

WSAN4CIP

Deliverable 5.1

Report on system integration

Editor:	José Serrano, TECNATOM
Deliverable nature:	Report (R)
Dissemination level: (Confidentiality)	Public (PU)
Contractual delivery date:	M28
Actual delivery date:	M30
Suggested readers:	Technology suppliers, technology integrators
Version:	1.0
Total number of pages:	55
Keywords:	Integration, testing, prototypes

Abstract

This deliverable describes the activities performed during integration of system prototypes for EDP and FWA demonstrators. Both systems contain multiple components developed by different tasks and partners in the project. Tasks 5.1 and 5.2 have been devoted to specifying and implementing the integration process of these components into the two lab prototypes. This report explains this work. The resulting prototypes do represent deliverable D5.2.

Disclaimer

This document contains material, which is the copyright of certain WSAN4CIP consortium parties, and may not be reproduced or copied without permission.

All WSAN4CIP consortium parties have agreed to full publication of this document.

The commercial use of any information contained in this document may require a license from the proprietor of that information.

Neither the WSAN4CIP consortium as a whole, nor a certain party of the WSAN4CIP consortium warrant that the information contained in this document is capable of use, or that use of the information is free from risk, and accept no liability for loss or damage suffered by any person using this information.

Impressum

[Full project title] Wireless Sensor Networks for the Protection of Critical Infrastructures

[Short project title] WSAN4CIP

[Number and title of work-package] WP5. Integration.

[Document title] Deliverable D5.1 Report on System Integration

[Editor: Name, company] José Serrano, Tecnatom.

[Work-package leader: Name, company] Augusto Casaca, INOV

[Estimation of PM spent on the Deliverable] 10 p-m

Copyright notice

© 2011 Participants in project WSAN4CIP

Executive summary

This deliverable contains a report on the integration tasks performed in work-package 5 of the WSAN4CIP project. The aim of the project is to advance research in the protection of critical infrastructure through the use of wireless sensor and actuator networks (WSAN). In previous work-packages, the project has specified some scenarios for demonstrations of the possible application of WSAN technology to protection of critical infrastructure. Two demonstrators are designed in the project and specified in work package 1:

- ▲ An electrical substation near Lisbon (Portugal)
- ▲ A water treatment station near Frankfurt-Oder (Germany)

All partners of the project have been involved in the development of different technological components that will be tested in the two demonstrators mentioned before. These technological components have been developed during the project in work packages 2, 3, and 4.

Work package 5 is in charge of integration and is responsible for testing the components individually (unit tests) and in an integrated system (system tests) before its implantation in the final demonstrators. All components have been tested individually in the lab by their authors.

Each demonstrator has a partner that is responsible for coordinating the activities. This partner is also leading the test of all the components of their demonstrator.

These integration activities (both unit tests and system tests) are described in this deliverable. Test cards have been defined for each component and demonstrator. This document contains the test cards as well as the results of the tests. The results show that all tested components behave as expected in the lab and can be implemented in the real demonstrators.

List of authors

Company	Author
TECNATOM	José Serrano
INOV	Helena Sarmiento, António Grilo, José Gonçalves
BME	Tamás Holczer
NEC	Jens Matthias Bohli
UMA	Rafael Rodríguez, Jaime Chen, Daniel Garrido
IHP	Krzysztof Piotrowski
SRX	Marcel Selhorst

