OK. FINAL TEST OK.

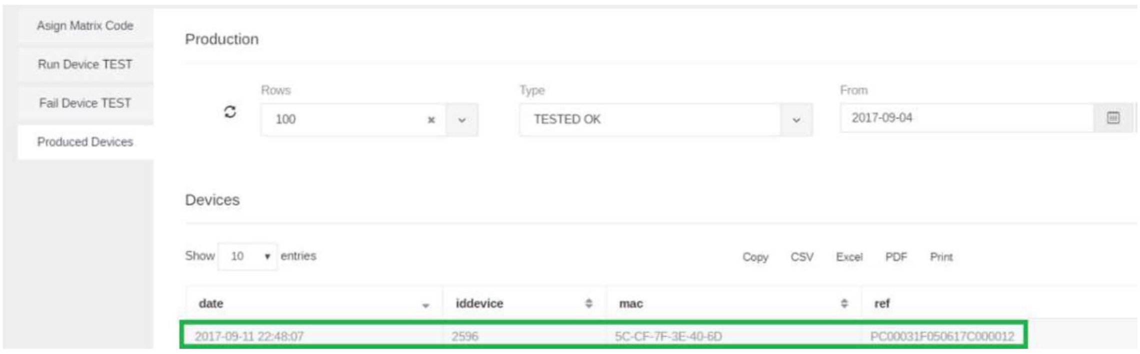
Fail Devices TEST

Mac: LC40_00-80-E1-B8-8F-98 - 00-80-E1-B8-8F-98 (2017-08-10 16:09:52)

TEST Console:

```
2017-09-11 22:47:24 - TESTED Device mac: 5C-CF-7F-3E-40-6D. Result: 1. Report:
uController OK.
WiFi connection OK.
Server connection OK.
LEDS OK.
Dimmer and Sensor OK.
SensorTemp OK.
Relay OK.
CT OK.
FINAL TEST OK.
```

4. When this text finish, the number MAC of the device disappears. The reason is that the device has been register in ***Produced Devices***.


Production

Rows: 100 | Type: TESTED OK | From: 2017-09-04

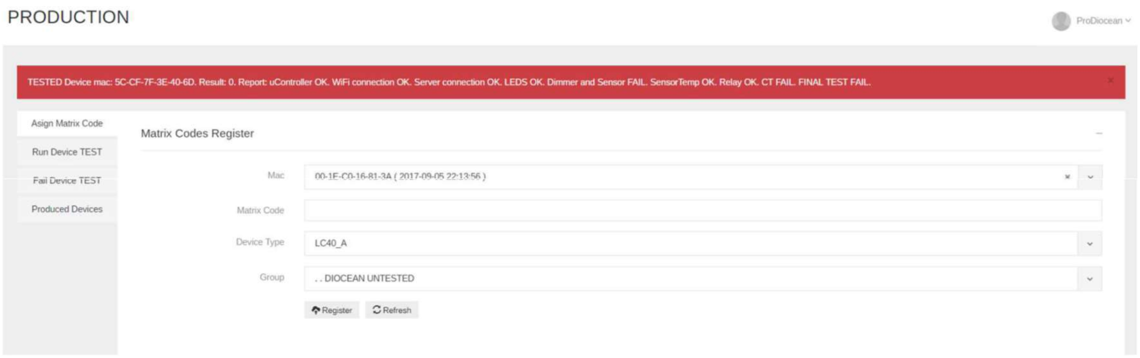
Devices

Show 10 entries

date	iddevice	mac	ref
2017-09-11 22:48:07	2596	5C-CF-7F-3E-40-6D	PC00031F050617C000012

5. The test is finish.

If the test fails, we can see in the console test the description of the fail


PRODUCTION

TESTED Device mac: 5C-CF-7F-3E-40-6D. Result: 0. Report: uController OK. WiFi connection OK. Server connection OK. LEDS OK. Dimmer and Sensor FAIL. SensorTemp OK. Relay OK. CT FAIL. FINAL TEST FAIL.

Matrix Codes Register

Mac: 00-1E-C0-16-81-3A (2017-09-05 22:13:56)

Matrix Code:

Device Type: LC40_A

Group: . . DIOCEAN UNTESTED

TH_02 Test probe connectivity.

The platform shows us a report with the results to review them and establish if the test was correct.

TS_03 Test setup correct interpretation of commands

For this test the only thing that must be sure that the equipment is connected to the platform, for I the blue led must be fixed.

TT_03 Test tool correct interpretation of commands

The only tool is our platform.

TH_03 Test hook correct interpretation of commands

In the last test we probe the response of the device, we sent several commands to the device through the platform, and we check that the device sends measurements correctly.

We follow the next checklist in the verification:

1	If possible, activate the AC OUT 110-230 V from the application (e3tapp.com)?	☐ Yes ☐ No ☐ N/A
2	The device executes the orders of regulation 0-10 V from the application (e3tapp.com)?	☐ Yes ☐ No ☐ N/A
3	The six inputs analog/digital for the sensors works?	☐ Yes ☐ No ☐ N/A
4	The six relay control outputs work by activating them from the application (e3tapp.com)?	☐ Yes ☐ No ☐ N/A
	COMUNICATION WIFI 802.11 B/G/N 2.4 GHZ	
1	The device connects to the router configured in the application e3tapp.com?	☐ Yes ☐ No ☐ N/A
2	The device connects correctly with e3tcity cloud?	☐ Yes ☐ No ☐ N/A
	SINGLE-PHASE CONSUMPTION MEASURES	
1	Are the TOTAL POWER ACTIVE (W) measurements monitored correctly from the application e3tapp.com?	☐ Yes ☐ No ☐ N/A
2	Are the TOTAL POWER REACTIVE (VAr) measurements monitored correctly from the application e3tapp.com?	☐ Yes [✓ No] ☐ N/A
3	Are the Voltage (V) and Intensity (A) measurements monitored correctly from the application e3tapp.com?	☐ Yes ☐ No ☐ N/A
	THREE-PHASE CONSUMPTION MEASURES	
1	Are the TOTAL POWER ACTIVE (W) in PHASE1 measurements monitored correctly from the application e3tapp.com?	☐ Yes ☐ No ☐ N/A
2	Are the TOTAL POWER ACTIVE (W) in PHASE2 measurements monitored correctly from the application e3tapp.com?	☐ Yes ☐ No ☐ N/A
3	Are the TOTAL POWER ACTIVE (W) PHASE3 measurements monitored correctly from the application e3tapp.com?	☐ Yes ☐ No ☐ N/A
4	Are the TOTAL POWER REACTIVE (VAr) PHASE1 measurements monitored correctly from the application e3tapp.com?	☐ Yes ☐ No ☐ N/A
5	Are the TOTAL POWER REACTIVE (VAr) PHASE2 measurements monitored correctly from the application e3tapp.com?	☐ Yes ☐ No ☐ N/A
6	Are the TOTAL POWER REACTIVE (VAr) PHASE3 measurements monitored correctly from the application e3tapp.com?	☐ Yes ☐ No ☐ N/A
7	Are the POWER FACTOR PHASE1 measurements monitored correctly from the application e3tapp.com?	☐ Yes ☐ No ☐ N/A
8	Are the POWER FACTOR PHASE2 measurements monitored correctly from the application e3tapp.com?	☐ Yes ☐ No ☐ N/A

9	Are the POWER FACTOR PHASE measurements monitored correctly from the application e3tapp.com?	☐ Yes ☐ No ☐ N/A
10	Are the TOTAL POWER ACTIVE (W) TOTAL measurements monitored correctly from the application e3tapp.com?	☐ Yes ☐ No ☐ N/A
11	Are the TOTAL POWER REACTIVE (VAr) in TOTAL measurements monitored correctly from the application e3tapp.com?	☐ Yes ☐ No ☐ N/A
12	Are the POWER FACTOR TOTAL measurements monitored correctly from the application e3tapp.com?	☐ Yes ☐ No ☐ N/A
13	Are the Voltage (V) and Intensity (A) PHASES measurements monitored correctly from the application e3tapp.com?	☐ Yes ☐ No ☐ N/A

TH_03 Test probe correct interpretation of commands

The platform shows us in real time if our commands are being interpreted correctly in real time.

3.3.12.6 Test description

3.3.12.6.1 Scenario: Testing power-on, power-off and reception of measurements of a luminaire.

Use case Testing device.

In the following, we detail a scenario where we will check the operation of a system that must control the lighting of luminaire and receive measures from it.

Test probe electrical

T1.1.1 Test electrical

ID	**T1.1.1**
Test	This test is to check the correct manufacture of the equipment
Type	Physical
Setup	TS_01
Start	Connected to a battery
Req.	Battery and tools
Input	
Output	
Logs	Our database
Outcome	Fail/Pass

T1.1.2 Test connectivity

ID	**T1.1.2**

Test	Probe the connection with de cloud
Type	Cloud
Setup	TS_02
Start	Connected to the platform
Req.	
Input	Commands from the platform
Output	Error report
Logs	Our database
Outcome	Fail/Pass

T1.1.3 Test correct interpretation of commands

ID	T1.1.3
Test	Probe the response of the device
Type	Cloud
Setup	TS_03
Start	Connected to the platform
Req.	
Input	Commands from the platform
Output	Correct response to commands
Logs	Our database
Outcome	Fail/Pass

3.3.12.7 Test outcome overview

Test outcome

The following table will provide an overview of the test result of all the performed tests in this FAT.

Test	Description	Outcome
T.01	Test electrical	Fail/Pass
T.02	Test connectivity.	Fail/Pass

T.03	Test correct interpretation of commands	Fail/Pass
FAT Outcome		**Fail/Pass**

Table 119: Test outcome overview

3.3.12.8 Integration ethics and security

Introduction

During the Ethical Review it was requested that ethics had to be included within the project scope. This section addresses recommendations and proposed solutions to the ethical and security points of the integration.

E3T-PDI

In terms of security, we are able to detect if our equipment has been manipulated, since from our platform we can read internal parameters of the devices and detect errors.

Furthermore, all our communication between devices and the platform is secure, since the connection is encrypted, also all our control commands have a token generated by us, to avoid external manipulations.

In the ethical part, this project by its design is completely ethical, with the demonstration that the connection between platforms is possible, many possibilities open up to be able to interconnect many parts of a city.

With our system, to be able to control the consumption, luminosity and a better management of on and off we can have a lower energy consumption and therefore contaminate less.

4 Open Call Third Parties Evaluation

4.1 Introduction

The main objective of the open call has been testing the INTER-IoT proposed components and methodology through new scenarios, platforms and components to achieve interoperability between IoT platforms. The proposals are providing support to validate INTER-IoT components in scenarios deployed in different application domains. Allowing the evolution of the INTER-IoT products or parts of them (i.e. INTER-LAYER and INTER-FW) as a whole to match the needs of proposers, but at the same time evolve their products in order to add new interoperability features.

The participation of the third parties within INTER-IoT project has been designed as a collaboration between the individual entities and the consortium. However, as this process has been considered and treated as the seed of the INTER-IoT ecosystem, collaboration between different third parties has been encouraged.

Third parties currently participating in INTER-IoT are two large contributions and ten small contributions:

- Large contributions:
  - Integrating sensiNact platform with INTER-IoT Framework, CEA - *Commissariat à l'énergie atomique et aux énergies alternative*, (France).
  - INTER-OM2M, Vrije Universiteit Brussel, (Belgium).
- Small contributions:
  - INTER-HARE platform: Integration of multiband IoT technologies, Universitat Pompeu Fabra (Spain).
  - Mission Critical operations based on IoT analytics (MiCrOBIoTA), Nemergent Solutions S.R.L., (Spain).
  - Interoperable Situation-Aware IoT-Based Early Warning System, University of Twente, (The Netherlands).
  - SENSHOOK, Irideon SL, (Spain).
  - SOFOS: A software-defined end-to-end IoT gateway with virtualization capabilities, INFOLYSIS P.C., (Greece).
  - E3Tcity Smart City Platform and Devices Integration, E3TCity, (Spain).
  - ACHILLES: Access Control and autHenticatIon deLegation for interoperabLE IoT applicationS, Athens University of Economics and Business – Research Center (AUEB), (Greece).
  - INTER-HINC: Interoperability through Harmonizing IoT, Network Functions and Clouds, TU Wien - Vienna University of Technology, (Austria).
  - A Semantic Middleware for the information synchronization of the IoT devices, Institute of Industrial Technologies and Automation - National Research Council (ITIA-CNR), (Italy)
  - SecurIoTy - security for the IoT, AvailabilityPlus GmbH, (Germany).