Table of Contents

1	Introduction	11
2	Overview of EDP Prototype.....	12
2.1	Hardware - Power Line Current Active Monitor	13
2.1.1	Current transformer terminology and definitions	14
2.1.2	Implementation of current sensing WSN node	14
2.1.2.1	Specifications	14
2.1.2.2	Current transformer	15
2.1.2.3	Sensor Conditioning Electronics	15
2.1.2.4	Packaging	17
2.1.3	Tests and Results	17
2.1.3.1	Accuracy Tests	17
2.1.3.2	High Voltage Tests.....	18
2.2	Hardware – Circuit Breaker Trip Coil Active Monitor.....	19
2.2.1	Implementation of Trip Coil WSN Node.....	19
2.2.1.1	Magnetic Sensor	19
2.2.1.2	Sensor Conditioning Electronics	19
2.2.1.3	Trip Coil Actuator Circuit	20
2.3	Hardware – Temperature Active Monitor.....	20
2.3.1	Implementation of Temperature WSN Node	20
2.3.1.1	Temperature Sensor.....	21
2.4	Hardware – Video Surveillance, Intrusion and Hotspot Detector.....	21
2.4.1	Implementation of video surveillance, intrusion and hotspot detector WSN Node.....	21
2.5	Software	22
2.5.1	Description of GUI	22
2.5.2	Description of Gateway SCADA – WSN.....	25
2.5.3	Description of WSN Node.....	26
2.5.3.1	Trip-coil WSN Node	26
2.5.3.2	Temperature WSN Node.....	27
2.5.3.3	Power-line WSN Node.....	27
2.5.3.4	Power Transformer WSN Node	28
2.5.4	Description of Secure micro kernel	29
2.5.4.1	The TURAYA Security Software Layer	31
2.5.4.2	TURAYA’s Hypervisor Layer	31
2.5.4.3	TURAYA trusted sensor node used inside WSAN4CIP.....	32
2.5.5	Description of DTSN.....	33
2.5.5.1	Subset of specification to be implemented	34
2.5.5.2	Software architecture.....	34
2.5.5.3	Interfaces provided to the applications and the routing layer.....	35
2.5.6	Description of Secure Routing.....	35
3	FWA prototype.....	36
3.1	Description of GUI	36
3.2	Description of Tiny DSM	37
3.2.1	The tinyDSM goals.....	38
3.2.2	The architecture and interfaces	39
3.2.3	The tinyDSM data sharing concept	42
3.3	Description of Secure Routing.....	46
3.4	Description of Secure Code Update.....	46
4	Specification of system tests	47
4.1	Test and result cards for EDP demonstrator	47
4.2	Test and result cards for FWA demonstrator	47
5	Integration plans.....	48
5.1	Integration activities for EDP prototype	48
5.2	Integration activities for FWA prototype.....	48
5.2.1	Description of the integration activities of the in lab prototype	49

6	Lessons learned	52
7	Conclusions	53

List of figures

Figure 1 - EDP demonstrator deployed at the MV/LV electricity distribution infrastructure.....	12
Figure 2 – INOV in-lab integration environment.....	13
Figure 3 – Wireless sensing node to measure current in scenario #5.....	14
Figure 4 – Electronic circuit for sensing and energy harvesting modes.....	16
Figure 5 – Packaging of current sensing wireless node.....	17
Figure 6 – Accuracy analysis	17
Figure 7 – Labelec environment to test the wireless current sensing node	18
Figure 8 – Accuracy analysis for high voltage	18
Figure 9 - KMZ10C pins	19
Figure 10 – Magnetic sensor’s conditioning electronics	20
Figure 11 – Trip coil actuator circuit.....	20
Figure 12 – Trip coil.....	20
Figure 13 – Temperature sensor	21
Figure 14 – Video surveillance, intrusion and hotspot detector WSN Node block diagram.....	22
Figure 15 – HV Screen view	22
Figure 16 – Detail of allowed actions on HV/MV power transformer.....	23
Figure 17 – Medium Voltage screen	23
Figure 18- Medium voltage towers screen	24
Figure 19 LV screen.....	24
Figure 20 Network monitoring screen.....	25
Figure 21 Configuration screen	25
Figure 22 – Trip-coil WSN Node.....	26
Figure 23 – Temperature WSN Node.....	27
Figure 24 – Power-line WSN Node.....	28
Figure 25 – Power Transformer WSN Node.....	29
Figure 26. Sensor nodes running the secure microkernel at the EDP substation.	30
Figure 27 - TURAYA-based Trusted Computing Architecture	31
Figure 28 - Components of the TURAYA Security Layer.....	32
Figure 31 - TURAYA software architecture used inside WSAN4CIP	33
Figure 32. DTSN software architecture.....	34
Figure 33 – FWA demonstrator main screen showing some random data values.....	37
Figure 34 – Text Stack (left) and two data boxes (right).....	37
Figure 35: The architecture of a system based on the tinyDSM middleware.....	40
Figure 36: The Operating System adaptation layer – its location and interfaces in a tinyDSM based system	41
Figure 37: The tinyDSM interfaces.....	41
Figure 38: The two-step compilation approach.....	42
Figure 39: The structure of an instance of a variable	43
Figure 40: Addressing of the data in the tinyDSM middleware.....	43
Figure 41: The data replication.....	44
Figure 42: The on-node system architecture of the FWA demonstrator	49
Figure 43: The test network used in the integration phase	50

List of tables

Table 1 – CTS80-1 accuracy	15
Table 2 – Measurement of sensing current.....	18
Table 3 – Integration plan for the EDP prototype.....	48
Table 4 – Integration plan for the FWA demonstrator.....	48

Abbreviations

SCADA: Supervisory Control and Data Acquisition.

WSAN: Wireless Sensor and Actuator Network

Definitions

No special definitions are contained in this deliverable

1 Introduction

This report describes the integration activities performed in the work-package 5 (WP5). The goal of this WP is to “integrate the mechanisms and techniques developed in WP2, WP3 and WP4 in a WSAN deployed in the laboratory, whose configurations will be similar to the ones to be deployed in the field trial environments in WP7”.

WSAN4CIP project follows a schema where WP1 defined two real-life demonstrators for WSAN applied to protection of Critical Infrastructures. Components are developed independently in work packages 2, 3, and 4. These components are integrated and tested in lab prototypes in WP5. After integration and test, WP7 will be validated in the two demonstrators defined in WP1.

The first demonstrator is related to the protection and monitoring of an electrical substation. It is led by consortium partners INOV and EDP. The second demonstrator is in the domain of water distribution and is led by consortium partners FWA and IHP.

In section 2 and 3, both lab prototypes are described. Sections 4 and 5 deal with tests that have been specified and performed in the lab prototypes. Section 6 shows the integration process, with descriptions of the activities and milestones. Section 7 shows lessons learned during the integration process along with recommendations for set-up of demonstrators. Lastly, section 8 contains the conclusions.

2 Overview of EDP Prototype

The electrical energy distribution infrastructure is a critical infrastructure that requires protection for safety and security reasons. The energy distribution infrastructure mainly consists of a set of substations, Medium Voltage (MV)/ Low Voltage (LV) power transformers outside the substation, MV power lines connecting substations to MV/LV power transformers and LV power lines from the MV/LV power transformers to the customers. Some industrial customers may also get direct MV power lines. Associated to this infrastructure we should also consider the SCADA system, which is a supervisory control and data acquisition system for the infrastructure. This infrastructure is illustrated in Figure 1.

For safety reasons remote surveillance of the electrical energy distribution network is already established to some extent based on wired sensors. The use of wireless sensor and actuator networks (WSAN) can lead to a more powerful and efficient protection scenario for the substations, power lines and power transformers. The higher deployment flexibility allows wireless sensors to capture more status parameters than the existing fixed sensor infrastructure. Specific actuators can also be included in the infrastructure as part of the WSAN. The wireless nature of communication can also contribute to avoid critical points of failure.