The relationship between the different small contributions and the INTER-IoT architecture is presented in Figure 131. Different contributions are related with a specific aspect of the INTER-IoT architecture. However, three contributions (i.e. ITIA-CNR, E3TCITY and U. Twente) are addressing more than one element of the architecture:

- ITIA-CNR is dealing with the middleware and semantics as the semantic interoperability solution proposed uses semantics but also a cloud based solution that can be incorporated as a bridge to the MW2MW component.
- E3TCITY with device and also middleware, as their smart light devices can be connected using MQTT with the INTER-IoT gateway and with the middleware using their cloud solution.
- U- Twente contribution initially dealt with the integration of and Early Warning System application with INTER-IoT, however during the execution of the collaboration semantics and the SAREF based ontology turned to be a very relevant contribution to IPSM and GoIoTP, working also in this aspect of the architecture.

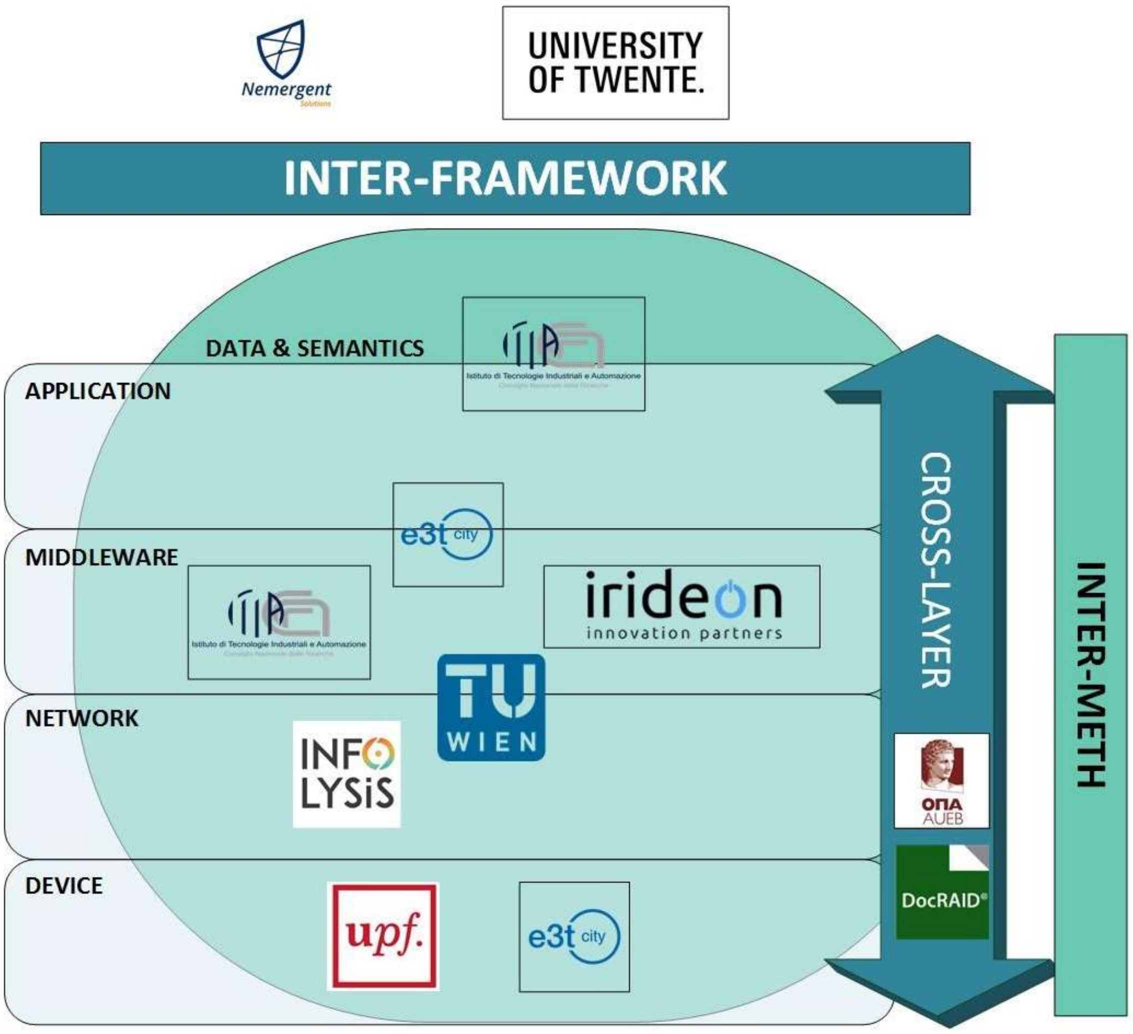


Figure 131: INTER-IoT areas covered by the Small Contribution third parties.

First evaluation of the small collaboration of the open call was held on 29th -30th May. During the presentation the third parties presented the work done so far, presented a potential business model individual and joint with INTER-IoT and project progress report. The ten small contributions made the presentation and received the pre-financing.

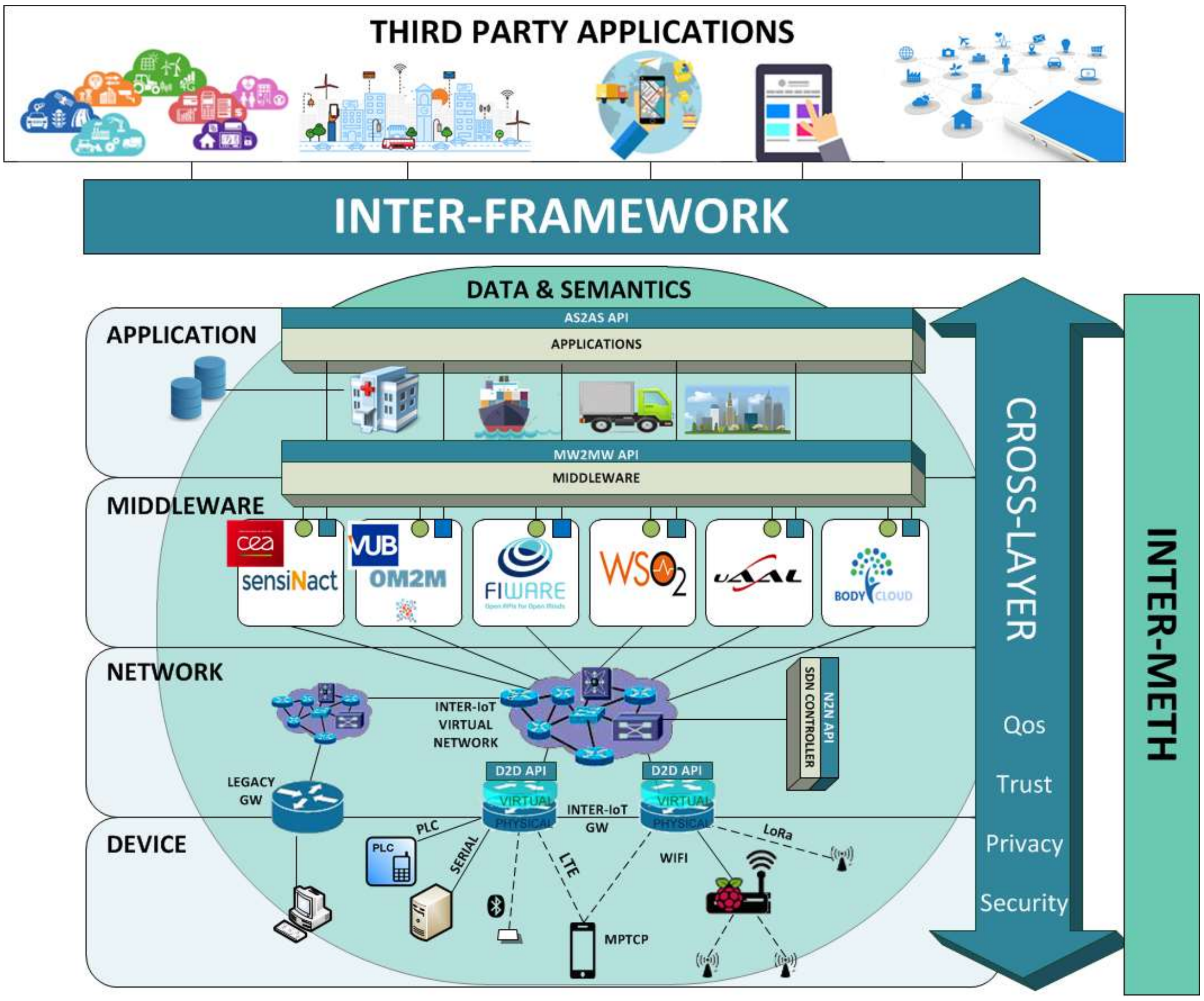


Figure 132: INTER-IoT areas covered by the Large Contribution third parties.

Large contributions made their first review presentation in M19 during the 6th plenary meeting in Eindhoven. VuB and CEA presented their preliminary work and the plans to achieve a successful integration of their devices, platforms and applications. VuB was including OM2M and CEA sensiNact, which in February 2017 become an Eclipse funded project.

4.2 Second Evaluation

Second evaluation of the third parties collaboration was scheduled in 17th/18th January 2018. The process to be followed by the third parties to participate in the review, show their progress and access the second leg of funding was specified in the collaboration agreement:

- Submission of a detailed PPR with the different planned deliverables as annexes of the document.
- Submit annex 5.2 and annex 3 of the collaboration agreement, corresponding to the request of the midterm funding and providing a lump sum justification for the 30% of the budget. (18.000€ for small contributions and 37.500€ for large contributions).
- Attend a workshop, present the current work and future progress and attend the questions of INTER-IoT PCC.
- Obtain consortium approval of the corresponding contribution.

In order to exchange information and find potential synergies the evaluation was carried out with an open workshop structure, in which the different third parties could interact. The evaluation included a poster session to have a different way of exchanging ideas between the third parties and the consortium, following one of the approaches discussed during ecosystem building sessions of IoT-EPI.

The following sections detail the evaluation details and progress of the twelve third parties.

4.3 Large contributions

25 - INTER-OM2M

Vrije Universiteit Brussel (VUB)

Description

This work concerns the contribution of the research team of the ETRO department of Vrije Universiteit Brussel to the logistic application in the port of Valencia. The main goal is to provide easy to use data to workers in the site. In particular, the Stickntrack GPSs, which are IoT devices using the Sigfox network for the communication, are used. These devices are endowed with an accelerometer and a GPS module to provide activity and location information about the specific asset with which they are stuck. The Stickntrack GPSs have been stuck with trucks and equipment in the port and are currently under test.

The communication between the OM2M platform and a non-oneM2M entity, which can be any kind of IoT device, is only possible with the inclusion of a custom Interworking Proxy Entity (IPE) in the OM2M Infrastructure Node (IN). Since the Stickntrack GPSs are programmed to send measurements to the Sensolus cloud periodically (every 20 minutes), the OM2M platform should communicate with the Sensolus cloud to retrieve this information. This is now possible through our custom IPE that provides two services: the periodic retrieval of data managed by a timer and the provision of a friendly Graphical User Interface (GUI) with which the user can interact.

The Stickntrack GPSs are unable to provide location information whenever they lose the GPS signal. However, this problem was solved with the inclusion of Stickntrack geobeacons that can be deployed indoors and fixing their positions for locating the assets. The Stickntrack geobeacons are very simple devices, which do not have any network connectivity, and they just broadcast a Bluetooth Low Energy (BLE) signal over the air. The estimated range of such signal depends on the environment and can be between 20 meters and 70 meters. A Stickntrack GPS can detect a maximum of two signals at the same time but it only considers the strongest one estimated its position with the position of the correspondent Stickntrack geobeacon.

To complement the measurements serving the logistics applications of the port, VUB will include some environment/logistics related measurements, featuring Sigfox/Lora as underlying radio technologies. These measurements will be made available through the oneM2M framework, maintained at VUB. The open source OM2M-based implementation will be deployed as common service layer to allow the use of different application protocols such as MQTT and CoAP in the devices. A gateway (e.g.

Raspberry Pi) will play a central role in forwarding the gathered data to the OM2M server.

On top of that, environmental measurements and measurements of the city of Valencia, available under the FIWARE framework will also be made available under the oneM2M VUB platform to complement the measurements made by VUB itself. Basic security considerations will be taken care of. The applications of the city of Valencia will be able to access and exploit the extra measurements produced by the equipment installed by VUB at the port. To this end, an INTER-IoT – oneM2M bridge will be developed by VUB.

An OM2M bridge is in the process of being implemented to allow the communication between the OM2M platform and the Middleware-to-Middleware (MW2MW) framework. Important preliminary steps have been taken make it possible to retrieve IoT sensor data from a different platform such as Fiware and vice versa.

Another important future work will be the inclusion of an authentication mechanism to provide some basic security feature and avoid illegal access to the IoT sensor data stored in the OM2M platform. Some first steps have been taken.

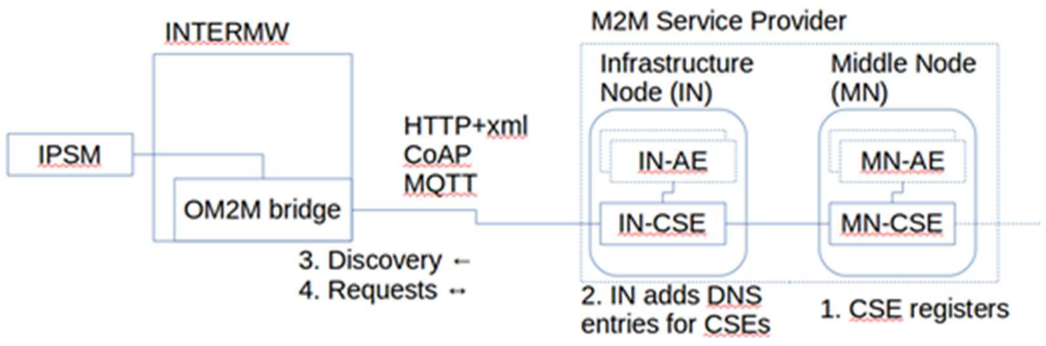


Figure 133: OM2M bridge integration in INTER-IoT architecture.

The methodology and lessons learned from this contribution and its integration in Inter-IoT will be provided to a group of Belgian SMEs under the format of tutorials, workshops and demonstrations. Thanks to the embedding of this project in several national projects involving many Dutch and Belgian companies and 4 ICT network organizations, some supported by major national telecom operators, the results of this project will have immediate industrial impact.

Progress

The large contribution is progressing as expected and the interaction with the different development teams is very fluent. The scheduled activities have been executed as planned:

- Integration and storing of the measurements of the trackers in the university oneM2M framework.
- Provision of a user-friendly interface to interact and retrieve data from the university oneM2M framework.
- Bridge (including syntactic translator), this is work in progress.
- Demo 1, subscription and notification through the bridge, this is work in progress.

The research team has performed different dissemination actions:

- Standardization activity: ETSI OneM2M week that took place in October 2017 in Sophia Antipolis.
- Article in journal: S.Thielemans, D. Di Zenobio, A. Touhafi, P. Lataire and K. Steenhaut, “DC Grids for Smart LED-Based Lighting”, Energies, vol 10, no 10, pp 1-26, 2017.
- Workshop presentation: Wireless Community IoT Integration, Leuven (Belgium) 6 Feb 2018. Joint presentation between INTER-IoT consortium (Roel Vossen – NEWAYS) and VUB team.
- Cross dissemination: presentation to Flemish companies at PROXIMUS during Horizontal-IoT TETRA project.

Recommendations

The evaluation panel is highly satisfied with the contribution, some recommendations regarding the contribution are:

- Provide better documentation of the different interfaces developed and the preliminary bridge version.
- Test the compatibility of the OM2M bridge with the newly released OpenMTC platform also based in OneM2M.
- Provide a detailed inventory of sensors and their characteristics.

Table 120: INTER-OM2M Evaluation.

71 - Integrating sensiNact platform with INTER-IoT Framework

CEA - Commissariat à l'énergie atomique et aux énergies alternative

Description

The main goal of this collaboration is to perform the integration of sensiNact platform into the “INTER-IoT supported platforms” catalogue and to validate INTER-IoT tools and methodology with an IoT platform already used in real-life deployments, in particular in the smart city domain in Europe and in Japan. sensiNact is a horizontal platform dedicated to IoT and in particularly used in various smart city and smart home applications.

CEA collaboration with the INTER-IoT project is two-fold:

- To take benefit of INTER-IoT interoperability methodology and tools and include them into the sensiNact platform, so that it can be compatible with other “INTER-IoT supported” platforms.
- To provide to Inter-IoT the opportunity to validate their framework with the integration of sensiNact platform and thus access to all compatible data sets from sensiNact from different domains such as smart cities, smart farming, smart ski resort, smart building, smart living and well-ageing, etc. is now under further development and deployment in several other European projects such as BigClouT, FESTIVAL, OrganiCity, Wise-IoT, IoF2020 and ACTIVAGE.

CEA will test the integration with at least 3 other platforms showing that sensiNact is working in an interoperable way with other platforms. This is the most relevant objective since the goal is to be able to interoperate with other platforms to share and reuse data and functions. The integration will be validated via pilot applications, with at least 2 cross-domain applications using INTER-IoT APIs seamlessly gathering data or performing actuations at different underlying platforms. It is an achievable objective since sensiNact has already performed field trials in various European projects and has access to several open IoT devices and platforms. Pilot organization is very relevant in order to effectively validate the integration work, and during the first phase CEA will interact with the relevant stakeholders in INTER-IoT, specially VPF.

This collaboration will specifically engender closer collaborations between the sensiNact and INTER-IoT ecosystems. CEA and its partners are currently building a working group about open smart urban environments within the Eclipse Foundation, which is a well-known and active community of developers. sensiNact will form the core of this ecosystem to which different partners will participate with additional building blocks and tools (security, communication protocols, data analytics, etc.). We will join the forces with the INTER -IoT ecosystem and co-organise at least 2 joint events (workshop, hackathon, etc.) to disseminate. It will be achieved by taking benefit of available IoT related events such as IoT week, IoT-EPI meetups or Eclipse community events on IoT. Ecosystem enlargement is certainly relevant objective for both INTER -IoT and sensiNact.

As shown in the Figure below, sensiNact will be inserted to the INTER-IoT framework at the middleware layer. sensiNact will implement the APIs provide by the INTER-IoT Framework and build the necessary adapters for data model adaptation and transformation. sensiNact's middleware layer services such as resource discovery and lookup, security, data processing, etc., will be adapted to be compliant with the Inter-Framework.

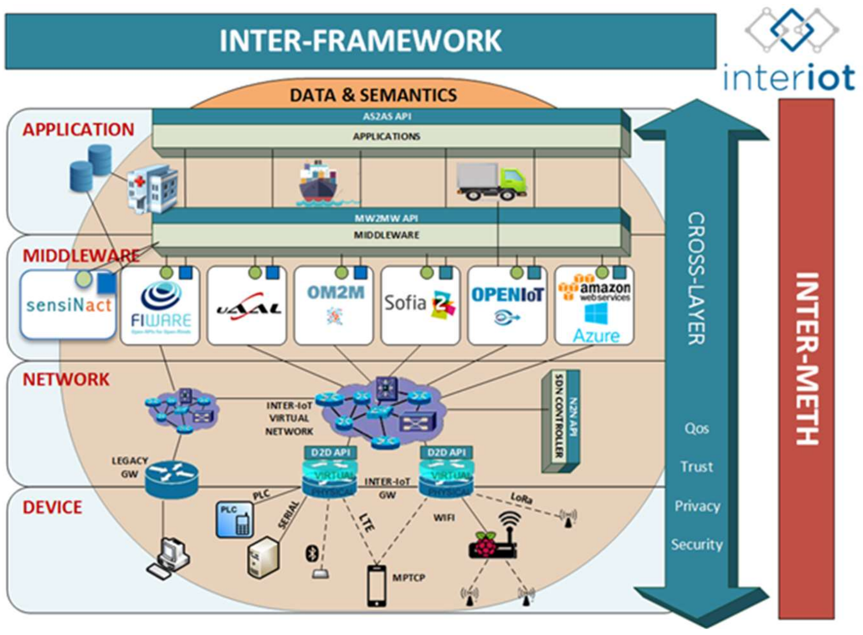


Figure 134. sensiNact bridge integration in INTER-IoT architecture.

Progress

The open call contribution is advancing as expected and this period goal was to gather the state of the art of the actual INTER-IoT framework exploring the deliverables 4.1 (Initial Reference IoT Platform Meta-Architecture and Meta Model) and 3.1 (Methods for Interoperability and Integration). Those reference documents state the base concept and the architecture of the INTER-IoT framework. The content was updated with D3.2, D4.3 and D4.5. The 2nd period work has mainly been conducted within the Work Package 2. Its main activity has been to provide the inputs necessary to evaluate the future integration of InterIoT with sensiNact platform through a Factory Acceptance Test (FAT) plan (Deliverable 6.2).Thus, the main contributions have been:

- Getting familiar with the INTER-IoT framework at inner INTER-IoT MW2MW component level
- Exploring Deliverable 4.1 and understanding the main concept and meta-architecture and data model of INTER-IoT
- Exploring Deliverable 3.1 and understanding the interoperability and Integration methodology of the INTER-IoT
- Access to the project
    - Code source repository
    - artifact repositories
    - documentation repository
- Provide first version of Factory Acceptance Test Plan document
- Provide feedback about the documentation validity
- Initial identification of the data sources to be integrated for interoperability, so as the different end user applications.
- Ongoing development of the bridge.

Recommendations

The evaluation panel is satisfied with the contribution, some recommendations regarding the contribution are:

- Provide better documentation of the different interfaces developed and the preliminary bridge version.
- Identify components for semantic interoperability.
- Provide a detailed inventory of sensors and their characteristics.
- Provide more dissemination activities linked to the collaboration.

Table 121: sensiNact platform Evaluation.

4.4 Small contributions

27 - INTER-HINC: Interoperability through Harmonizing IoT, Network Functions and Clouds
TU Wien - Vienna University of Technology
Description
IoT applications require various resources from Internet of Things (IoT), Network Function Virtualization (NFV), and cloud systems. Such resources will be provided on-demand, based on IoT-as-a-service models. They must be interoperable for the application use. In this project, we address IoT interoperability together with its counterparts of network function and cloud.

The interoperability is addressed within a context of resource slices, which can be provisioned and customized, on the demand, for different applications. Our approach “resource slice interoperability” not only leverages existing layered interoperability solutions but also builds cross-layered interoperability and cross-system interoperability solutions for a system of IoT, network function and cloud resources. Interoperability is addressed dynamically for application specific requirements, solving interoperability from the bottom up: we cannot enforce the same models for complex IoT providers; instead we devise dynamic interoperability solutions using metadata and provider-specific APIs. This enables us to provide slice-aware programming APIs for making IoT interoperability (through controlling IoT and IoT data pipelines).

The collaboration contemplates the development of a framework called INTER-HINC (Interoperability through Harmonizing IoT, Network Functions and Clouds) will be developed and validated with scenarios in INTER-DOMAIN covering both logistics and health care situations.

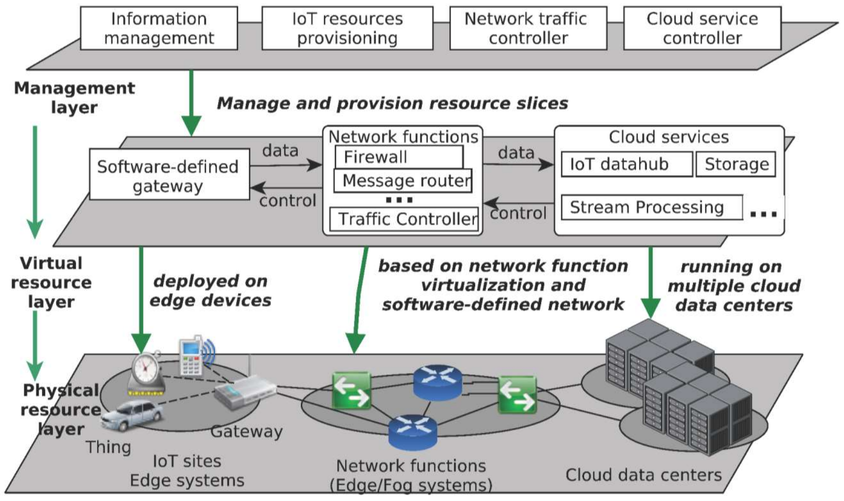


Figure 135: INTER-HINC architecture.

Progress

The contribution is advancing as expected and during this period, main activities have been associated with:

- Definition of the use cases.
- INTER-HINC architecture design and preliminary integration with INTER-IoT. Specifically, a conceptual architecture of Resource Slice Interoperability Hub (rsiHub) and its key elements. rsiHub presents a broad-view on our conceptual Resource Slice Interoperability, covering also important aspects of interoperability, such as recommendation of Interoperability bridges at runtime.
- Definition of information, interaction and integration models; programming APIs, testing services and Factory Acceptance Test.

During this reporting period, TU Wien has also disseminated ideas of INTER-HINC to various places, including:

- Hong-Linh Truong, "Managing and Testing Ensembles of IoT, Network functions, and Clouds", Center for Cyber-Physical Systems and Internet of Things, University of Southern California, 21 Sep, 2017

- Hong-Linh Truong, "Modeling and Testing Uncertainties in Application-oriented Slices of IoT, Network functions, and Clouds", Ericsson R & D, 13 June, 2017, Bangalore, India

Scientific papers relevant to INTER-HINC are on submission:

- Hong-Linh Truong, Duc-Hung Le, Hong-Linh Truong, Duc-Hung Le, Nanjangud Narendra, Interoperable Resource Slices for IoT Applications, Jan 2018. On submission.
- Hong-Linh Truong, Enabling Edge Analytics of IoT Data: The Case of LoRaWAN, Jan 2018. On Submission.