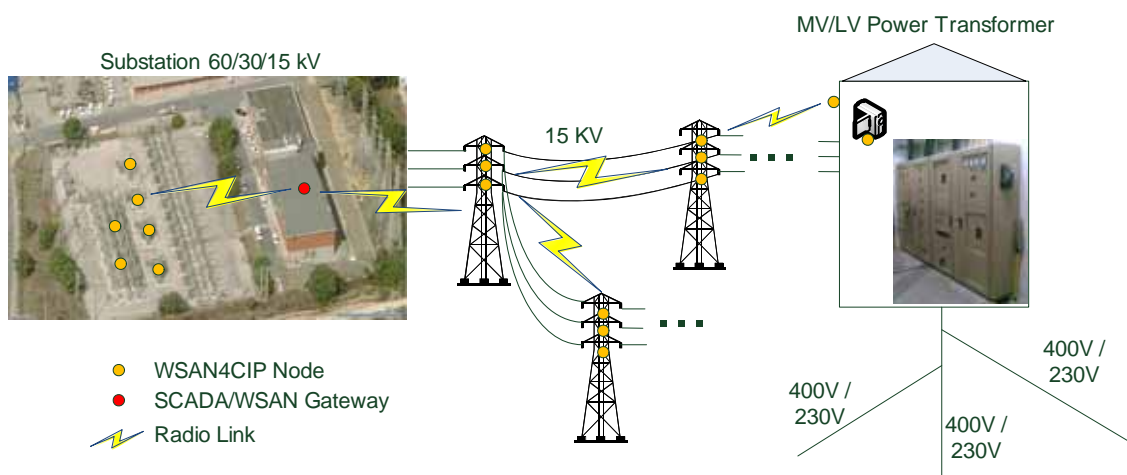


Figure 1 - EDP demonstrator deployed at the MV/LV electricity distribution infrastructure.

From the safety and security analysis made in the project we have concluded that based on the WSAN we can provide a set of solutions capable of improving the protection of this infrastructure. From the safety point of view, some remote monitoring processes have been identified to be deployed and from the security point of view, the use of cameras associated with WSAN can significantly improve the robustness of the solutions for the physical protection of substations and MV/LV power transformer installations.

In the safety area we have defined solutions for the remote active monitoring of: i) substation circuit breaker trip coil status; ii) substation power transformer oil temperature; iii) substation neutral reactance oil temperature; iv) substation neutral resistor coil box temperature; v) MV power line activity; vi) MV/LV power transformer hotspot detection. All the monitored parameters will be visualized at the SCADA system, through a special-purpose interface and a graphical user interface. In the security area, a combination of motion detector and video camera are deployed and integrated with the WSAN for improving the physical protection of the MV/LV power transformer installations.

The energy consumption of this WSAN node cannot be overlooked. Although the power distribution network carries energy by definition, its use as a WSAN power source presents many challenges. This is especially true in the MV power lines outside the substations, where the voltages are too high (e.g. 15 kV) to be used directly by the WSAN nodes. An energy harvesting solution was developed to recharge the batteries of those WSAN sensor nodes from the current that passes on the 15 kV line.

The difficulties related with integration and debugging of the deployed system prompted the deployment of an in-lab integration environment at the INOV premises, depicted in Figure 2. Here, all the scenarios are replicated, allowing the in-lab integration of the complete system before deployment.