Recommendations

The project is advancing as expected, however some corrective actions have to be performed in the work for the following period:

- The research team has to focus in order to reduce the scope of data gathering, as current proposal creates an execution risk due to its size.
- Identification of a potential simplification if the interoperability is performed with INTER-FW instead at individual layers.
- The research team has to improve documentation sent to the consortium.
- With the incorporation of the additional researcher in the team activity has improved, however the activity needs to be maintained.

Table 122: INTER-HINC Evaluation.

39 - SOFOS: A software-defined end-to-end IoT gateway with virtualization capabilities

INFOLYSIS P.C.

Description

The imminent arrival of the Internet of Things (IoT), which consists of a vast variety of devices with heterogeneous characteristics, means that future networks need a new architecture to accommodate end-to-end IoT networking, dealing with: i) the expected increase in data generation, ii) the problems related to the end-to-end IP networking of the resource-constrained IoT devices, iii) the capacity mismatch between devices, and iv) the rapid interaction between services and infrastructure.

Software defined networking (SDN) and network function virtualization (NFV) are two technologies that promise to cost-effectively provide the scale and versatility necessary for IoT services in order to address efficiently the aforementioned challenges. Moreover, given that SDN and NFV are considered a fundamental component in the 5G landscape, since it is widely recognized that 5G networks will be software-driven and most components of future heterogeneous 5G architectures should be capable to support software-network technologies, both SDN and NFV are promising candidate technologies for a Software Defined Approach of end-to-end IoT Networking.

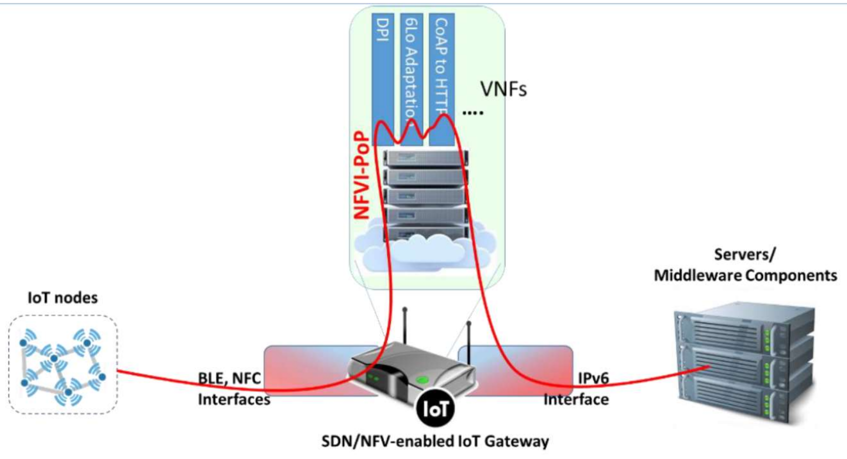


Figure 136: SOFOS proposed SDN/NFV end-to-end IoT Gateway.

SOFOS aims at advancing the existing INTER-IoT framework with SDN and NFV functionalities towards a Software-defined end-to-end IoT infrastructure with IoT service chaining support. The main objective of the proposed SDN/NFV-enabled framework is to enhance the interoperability of the INTER-IoT framework in order to facilitate the interoperable management of a large number of diverse smart objects that currently operate utilizing a variety of different IoT protocols.

The proposed software-defined end-to-end IoT addition with SDN/NFV capabilities on the INTER-IoT GW contributes positively beyond the current SoTA regarding the INTER-IoT objectives to the following aspects:

- By integrating SOFOS to the INTER-IoT infrastructure, the set of tools, components, mediators of INTER-IoT are extended in order to enhance the interoperability of different underlying IoT platforms via appropriate VNFs, such as mapping functions.
- Contributes to the design, implementation and integration of interoperable networking layer components (in the form of VNFs) for INTER-FW. The proposed SDN/NFV extension facilitates the deployment of virtual mapping functions and other networking layer components (such as virtual SDN switches), which are based on different standards higher-level communication standards (e.g. TCP/IP, HTTP, CoAP, etc).
- Implements an NFV-based virtualization mechanism for smart objects and platform of smart objects for INTER-FW, including orchestration mechanism for transferring VNFs (i.e. virtual objects/functions) between cloud platforms (i.e. NFVI PoPs).
- Implements NFV orchestration and cloud support mechanisms, which are integrated in INTER-FW, including support for different services, inter-cloud (inter-NFVI PoP) mechanisms applied to IoT and support for virtualization.

Progress

The collaboration is advancing as expected, more specifically the activities include the definition of business model, its financial aspects analysis, as well as the description of the use case/pilot & the requirements of the architecture in relation with INTER-IOT platform. Towards this direction, INFOLYSiS has analyzed and demonstrated:

- The business analysis and business model of SOFOS solution, including the business canvas component.

- The financial and cost benefit analysis of SOFOS solution, including reasonable assumptions and analysis constraints under different financial conditions and metrics
- The SOFOS platform architecture and integration to INTER-IoT platform: (i) Presentation and analysis of enabling technologies; (ii) SOFOS integration setup and architecture; (iii) Integration environment; (iv) Test setup for integration; (v) SOFOS integration to INTER-IoT platform
- Integration of the SOFOS components in the INTER-IoT GW and link with N2N INTER-IoT component.

Recommendations

The evaluation panel is highly satisfied with the contribution, some recommendations regarding the contribution are:

- Clear definition of the link between OpenDayLight used by SOFOS and RYU used by INTER-IoT, compatibility issue has to be demonstrated.
- Definition of the use case that currently is in a preliminary stage.

Table 123: SOFOS Evaluation.

42 - INTER-HARE platform: Integration of multiband IoT technologies

Universitat Pompeu Fabra

Description

The continuous emergence of new technologies based on the IoT paradigm has resulted in a heterogeneous ecosystem. The proposed INTER-HARE platform will create synergies between LPLANs and LPWANs, by building and testing an IoT platform easily scalable (both in coverage range and devices) and flexible (both in the considered use cases and the frequency bands from employed devices). The proposed work has a starting point in the HARE communication system previously developed in a research project.

Interoperability is provided by a hierarchical two-tier network, where dual-band devices simultaneously interact with end devices and the INTER-IoT gateway. INTER-HARE platform will allow in the mid-long term the deployment of advanced services based on sensor networks, from a wide range of everyday life applications, at reasonable costs, and low time-to-market.

The proposed approach splits the INTER-HARE platform into two networks with different purposes:

- Transport network: It involves all internal infrastructure responsible for gathering and transporting information from the end-devices to the physical gateway. Description of all communication layers (physical, link, network, transport, and application) is provided in the aforementioned document. First developments of these communication protocols and early simulations have been already conducted.
- Integration network: It is responsible for ensuring the communication between the physical gateway and the rest of the INTER-IoT system (or more specifically, with the virtual gateway). Integration tasks between a virtual

machine containing a first version of the virtual gateway and the physical gateway have already started.

Consequently, it is redefined the INTER-IoT gateway, which is considered the brain of the INTER-HARE platform and the single point of contact between the physical network and the rest of the INTER-IoT system. Due to its dual conception, it is easy to split its internal architecture into:

- Physical gateway: A combination of a wireless frontend (responsible for the communication with the rest of the transport network) and a controller (responsible for the communication with the virtual gateway). Both elements are connected through a serial link.
- Virtual gateway: A virtual entity which can be executed in a remote location, based in the Docker platform; i.e., a virtual container that provides an additional layer of abstraction and automation of operating-system-level virtualization on Windows and Linux.

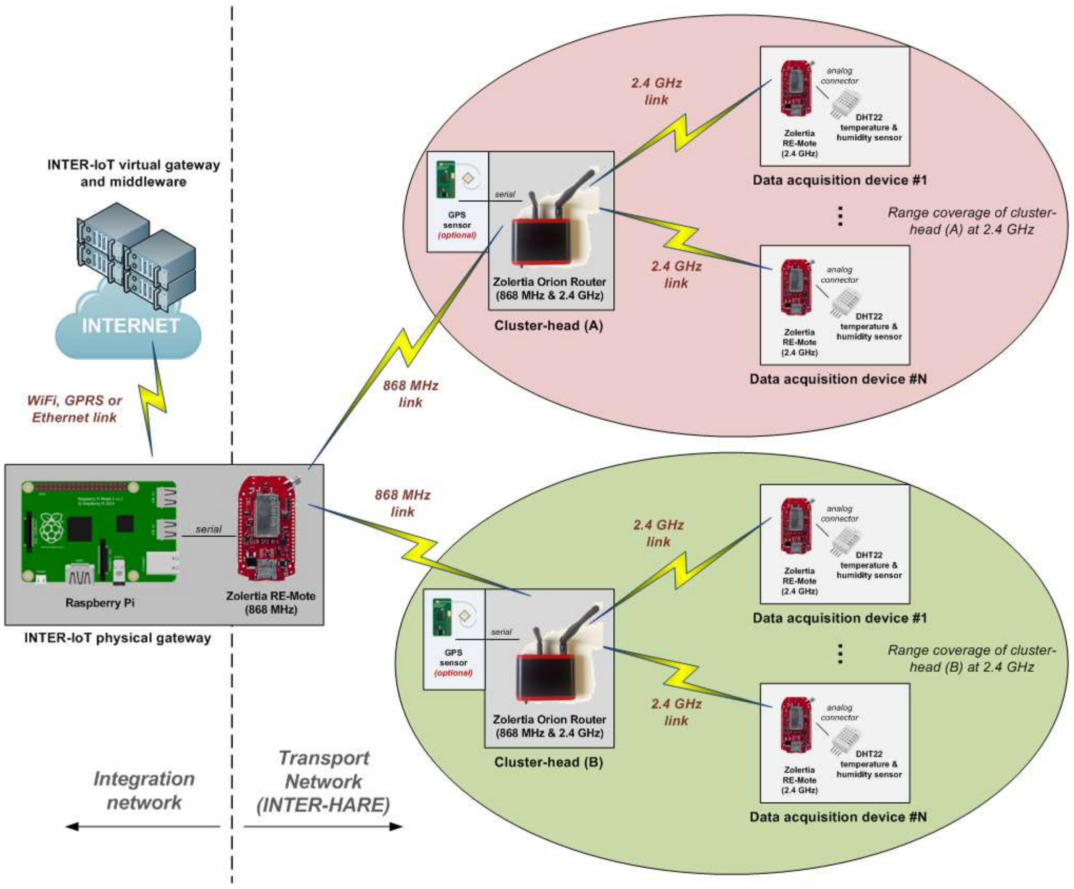

INTER-IoT virtual gateway and middleware
INTERNET
WiFi, GPRS or Ethernet link
Raspberry Pi
serial
Zolertia RE-Mote (868 MHz)
INTER-IoT physical gateway
Integration network
Transport Network (INTER-HARE)
868 MHz link
868 MHz link
GPS sensor (optional)
serial
Zolertia Orion Router (868 MHz & 2.4 GHz)
Cluster-head (A)
Cluster-head (B)
2.4 GHz link
analog connector
Zolertia RE-Mote (2.4 GHz)
DHT22 temperature & humidity sensor
Data acquisition device #1
Data acquisition device #N
Range coverage of cluster-head (A) at 2.4 GHz
Range coverage of cluster-head (B) at 2.4 GHz

Figure 137: D2D interoperability in INTER-HARE.

Progress

INTER-HARE project is advancing as expected and the main contributions of the collaboration are:

- Analysis of the existing INTER-IoT system, and definition of the INTER-HARE requirements.

• Design and development of INTER-HARE system, including the conceptual design of the INTER-HARE architecture and communication protocols and selection and validation of hardware components. • Definition of test setups for the FAT (Factory Acceptance Tests), providing a very detailed document that will provide a very good background for the SAT process. • First stages of the technical development (both in terms of internal communication within the INTERHARE network and integration with the already existing INTER-IoT system). The development is directly linked with the INTER-GW component. • Definition of the use case to develop the pilot, testing and evaluation. Regarding dissemination the UPF team participated in two brokerage events in which they presented the results of the collaboration and discussed the relevance of IoT interoperability associated to INTER-HARE: • Smart City Expo World Congress 2017 - (14-16 November 2017, Barcelona). • IoT Solutions World Congress 2017 - (3-5 October 2017, Barcelona).
Recommendations
The evaluation panel is very satisfied with the contribution, especially with the real demonstration of the hardware devices during the evaluation and the preliminary integration with the gateways, some recommendations regarding the contribution are: • Find a suitable, realistic use case in order to use INTER-HARE. • Define and analyse the scalability issues of the radio technology, the density of devices could be an issue in an environment like the port. • Identify licensing problems due to the proprietary nature of the HARE product.

Table 124: INTER-HARE Evaluation.

43 - Mission Critical operations based on IoT analytics (MiCrOBIoTA)
Nemergent Solutions S.R.L.
Description
Although Mission Critical (MC) communications have been traditionally supported over private radio technologies, the current trend is to foster interoperability and a more competitive marketplace based on open standards and mobile broadband radio technologies. In the last two years, the 3GPP has significantly advanced in the normative work concerning MC-PushToTalk (MCPTT) in Release 13 and MC-Video and MC-Data communications (such as file transfer or short data messaging) in Release 14. These technologies have been initially designed for covering the needs of the Public Safety community. Yet, new study items are now launched to extend the scope of this type of technologies to other mission critical scenarios such as railway and maritime control communications in the future Release 15. In parallel, the 3GPP has also progressed in the definition of Internet of Things (IoT) technologies based on mobile broadband networks, resulting on Cell-IoT (C-IoT) and Narrowband IoT (NB-IoT). However, the adoption of IoT communications for mission critical operations is not mature enough to provide a reliable framework for converged

and interoperable mission-critical operations. First, C-IoT and NB-IoT technologies have not reached the deployment level of other IoT technologies. Second, the adoption of IoT in MC communications has not received yet the required consideration in the standardisation process.

The anticipated benefits of interoperable “Mission Critical operations based on IoT analytics” (MiCrOBIoTa) are related with the previous technological proposals. MC organisations worldwide are looking at the technology as a driver for enhanced life-saving operations, but they are facing the traditional problems of segmented markets and non-interoperable technologies. In this sense, the objectives and proposals of the INTER-IoT architecture and components provide a perfect framework for advancing mission critical systems towards the adoption of interoperable IoT platforms with the required privacy, security and QoS requirements.

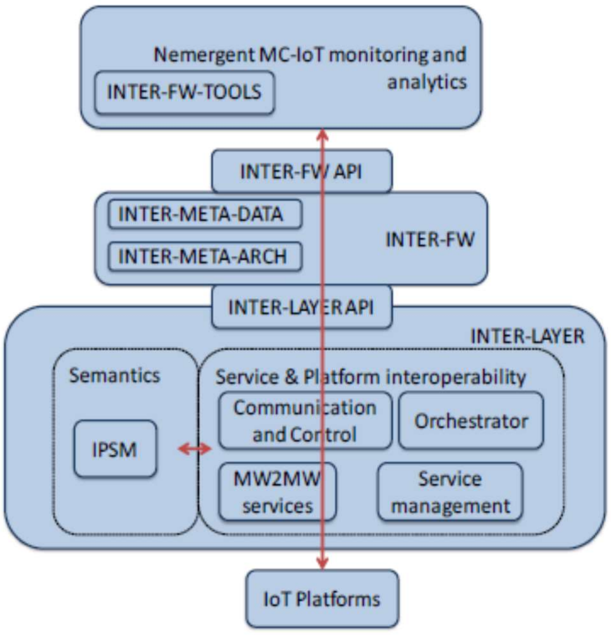


Figure 138: MiCrOBIoTa integration in INTER-IoT architecture.

MiCrOBIoTa contribution, aims at exploiting the INTER-IoT platform as a mean to gather information from heterogeneous sensors in a converged way. As a result, Nemergent will be able to integrate a new “IoT monitoring and analytics” component in its mission critical product portfolio, and especially into the Nemergent Control Room application. Specifically the proposed results and goals of MiCrOBIoTa are:

- To define (in collaboration with the INTER-IoT team) a cross-domain IoT scenario in the scope of mission critical operations, where different IoT devices related to port logistics and mobile on-body health sensors can provide valuable field information to mission critical operators.
- To contribute to INTER-IoT activities related to the definition of the common semantics and ontology regarding the specific mission critical scenarios.
- To develop the “Nemergent MC-IoT monitoring and analytics” component, which will interface with the INTER-IoT platform and with the Nemergent Control Room application.
- To adapt the Nemergent Control Room application to perform basic IoT-related operations, such as displaying the relevant events information and enabling

mission critical actuations through the rest of the Nemergent framework components.
- To actively support the INTER-IoT INTER-DOMAIN pilot and validation activities through the use of the evolved Nemergent Control Room.
- To provide the required feedback to the INTER-IoT consortium.
- To perform the proper dissemination and communication activities.

From the Business scenarios and Scenario use cases identified in INTER-IoT, his contribution will be mainly related to "INTER-LogP/Health scenarios: Accident at the port area". The overall idea is that the different Nemergent technologies are able to provide an enhanced emergency management framework to coordinate the port authority with other first responders.

Nemergent aims at contributing to this use case by developing an external application which will be able to interface with the different IoT platforms involved. The external application will gather accident alarms and will be able to identify possible sources of information related to the incident. The GUI will enable the CCE staff to send the most relevant information (guided route, health monitoring and location of actuators) to the different emergency units (e.g., an ambulance) connected to the integral system.

Progress

MiCrOBIoTa collaboration is advanced as expected, main contributions provided are related with:

- Review of the state of the art concerning the inclusion of IoT sources into the world of emergency management.
- Initial external discussions with partner companies, such as RKL INTEGRAL and ETELM SAS, related to the digitalisation process of companies' protection plans and the potential inclusion of IoT sources in them. Initial joint discussions concerning the proposed trial scenario with the rest of the INTER-IoT consortium (coordinator, stakeholders and other Third Parties).
- Design of the overall MC-IoT system and associated documentation.
- Analysis of the required extensions to the Nemergent Control Room system: modifications to the backend and front-end components, extensions to the connectors and design of new message formats.
- Analysis of the required extensions to the Nemergent MCPTT system: integration of talk group membership management and INTER-IoT players.
- Definition of the Nemergent Factory Acceptance Tests (FAT) and creation of the FAT document templates.
- Deployment of the required tools for component development and for FAT.
- Deployment and adaptation of an MQTT server as input to the MC-IoT system.
- Analysis of EDXL language format to be used in the MC-IoT system (in a future potential integration with other open caller's outcomes).
- Initial implementation activities at the Nemergent backend and front end so as initial tailored configuration at the Nemergent MCPTT system.

Nemergent has performed different dissemination actions:

- PSCE conference 2017 (28-29 November 2017) in Madrid Spain. Where presented the Nemergent activities (including cite to INTER-IoT project) and discussed the applicability of IoT to Public Safety.
- MILIPOL Paris 2017 event (Paris, 21-24 November 2017), where discussed the applicability of IoT to Public Safety and other sectors

• At local scope, "Jornadas de Gerencia de Riesgos y Emergencias" (18-19 May 2017).
Recommendations
The evaluation panel is very satisfied with the contribution, especially with the industrial dissemination actions and the standardization window that could be opened in the security and safety market for interoperability. Some recommendations for the improvement of the collaboration: • Although the integration of MiCrOBIoTa is planned through INTER-API, Nemergent is encouraged to develop a NodeRed node, in order to facilitate integration of their solution in a seamless way through AS2AS component. • The definition of the use case needs some refinement, mainly in the wearables and workers tracking. • Collaboration with U Twente and the ontology for safety and security will be highly appreciated.

Table 125:MiCrOBIoTA Evaluation.

49 - SENSHOOK
Irideon SL
Description
Many companies want to develop IoT products, however lack the necessary financial and human resources using existing methods. To address this need, IRIDEON has developed Senscape®, a disruptive, standards based platform which enables fast time-to-market development of IoT sensor-server applications and information services. To date, we have focused on the development of IoT devices using our own hardware and embedded operating system - SENSOS. Now, we wish to contribute to the INTER-IoT project with a new open tool called SENSHOOK, to enable full interoperability of our Senscape® IoT platform with other IoT platforms and services, and fully exploit the unique selling points of our existing technology. This will allow us to address a wider range of customers and applications, and to grow our revenue and the company, via more customer projects, via licensing of Senscape®, and exploitation of SENSHOOK as an open-source tool compliant with the INTER-IoT framework. Hardware devices to be integrated in the pilot have been developed by the company and use standard interfaces. Two products: tuatara and flyingdragon boards. The baseboards use SENSOS embedded operating system. SENSOS delivers best in class performance vs. power consumption and supports a set of advanced features including: remote device smart energy management; remote auto-check, remote data preview; multi-device time synchronization; remote updates; and secure communications.

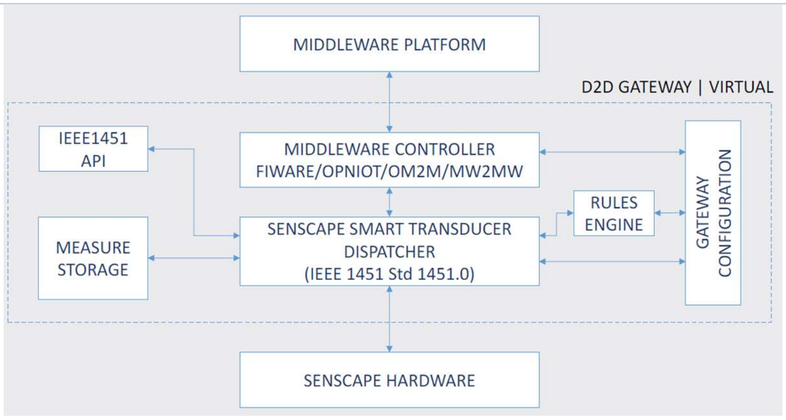


Figure 139: SENSHOOK Block Diagram.

Summary of SENSHOOK specifications:

- Senscape Interoperability with INTER-IoT will be done at device level.
- Senscape IoT devices will be accessed/controlled through a unifying interface and integrated into one of the following IoT platforms: FIWARE, OPENIOT, OM2M and/or MW2MW.
- This interoperability solution at device level will be achieved through a Device to Device Gateway (D2D Gateway).
- Senscape devices will approach for the implementation of a virtual Gateway.
- This Gateway will be developed in JAVA using the OSGi framework.
- All components of the Gateway will be packaged as OSGi bundles.
- IEEE1451 standard specification as basis of the gateway dispatcher.

Implementation of the IEEE1451 specification as basis of the SENSHOOK gateway dispatcher and additional system blocks like the API, configuration and measure storage. The driver with the standard protocol will be included in INTER-IoT gateway as an OSGi component.

The use case is related with the deployment of Zikka mosquito traps. The collaboration will include the manufacturing of 3 mosquito trap devices to be tested with SENSHOOK tool. These devices have been programmed to communicate with the tool via 3G using the IEEE1451 specification. The application developed by IRIDEON uses machine learning components to identify mosquito species, to map and model the distribution of vectors of disease and to plan surveillance control programs.

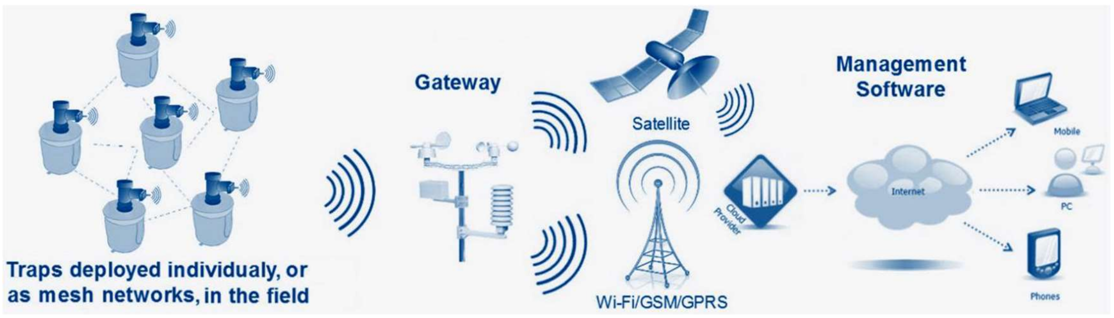


Figure 140: SENSHOOK use case.