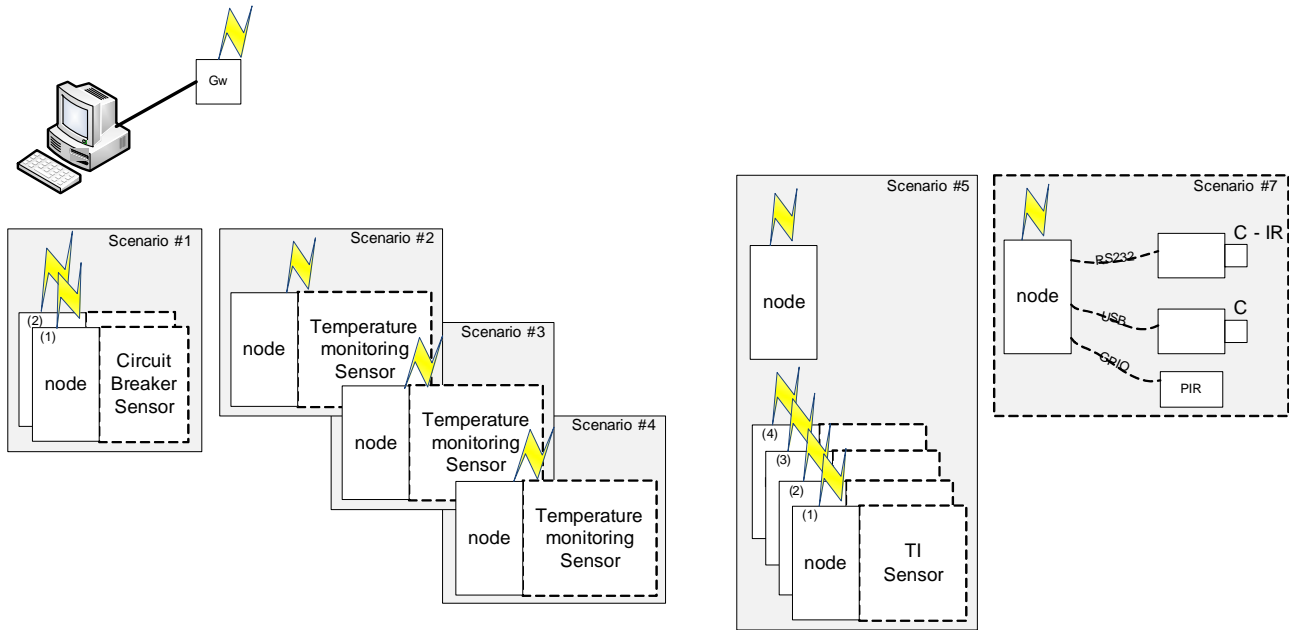


Figure 2 – INOV in-lab integration environment.

Technical intervention at the MV tower sensor nodes requires a specialized team and the MV line to be shutdown, affecting a significant number of clients, being thus very expensive. Consequently, once deployed, the MV sensor nodes cannot be easily retrieved to correct any faults that might be detected afterwards.

For this reason, the consortium decided to implement a more flexible backup solution using the LV (230 V) power lines located between the substation and the MV/LV power transformer. These LV sensor nodes are almost identical to the MV sensor nodes, comprising an identical PCB, power transformer, casing, software, etc. Although the LV sensor nodes could also use the same energy harvesting mechanism employed in MV, the lower currents passing through the LV lines make it more efficient to feed the LV nodes directly from the 230 V.

Since node retrieval and deployment at the LV lines is easier and less expensive (e.g., it can be performed without switching the power lines off, or, even if shutdown is required, it affects fewer EDP clients), the LV sensor nodes shall be deployed first. After the latter are deployed and tested, the team shall proceed with MV deployment. It should also be noted that EDP has also manifested its practical interest on the addition of LV current measurement since there is currently no alternative way to do it.

2.1 Hardware - Power Line Current Active Monitor

This scenario aims to monitor the status of a medium-voltage (MV) 15 kV aerial power line section (Figure 3). The electrical current flowing through each line (phase) is sensed and measured by a WSN node, using a current transformer (CT). The current to charge the battery, which powers up the WSN node, is also obtained from the MV line, using the same current CT. Therefore, two working modes are defined for this WSN node: current sensing and energy harvesting. Modes are controlled by the microcontroller of the wireless module (Silex SX-560 module) [2].

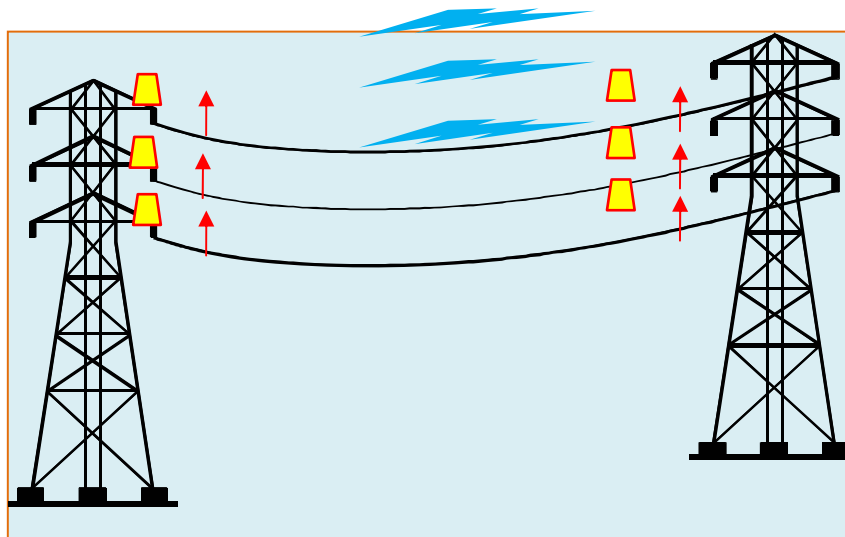


Figure 3 – Wireless sensing node to measure current in scenario #5

2.1.1 Current transformer terminology and definitions

A CT is an instrument transformer intended to have its primary winding connected in series with the conductor carrying the current to be measured or controlled. A large current in the primary winding is reduced to a lower level to be more easily measured.