Progress
IRIDEON contribution is advancing as expected, main contributions have been developed in line with the proposed workplan: • Study of INTER-IoT project. The information, deliverables and supporting documentation provided by the consortium were studied to gain an enhanced understanding of the project needs and scope. • First specifications regarding the implementation of the SENSHOOK tool in the Senscape® IoT platform defined preliminarily. • Partial implementation of SENSHOOK according to INTER-IoT requirements using IEEE1451 standard specification as basis of the system dispatcher. • Preliminary tests of SENSHOOK performed in IRIDEON's facilities to evaluate the performance and benefits of the tool. • Draft Business models and Exploitation plans for Senscape® + SENSHOOK and the Smart Mosquito Trap to be piloted in the port of Valencia. • Specification of the data model and OSGi bundle to be integrated in the architecture. IRIDEON adapts one of the major international standards for the control and reading of smart transducers: IEEE1451, and makes it compatible with existing lightweight data communication protocols and data formats used in IoT applications. With a sensor-centric approach, in which each sensor or actuator can be discoverable, accessible, and usable via TEDs described in the standard, and sensor data can be automatic and correctly transformed before being processed and analyzed for an upper application layer. IRIDEON has had a very active dissemination activity of its own products and the collaboration with the INTER-IoT project. Participation in events to disseminate Senscape® and SENSHOOK: • Healthio Congress 2017. • Open EUREKA Innovation Week 2017. • Barcelona INNOVA 2017 • Construmat 2017 • BizBarcelona 2017 • IoT Solutions World Congress 2017 • Smart City Expo World Congress 2017.
Recommendations
The evaluation panel is very satisfied with the contribution, however some recommendations and improvements are required: • Clarification of the licensing issues and open source components to be used within the collaboration and how they will affect INTER-IoT. • IRIDEON has developed a virtual gateway following INTER-IoT architecture, analysis of the potential integration at virtual level instead than at physical level has to be further evaluated. • Definition of the interface between Senscape boards and physical gateway have to be revisited in order to avoid communication errors. • A UPV developer will join IRIDEON and UPF in a code camp in order to define different tips for development in Barcelona, other interested partners related with D2D integration may join.

Table 126: SENSHOOK Evaluation.

52 - ACHILLES: Access Control and autHenticatIon deLegation for interoperabLE IoT applicationS

Athens University of Economics and Business – Research Center (AUEB)

Description

Nowadays, “smart” Things have become an integral component of many systems. Industrial production, health services, traffic monitoring, and many other fields have been greatly improved by the widespread adoption of sensors, actuators, embedded systems, identification technologies, and mobile devices. With the pervasiveness of and advances in networking technologies and the mass production of smart(er) Things, we are on the verge of breaking the currently existing vertical application silos and move towards a horizontal Internet of Things (IoT). The IoT will enable the creation of general purpose applications that will harvest the full potential of the unique features of smart Things.

This vision of the IoT raises significant security and privacy concerns. With applications impacting not only the virtual world, but now directly the real world, concerns are widespread and justified. Things can be intrusive, have access to sensitive information, can be easily tampered with and at the same time Things might not have enough computational, storage, or energy capacity, and in general they have significant limitations.

ACHILLES will provide a solution to the problem of access control and Thing authentication for the IoT. Access control and endpoint authentication in the IoT is a challenging problem. Things are usually small devices with limited storage capacity, power, energy, and processing capabilities, in order to be inexpensive and practical. In many cases Things are “exposed” to tampering, whereas in many application scenarios, after Things are deployed, it is not easy to access them. Things usually are not able to perform “heavy'” tasks, such as complex cryptographic operations. Storing user credentials or any other sensitive information in a Thing creates security risks, adds storage overhead, and makes security management an impossible task. When it comes to interoperable applications, Things (or even gateways) cannot interpret complex business roles and processes. Moreover, companies are not willing to share sensitive information about their users with a Thing (or a gateway), even if this information is required by an access control mechanism, neither do they want to invest in yet another security system.

The ACHILLES project will overcome these limitations by allowing the delegation of security operations to a third party, referred to as the Access Control Provider (ACP), which can be implemented by a trusted separate entity, or even the service provider itself. The ACHILLES concept is depicted in the following figure.

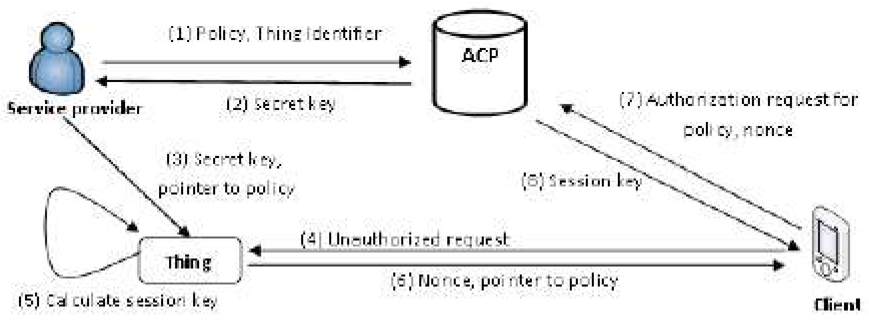

(1) Policy, Thing Identifier
ACP
Service provider
(2) Secret key
(7) Authorization request for policy, nonce
(3) Secret key, pointer to policy
(8) Session key
(4) Unauthorized request
Thing
(6) Nonce, pointer to policy
(5) Calculate session key
Client

Figure 141: ACHILLES concept architecture

The main idea of the ACHILLES concept is that IoT service providers store access control policies in ACPs and in return ACPs generate secret keys which are stored in Things. These keys are generated, during a setup phase, using a secure hash with input the Thing identifier. Additionally, Things are configured with pointers (e.g., a URL that points to an ACP and a particular file) to the access control policies that protect sensitive resources. Every time a client requests access to a protected resource the Thing uses a secure hash function to generate a session key. The secret key used by that function is the key generated by the ACP and the hash inputs are: (a) the pointer to the policy that protects the resource and (b) a random nonce. The Thing transmits the nonce and the pointer to the client and the client requests authorization from the appropriate ACP (over a secure channel). The ACP has all the necessary information required to calculate the session key: if the client is authorized, the ACP calculates the session key and transmits it back to the client. Provided that: (i) the Thing has not lied about its identity and (ii) the messages exchanged between the client and the Thing have not been modified, the Thing and the client end up sharing a secret key. This key can be used for securing subsequent communications (e.g., by using DTLS).

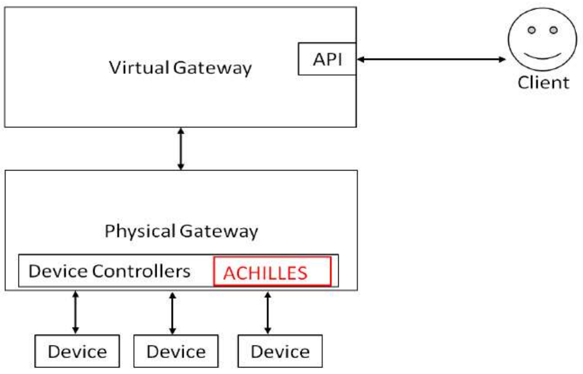


Figure 142: ACHILLES integration in INTER-IoT Gateway

ACHILLES component will be installed in the gateway as an independent bundle, and will provide access control to the architecture.

Progress

ACHILLES contribution is advancing as expected, main contributions have been developed in line with the proposed workplan:

- Development of an open protocol to allow existing user management systems to be used for access control, by all layers of the INTER-IoT platform. Policies will be built and managed in these systems and Things and gateways will be oblivious to them.

- Definition of the exploitation plan and business models to facilitate interoperability, innovation, and B2B services, by allowing the use of (pointers to) policies (e.g., https://companyA/customers) without needing to have access to the policy implementation details (e.g., who the customers of Company A are).
- Definition of the security management framework, by enabling access control policy modifications without communicating with the Things (or gateways). Service providers will be able to modify security policies even after Things and gateways have been deployed.
- Definition of tools for things gateway mutual authentication and appropriate APIs for the INTER-FW, which will allow end-users to create and access protected resources.
- Preparation of the integration plan that was included in D6.1, so as the Factory Acceptance Testing specific to the contribution.
- Preliminary integration in the gateway with the definition of the interfaces and the structure of the software artifact. Main premise is that things and gateways will only have to follow a simple communication protocol and will not have access to any information related to end-users.

No dissemination or communication action has been identified by the research group.

Recommendations

The evaluation panel is very satisfied with the contribution, however some improvements are recommended:

- Security definition has to be in line with INTER-IoT IdM proposal, so it is recommended to modify the architecture accordingly.
- Consider the possibility of extending the Access Control mechanism to other layers like middleware and AS2AS if needed.
- Contribute to dissemination actions, mainly in the industrial area.
- Identify potential standardization paths for the contribution in order to increase visibility.

Table 127: ACHILLES Evaluation.

53 - A Semantic Middleware for the information synchronization of the IoT devices

University of Twente

Description

An Early Warning System (EWS) is an integrated emergency system that provides services to monitor, detect and alert emergency risks. The core idea of INTER-IoT-EWS project is to develop an interoperable EWS on top of INTER-IoT application layer (AS2AS, NodeRed) to detect and alert accidents and risks of accidents in the Valencia port area, interoperating with IoT platforms, medical wearable devices (Shimmer ECG) and Emergency Management Systems (open call partner Nemergent solutions).

Besides the EWS, the INTER-IoT-EWS collaboration offers bi-directional semantic translations between the Smart Appliances REFerence ontology (SAREF) and the

W3C Semantic Sensor Network Ontology (SSN), to be configured within the Inter-Platform Semantic Mediator (IPSM). This also includes the investigation of existing ontologies and possible alignments and extensions. For example, the extension of SAREF for e-Health (based on HL7) and logistics (based on LogiCO).

In accordance with the INTER-IoT challenges, the collaboration supports the achievement of semantic and syntactic interoperability among IoT platforms, i.e. enable data to be understandable for both sender and receiver platforms. In particular, we focus on coordinating emergency services based on IoT devices, alerting the involved parties, e.g. emergency command control, haulier, terminal operator, first responders and employees, when an accident occurs.

The objective is to use and contribute to the semantic interoperability capabilities of the INTER-IoT framework through the IoT-based EWS, enabling data exchange among heterogeneous IoT platforms by developing emergency application services that require IoT semantic translations from IPSM.

The innovation capacity of the solution is leveraged by stressing the role of the OASIS Emergency Data Exchange Language (EDXL) for emergency services, applied in cross-domain scenarios in logistics/transportation (INTER-LogP) and healthcare (INTER-Health). The collaboration exploits standards, ontologies and data models for the description of decision rules to detect emergency situations, e.g. W3C SSN, SAREF, OASIS EDXL (CAP, SitRep, TEP, HAVE, DE, RM) and, if necessary, consider parts of standards/ontologies related to e/m-Health and logistics.

INTER-IoT-EWS project is based on U. Twente current research in semantic interoperability of EWSs for disaster management, which combines our recent research in e-Health standards, context-awareness and logistics. The potential impact of this project on the IoT solutions industry is innovation regarding data exchange of these different data formats to detect and alert emergencies. In particular, a side-effect benefiting INTER-IoT will be the enablement of interoperability between the most notable ontologies in the IoT context, namely SSN and SAREF. Furthermore, it proposes a low-cost business model for logistics, healthcare and insurance companies. For example, in the port of Valencia (transportation use case), transportation companies, haulers, terminal operators and insurance companies can benefit from the IoT EWS by reducing disaster risks involving the involved employees (e.g. trucks' drivers) and the goods being transported.

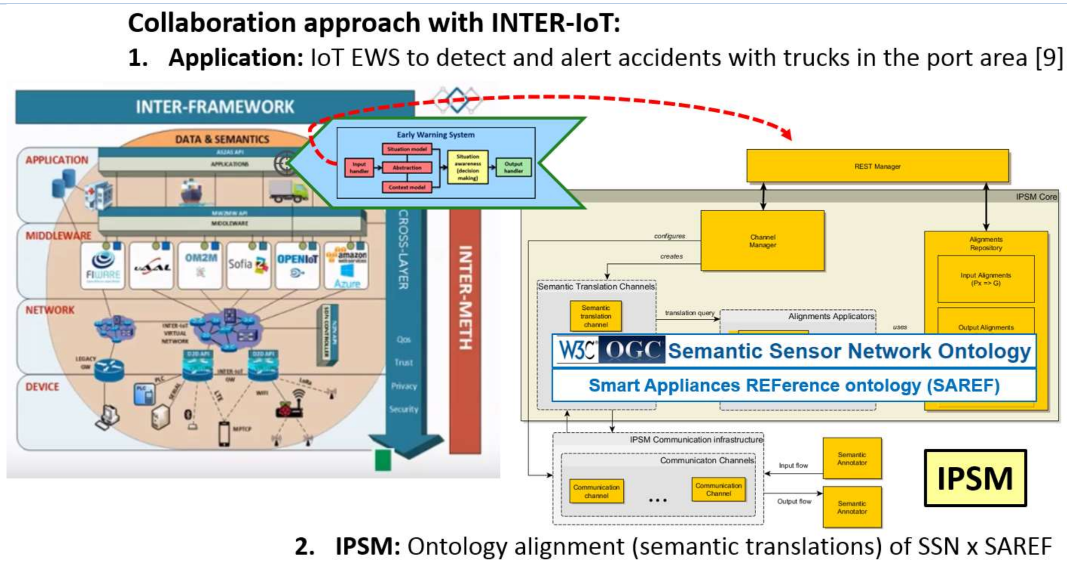

Collaboration approach with INTER-IoT:
1. Application: IoT EWS to detect and alert accidents with trucks in the port area [9]
INTER-FRAMEWORK
DATA & SEMANTICS
APPLICATION
MIDDLEWARE
NETWORK
DEVICE
CROSS-LAYER
INTER-METH
Early Warning System
REST Manager
IPSM Core
Channel Manager
Alignments Repository
Semantic Translation Channels
Alignments Applicators
Semantic Sensor Network Ontology
Smart Appliances REFerence ontology (SAREF)
IPSM Communication infrastructure
Communicaton Channels
Semantic Annotator
IPSM
2. IPSM: Ontology alignment (semantic translations) of SSN x SAREF

Figure 143: Conceptual architecture of semantic brokering for context-aware decision support

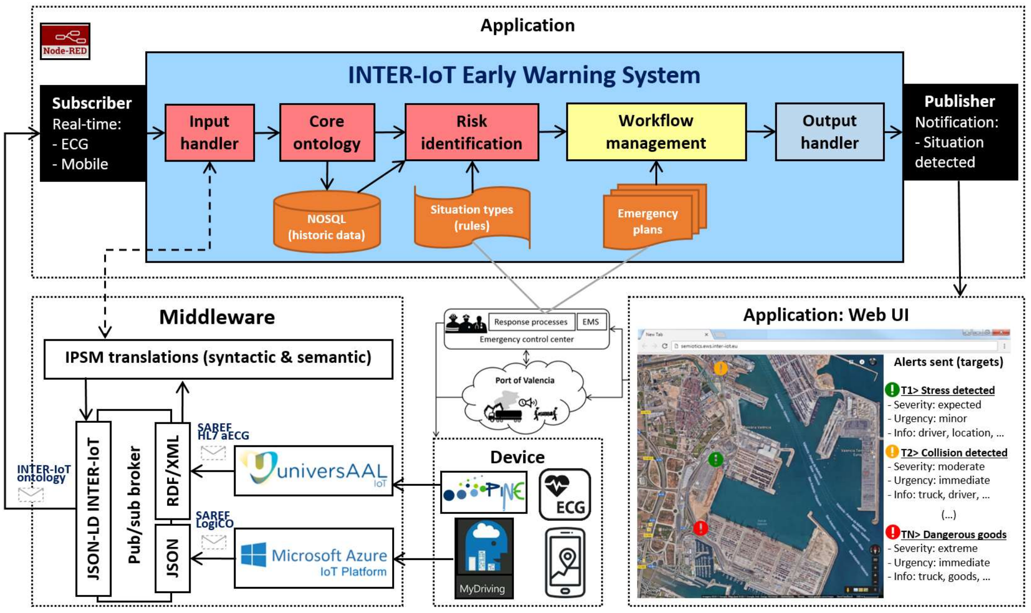


Figure 144: Integration in INTER-IoT

Progress

The collaboration is advancing as expected, and the main developments associated with it are:

- Exploitation of ontologies and standards that can be applied within INTER-DOMAIN use cases. Ontologies include SSN, SAREF, LogiCO, EDXL, incident management ontology. This development is included in deliverable D1.1.
- Configuration of SAREF - SSN translation and deploy in IPSM – this activity is in progress. Samples generated and initial tests performed over IPSM (INTER-IoT test environment).

- Description of decision rules for emergency services, framing the situations identified for each use case, this is included in deliverable D2.1.
- Definition of 5 use cases conceived and described in D2.1.
- Design EWS and integration plan: architecture, components and initial tests performed (e.g. MyDriving application). INTER-IoT D6.1 and D6.2 contain the U. Twente contribution.
- Preliminary integration of components of the EWS and integration with INTER-IoT components (IPSM and INTER-MW).
- Business model, exploitation plan and economic evaluation. Business model extended with cost estimative of the IoT EWS and the responsible (stakeholders) for each component.

The research group has participated in different scientific dissemination activities:

- João Moreira, Laura Daniele, Luis Ferreira Pires, Marten van Sinderen, Katarzyna Wasielewska, Pawel Szmeja, Wiesław Pawłowski, Maria Ganzha, Marcin Paprzycki “Towards IoT platforms’ integration: Semantic Translations between W3C SSN and ETSI SAREF”, Semantics. Workshop Semantic Interoperability and Standardization in the IoT (SIS-IoT), Amsterdam, 2017.
- João Moreira, Luís Ferreira Pires, Marten van Sinderen, Roel Wieringa, “Semantic Model-Driven Development of Interoperable IoT-based Emergency Services: the INTER-IoTcase study”, CTIT, Enschede, 2017.
- João Moreira, Luís Ferreira Pires, Marten van Sinderen, Roel Wieringa, Prince Singh and Patrícia Dockhorn Costa , “Improving the semantic interoperability of IoT Early Warning Systems: the Port of Valencia use case”, I-ESA 2018: "Enterprise Interoperability VIII (2018)", 2018

Recommendations

The evaluation panel is highly satisfied with the contribution, specifically with the improvement in the ontology and the use of IPSM. U. Twente is recommended to:

- Continue integration with IPSM and improvement of the emergency ontology.
- Participate in standardization forums like ETSI or W3C as the contribution to semantics is very relevant and there are no other related work.
- Enlarge collaboration with Nemergent use case, as the semantic component can be of use.
- As it is a small contribution, there is a risk if the third party does not focus on a specific use case.

Table 128: INTER-EWS evaluation.

66 - A Semantic Middleware for the information synchronization of the IoT devices

Institute of Industrial Technologies and Automation - National Research Council (ITIA-CNR)

Description

The collaboration proposes the development of a new component, called Semantic Middleware, to be added within the set of middleware modules supported by the INTER-IoT approach. The new component aims to enable near real-time signaling capabilities to all the devices connected the IoT platform, also thanks to the interaction with various modules provided by of the INTER-IoT approach.

The current state of the art points out that many solutions available in literature enable the data exchanges among the devices. However, as the content meaning of the exchanged data expressions is left to the interpretation of the receivers, these solutions do not support the information synchronization, thus limiting the semantic interoperability of the involved devices. In order to contribute to bridge this gap, Semantic Middleware allows to express all the exchanged information (included the synchronization requests) under the form of semantic model. This way, it allows a more flexible and adaptable characterization of the data subscribers' needs and of the data providers' capabilities, while it enhances the interoperability between all the involved devices.

The Semantic Middleware allows to express both the synchronization requests and all the exchanged information under the form of semantic model. Such a semantic-enabled solution enable a more accurate, flexible and adaptable characterization of the subscribers' needs and of the providers' capabilities, while it contributes to enhance the interoperability between all the involved devices.

The motivation behind the proposal of the Semantic Middleware is in line with the overall goal of the INTER-IoT project ("providing an interoperable and open IoT framework, for seamless integration of heterogeneous IoT platforms, regardless of the application domains"). Specifically the Semantic Middleware can mainly contribute to the INTER-IoT objective of defining techniques and tools for interoperability at the different IoT Platform layers. In fact, acting as a central point which dispatches information between devices and applications, it addresses the interoperability objectives of INTER-IoT at a semantical level. Thus, it supports the INTER-FW module in the role of facilitating the creation of an ecosystem of interoperable and open IoT platforms. Thus, development time of novel IoT services and applications can be shortened, and these services can be provided a top interoperable IoT platform maturing the IoT interoperability.

Moreover, the activities performed to develop the Semantic Middleware can be gathered into the activities to design and implement smart IoT application service gateway and virtualization, which are considered essential actions to implement the above mentioned objective of the INTER-IoT project.

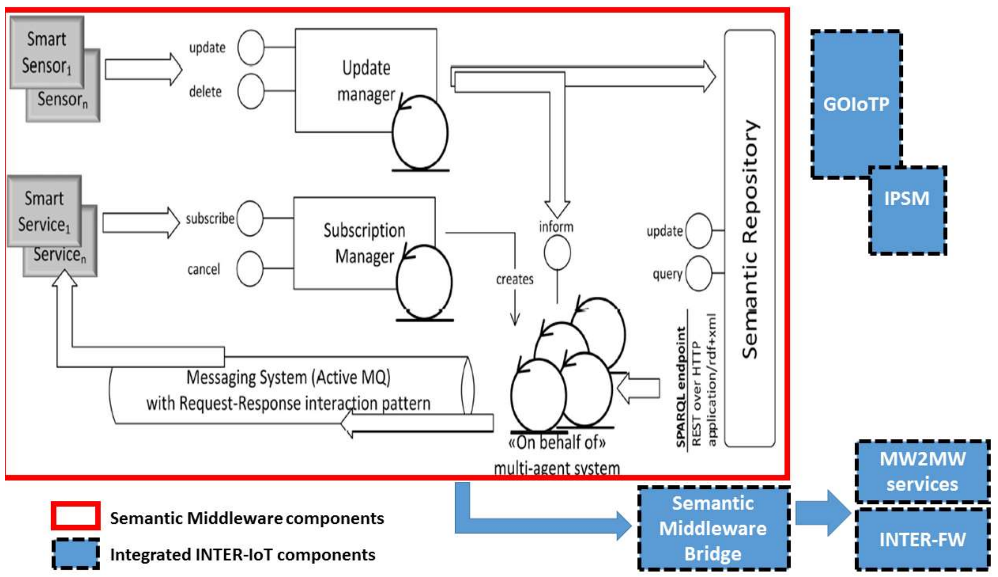


Figure 145: Integration in INTER-IoT

Progress

The progress is advancing as scheduled main contributions are:

- Definition of the implementation of the “SM Bridge" which handles the connection of SM with all the underlying platforms, with the MW2MW services and with the INTER-FW.
- As SM is agnostic to the metamodel of the IoT platform ontology, SM uses the GOIoTP defined in INTER-IoT, paired with an application ontology specific of the analyzed scenario. A link between application ontology and GOIoTP is under implementation process.
- Definition of the use case related with the pallet position.
- Preparation of the integration plan that was included in D6.1, so as the Factory Acceptance Testing specific to the contribution.
- Definition of initial business model and exploitation

Recommendations

The evaluation panel is satisfied with the contribution, some recommendations for the collaboration:

- Clear definition and specification of the ontology in order to be aligned not only with GoIoTP but with the other extensions developed by third parties.
- Identification of the main features of the Semantic Middleware in order to align it with the development of the bridge.
- Clarification of the use case in order to align it with the needs of the different stakeholders.

Table 129: Semantic Middleware Evaluation.

70 - SecurIoTy - security for the IoT

AvailabilityPlus GmbH

Description

Security is paramount for the safe and reliable operation of IoT connected devices. Currently there is consensus that in order for IoT to become widespread, security issues have to be resolved. There is less consensus on how to best implement security in IoT. In our approach SecurIoTy, we address all IoT security dimension such as confidentiality, integrity and availability for data in transit and at rest.

SecurIoTy is a smart cyber security solution to secure the internet of things using crypto proxy technology. SecurIoTy provides an important building block for the establishment of safe, reliable and large scale IoT systems. SecurIoTy protects data at rest and in transit with a zero-knowledge policy. SecurIoTy integrates at the application or middleware layer of Inter-IoT.

SecurIoTy employs the Rational Unified Process (RUP), an iterative software development process framework originally developed by IBM. RUP gives a detailed canvas for software development and integration projects. Besides implementing the

RUP framework, the research team has implemented these components that are documented in the provided deliverables:

- Software requirements specification: Description of the functional and non-functional requirements
- Use cases: Definition of the use cases.
- Test cases: Definition of the test cases, test plan and acceptance criteria according to global project specifications, implemented automated and semi-automated tests
- Data model: Description of the SecurIoT data model
- Software architecture: Definition of the software architecture
- GUI design: Definition of the user interface as far as necessary given machine to machine communication
- Implementation: Iteration 1 of adapting the SecurIoT system to INTER-IoT layers.

Currently, it has set up server infrastructure in hardware and software including multiple storage sites at multiple physical locations. Also a security gateway to securely expose our services through the internet. As a result, standalone SecurIoTy services can now be used at this web site: https://interiot.docraid.com.

Preliminary integration with INTER-IoT has been performed through AS2AS and the team has set up the Node-red environment to hook up SecurIoTy.

As detailed in the INTER-IoT Architecture, security is the cross layer between applications, middleware, network and devices. The proposed SecurIoTy cross layer will provide security mechanism such that devices will be enabled to secure data while in transit as well as at rest.

SecurIoTy will provide communication mechanisms which allow devices to be shielded and to communicate via protected channels as well as store data in protected storages. As such our proposed collaboration will interact and provide interaction mechanisms on the device level, the network level, the middleware level and the application layer.