The CT includes a primary winding, a magnetic core and a secondary winding. In a window type CT [3], the secondary winding is insulated from and permanently assembled on the core, but no primary winding exists as an integral part of the structure. Primary insulation is provided in the window, through which one turn of the line conductor can be passed to provide the primary winding. The number of secondary turns determines the amplitude of the output current.

A CT can have a solid or split core. Solid core CTs features a closed loop. A split-core CT can be temporarily opened to be placed around existing wires, without disconnecting the primary circuit.

The performance of a CT involves the specification of:

- The rated current (I_p): the primary current upon which the performance specifications are based [3].
- The rated secondary current (I_s): the primary rated current divided by the marked (nominal) ratio ($I_p:I_s$) stated on the CT [3].
- The accuracy: the percent difference between the actual secondary current and the rated secondary current [3]. For example, an accuracy of 0.3 certifies the CT is accurate to within 0.3 percent of its rated ratio value for a primary current of 100 percent of rated ratio. A CT with a rated ratio of 400:5 with accuracy class of 0.3, when operating with 100 percent of rated ratio (400 A), produces a secondary current between 4.9850 A and 5.0150 A.
- The burden: the electrical impedance of the secondary current loop [3] [4]. It determines the active and reactive power at the secondary terminals. The burden is expressed either as total ohms impedance or as the total volt-amperes and power factor of the secondary devices and leads. The VA burden of a CT is determined by the resistivity of circuit on the secondary, multiplied by the current squared through the secondary.

2.1.2 Implementation of current sensing WSN node

2.1.2.1 Specifications

Current sensing specifications are:

- Minimal current in the line: 5A.

- Maximal current values: 300 A.
- Maximal duration of short circuit current duration (before actuation of protections): 3 s
- Accuracy in current measurements shall be up to 10%.
- Line current sampling frequency controlled by SCADA.

As the wireless sensor node is powered from the MV line, other specifications include:

- Current to charge the battery shall be less than 1 A.
- Only positive power supplies are generated for the WSN node.

2.1.2.2 Current transformer

We chose a split-core CT, the Hobut CTS80-1, with a ratio of 400/5 A [6]. It supports a primary current up to 400A. This CT is suitable for three classes of accuracy (Table 1) 0.5% up to 2.5 VA; 1% up to 6 VA; and 3% up to 10 VA, being adequate to the specifications for current measurement accuracy (10%).

Table 1 – CTS80-1 accuracy

I _{prim}	VA (burden)		
	class 0.5	class 1	Class 3
100			1.5
150			2
200		1.5	2.5
250		2	4
300	1.5	4	6
400	2.5	6	10

For 100 A, the accuracy is 3% for a burden of 1.5 VA (Table 1). No values are specified for lower currents. Therefore, for an accuracy of 3%, the burden resistor cannot exceed 0.96 Ω (equations 1 and 2) [5].

$$1.5 = I_{\text{sec}} \times V_{\text{sec}} = I_{\text{sec}} \times (R_{\text{burden}} \times I_{\text{sec}}) = \left(\frac{100}{80}\right)^2 \times R_{\text{burden}} = 1.5625 \times R_{\text{burden}} \quad (1)$$

$$R_{\text{burden}} = \frac{1.5}{1.5625} = 0.96 \Omega \quad (2)$$

For higher currents and the same accuracy, burden resistor can be higher. We use in our circuit 0.82 Ω¹.

2.1.2.3 Sensor Conditioning Electronics

Figure 4 presents the schematic diagram of the main circuit. Additional circuitry is presented in the Annex. In the sensing mode, the secondary current is measured based on the voltage at the terminals of the 0.82 Ω resistor (R1). The harvesting mode uses secondary current to charge the battery.

¹ The drain to source resistance of transistor RFP30N06LE is 0.047 Ω << 0.82 Ω

2.1.2.4 Packaging

The electronic circuit is enclosed in an aluminium alloy box with a cubic shape (Figure 5). The aluminium box is completely covered with a Scotch™ 23 Electrical tape, a highly conformable Ethylene Propylene Rubber (EPR) based, high voltage splicing tape. It is a non-vulcanising, shelf-stable tape with excellent electrical properties. 23 Tape can be used as insulation for low-voltage applications as well as insulation for splices up to 69kV.



Figure 5 – Packaging of current sensing wireless node

To guarantee high resistance to outdoor weather conditions, namely to humidity, rain and UV rays, the box is then painted with liquid silicon. This node will be suspended on 15 kV aerial power line (see Figure 5).

2.1.3 Tests and Results

2.1.3.1 Accuracy Tests

For testing purposes, we inject a current in a cable (emulating the line), which represents the primary of the current transformer, and use the electronic circuit (Figure 4) to measure the secondary current. In fact, we measured the voltage at the terminals of the 0.82 Ω resistor, using an ADC (AD1 in Figure 4). The Silex SX-560 microprocessor computes the current.

Results are presented on Figure 6. The green line presents current values measured at the primary winding (the line) with a commercial current transformer, class 3. Values obtained by the developed sensing node are presented by the blue line. Values are calculated by software based on the voltage value measured by the ADC. Maximum error of 14% is obtained for a current of Accuracy is less than ±2% of FSO (Full Scale Output).

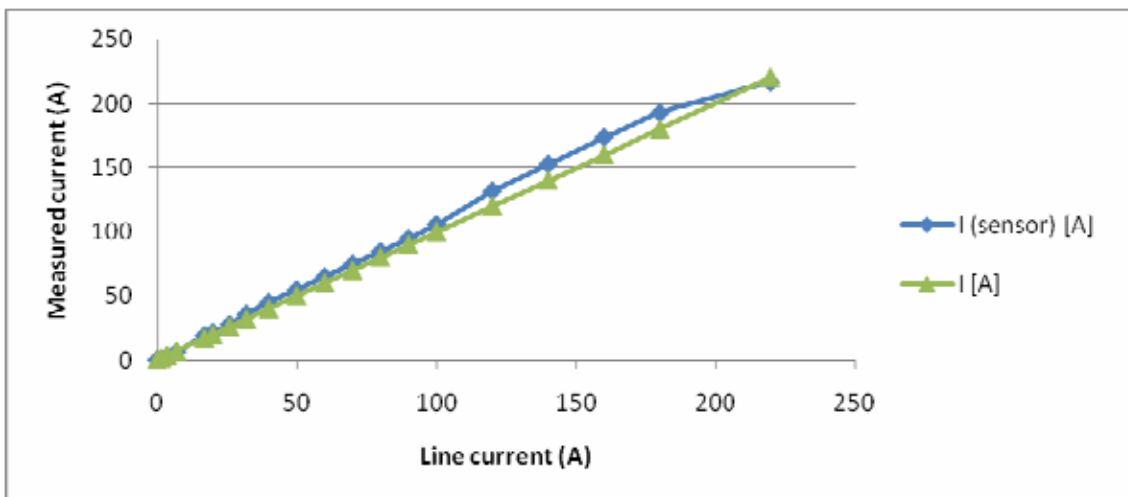


Figure 6 – Accuracy analysis

2.1.3.2 High Voltage Tests

Power lines generate electric and magnetic fields. In order to find potential problems, the wireless current sensing node was tested exposed to the electromagnetic fields of a 15 kV line. These tests were performed for the first version of the electronic circuit. Figure 7 presents the test environment at Labelec².



Figure 7 – Labelec environment to test the wireless current sensing node

Firstly, the node was exposed to electric fields originated by voltages of 9kV, 13 kV 20 kV. No problems were detected on the wireless communication nor on the node functionality. The second set of tests, include the measurement of sensing current with a 15kV voltage (Table 2). Finally, we could notice the corona effect 0 at 20 kV.