The application service layer component offers primarily HTTP(S) and potentially also REST interfaces to service the INTER-FW parts of the INTER-IoT Architecture. At the middleware layer, we will initially offer interfaces to connect to cloud storage services which are operated by AvailabilityPlus (DocRAID® CloudRAID), and optionally other storage providers. On the network and device level it provides interfaces to shield devices and let devices communicate via protected channels.

SecurIoTy will thus provide a security framework which:

- serves all INTER-IoT Architecture layers (application, middleware, network, devices)
- enables scalable security to address the security needs of a particular application, with a particular focus on logistics use cases.
- (3) addresses all security dimensions.

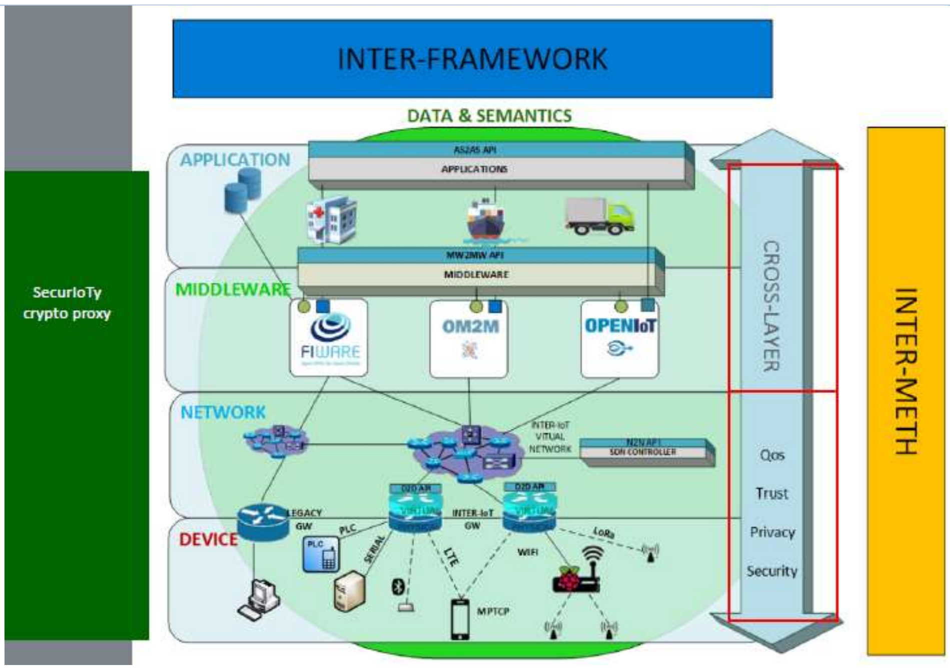


Figure 146: SecurIoTy Integration in INTER-IoT

Progress

The collaboration is advancing as scheduled, main progress is related with the following lines:

- Software requirements specification, the Use cases, Test cases, Data model and Software architecture. Adaptation of test cases from iteration 1 to the INTER-IoT requirements. Set up of automated and semi-automated tests for the software. Alignment of the test cases with requirements from INTER-IoT. Results are delivered in the document "D6.2_Factory_Acceptance_Test_Plan_securIoTy_version5".
- Extension of the GUI design to interface design to reflect requirements from INTER-IoT. Initial integration scenarios of SecurIoTy and INTER-IoT are: a) authentication, b) sensor hub, c) reporting, storing/retrieving, logging and documentation, provided this, SecurIoTy will need access to the respective services of INTER-IoT. Given the current architecture of INTER-IoT, SecurIoTy will be useful for gathering data from sensors and storing the reports in SecurIoTy. Node-red is used to manage sensor nodes, SecurIoTy will be used as a building block to store/retrieve data. As a result, AvailabilityPlus has set up interfaces to connect and integrate with the Node-red architecture.
- Given the integration scenarios iteration 1 was finished of implementing interfaces and to adapt the architecture of our solution to reflect this concrete application scenario and to set it up tangibly. AvailabilityPlus has set up server infrastructure in hardware and software including multiple storage sites at multiple physical locations and a security gateway to expose services through the internet. The security gateway supports these protocols: http, https, WebDAV.

Recommendations

The evaluation panel is satisfied with the contribution, however some recommendations are issued:

- Licensing has to be clarified as DocRaid is a proprietary system. The connector needs to be open source and has to be perfectly documented.
- Devices for the piloting have to be identified.
- Further connection to the INTER-IoT architecture, e.g. INTER-FW has to be defined.
- Collaboration with other third parties in order to find points of agreement (e.g. ACHILLES) is advised.

Table 130: SecurIoTy Evaluation.

74 - E3Tcity Smart City Platform and Devices Integration
E3TCity S. L.
Description

This project is aimed to integrate E3Tcity vertical platform with the Middleware Layer of INTER-IoT Inter Layer Platform. This development will provide INTER-IoT with a whole device/cloud/app vertical solution to be applied in the Smart Port pilot. E3Tcity Smart City platform allows control and monitoring of different types of installation, which include lighting, irrigation, HVAC, energy measurement, sensing, and generally any system to be controlled remotely.

The platform is currently being deployed in more than 20 towns in Spain, with services spanning from public lighting control to mobility control solutions such as traffic, parking, crowd, traffic lights, irrigation and water quality, and heating, ventilation and air conditioning control.

Several interoperability options have been reviewed, choosing two of them as the most suitable for the project:

- Device to Device Interoperability
- Platform to platform Interoperability.

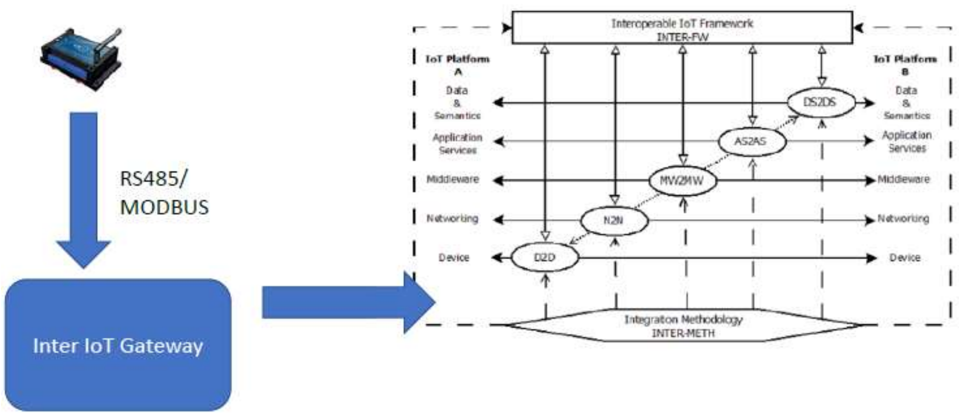


Figure 147: E3TCITY D2D interoperability

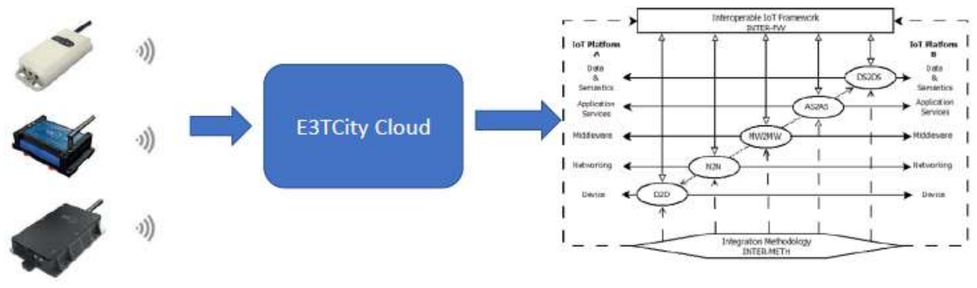


Figure 148: E3TCITY MW2MW interoperability

The collaboration will include the platform and the developed components in the smart lighting pilot in the port premises, connecting some of the nodes to the INTER-IoT GW and others to the MW2MW interoperability components. Both IoT platforms NPV and VPF will be used to demonstrate the feasibility of the proposal.

Progress
The collaboration is progressing as expected, main advances: • E3Tcity is working on a driver for INTER-IoT gateway that will allow the device to communicate with e3tcity controllers. Driver will be ready according InterLogP pilot. • E3Tcity is working on platform to platform integration with INTER-IoT MW2MW layer. Integration will be tested in InterLogP pilot. • Definition of the FAT for the collaboration, included in D6.2 • Definition of the integration in the smart lighting pilot. • Start of deployment of lights, nodes and PIRs
Recommendations
The evaluation panel is very satisfied with the contribution some recommendations: • Provide better documentation for the testing and evaluation, checking the documentation provided by other third parties. • Identify the connection mechanism of the PIR. • Interact with other third parties, mainly those working with semantics as the smart lighting use case will be of interest. • Participate in some dissemination events related with smat cities.

Table 131: E3TCity Evaluation.

5 Conclusions

The document has described the Factory Acceptance Test and initial integration plans from the two pilots included in the proposal, i.e. INTER-LogP and INTER-Health. The first pilot due to its extension and needs has been split in different use cases. Additionally, for INTER-DOMAIN pilot, INTER-IoT consortium introduced twelve third parties, ten small and two large contributions that are bringing different technologies to INTER-IoT so as different use cases.

Following an industrial approach the integration has been structured in three steps, first one was included in D6.1, in which the pilots, use cases and technologies to be used were described; D6.2 provides FAT documents and D6.3 will provide Site Acceptance Test (SAT) documents. Once integration is provided and testing of the integration validated, the different pilots and use cases will go through evaluation process defined in WP7 and will validate the different KPIs specified.

The FAT documents provided by the internal pilots and by the third party collaborators, have followed a common template in which they provided the information related with the pilot and use cases. The information is maintained in a separate document per pilot, so the working documents are fourteen separate files that for commodity have been integrated and adapted in a single document.

The goal of this process is that the different systems will then undergo their own defined FAT to test the readiness of the system. This is done in a LAB setup which approaches the actual field deployment as much as possible. When the FAT has been successfully executed and has been approved the system can advance to field integration and undergo the SAT. The different contributions are advancing in this process, and it is reflected in the midterm evaluation carried out in M25 (January 2018), and reflected in section 4 of the document.

The FAT documents describe all aspects of the FAT, form defining the versions of the used components and deliverable checklist up, test setup, tooling, test description, etc. to be able to test the readiness of the system under test. The document from each pilot contains the working documents of each pilot and use cases (where needed). The FAT documents will evolve as the pilot responsible carry out the testing. Currently all of them are involved in this process and integration is under execution.

The template of the FAT document is structured in eight sections: (i) System description; (ii) Use case oriented pilots; (iii) Deliverables and version overview; (iv) Requirements and scenarios; (v) Test environment; (vi) Test description; (vii) Test outcome overview; and (viii) Integration ethics and security.

The two internal pilot of the project, have advanced the integration activity before the start of WP6, so activity is more advanced and in lab testing is more advanced, and in the case of INTER-HEALTH, SAT has already been performed and it has been deployed in the field and the pilot has already been its exploitation because due to the requirements of the experimentation. Regarding INTER-LogP FAT has been performed and SAT is under evaluation.

Third parties have their own schedule and as indicated in the individual evaluation, all of them are executing FAT, and will be finished before March in order to run SAT and after that the evaluation of their use cases in the field as indicated by WP7 methodology.

Section 4 of this deliverable included the outcome of the midterm evaluation of the open call third parties. First evaluation was conducted in June 2017 for the small contributions and in

July 2017 for the large contributions. Preliminary evaluation, set the floor for developments and established for all the contributions the individual and joint exploitation plans. The midterm evaluation covered the period till December 2017. All the contributions presented their advances in terms of integration and Factory Acceptance Testing, both included in D6.1 and this deliverable (D6.2).

Some hints and conclusions related with the evaluation are:

- All the evaluations were satisfactory and the different third parties are advancing as scheduled.
- The integration process is ongoing, preliminary results were presented, however some aspects of the documentation from INTER-IoT have to be polished and improved in order to make the process seamless.
- Use cases for the third parties are now aligned with the requirements of the stakeholders, and clarification and improvements requested in the first review were attended.
- A review of the mentorship over the third parties was revisited and some changes were done in order that the different third parties have an adequate supervision and access to the latest releases.
- A joint session with all the third parties, come out with relevant synergies and some joint evaluation pilots, e.g. U. Twente and Nemergent related to emergency management.
- No Ethical issues were identified, except for U. Twente use case, however the proper anonymization techniques and data management policies proposed by the consortium will be put in place.
- The two large collaborations will integrate two new platforms: OM2M and sensiNact. Both contributions give access to standardization in ETSI where VUB is very active and a link with ECLIPSE in which sensiNact has become a sponsored project.
- Next review will be the final one, in October 2018 and will show the final integration of the open call and the evaluation of the components.