Table 2 – Measurement of sensing current

Line current (A) ³	Measured current (A)
25	37.3
50	64.5
100	116
150	165

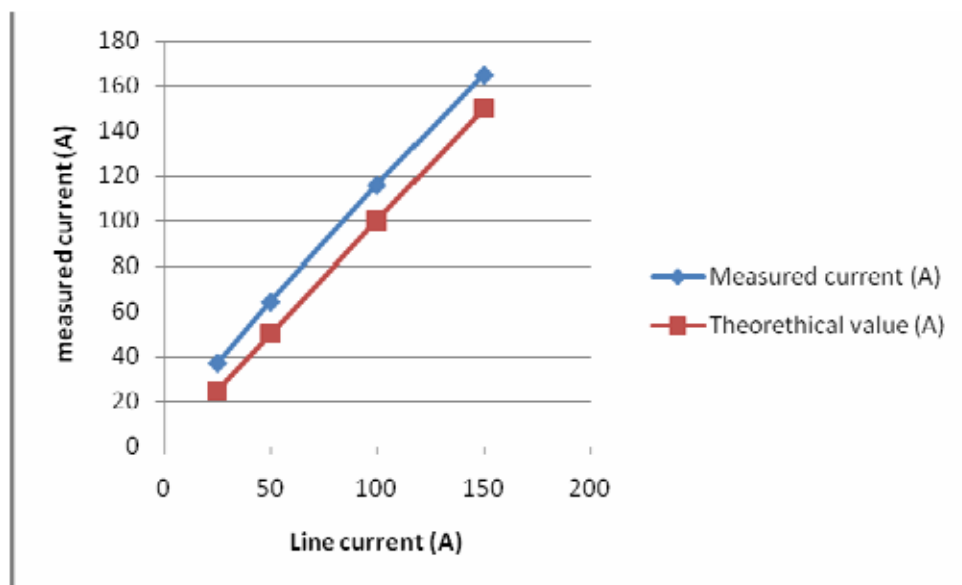


Figure 8 – Accuracy analysis for high voltage

The objective of these experiments is to expose the electronic circuit to electromagnetic fields of a 15 kV line. However, we also analysed the accuracy of measurements. We expect better accuracy with the second

² Labelec is high voltage testing laboratory belonging to the Electricity of Portugal (EDP) group.

³ Labelec referred error up to 2% in this value

version of the electronic circuit, but due to the expensive costs, the circuit will not be tested again in a high voltage environment. Table 2 and Figure 8 present the obtained values. Measured values present an offset to the theoretical values. This difference can be corrected by SW.

2.2 Hardware – Circuit Breaker Trip Coil Active Monitor

The trip coil is a component of a circuit breaker that permits to interrupt the electrical current flow to the power line. The circuit breaker is actuated by applying a 110V DC voltage to the trip coil terminals. Even in normal operation, the coil can be damaged. In scenario #1, the trip coil is actuated to monitor its status.

For monitoring purposes, a low DC current is sent through to the trip coil. If the current flows, the coil is working properly. To verify the existence of a current flow, the magnetic field is sensed. We selected a magneto-resistive sensor, where the resistivity of a material changes due to a magnetic field.

2.2.1 Implementation of Trip Coil WSAN Node

This section describes the implementation of each component of the trip coil WSAN node.

2.2.1.1 Magnetic Sensor

The selected sensor, the KMZ10C, is a magnetic field sensor, employing the magnetoresistive effect of thin-film permalloy. When the current passes through the material, the internal magnetization vector is parallel to the current flow. When an external magnetic field is applied, the current direction changes by an angle that depends on the strength of the magnetic field. There is a resistivity change due to a magnetic field that depends on the angle formed by the internal magnetization vector and the direction of the current flow. Resistance is largest if the current flow and the internal magnetization vectors are parallel. Four sensors, connected in a Wheatstone bridge configuration, form the complete sensor (Figure 9).

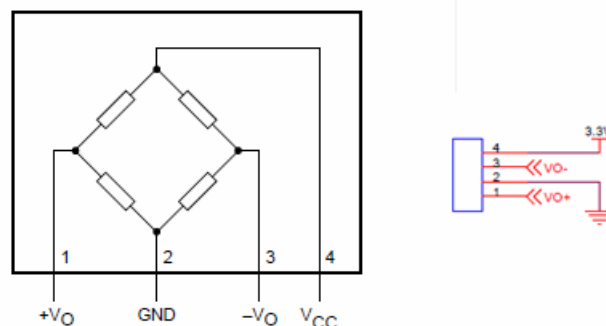


Figure 9 - KMZ10C pins

2.2.1.2 Sensor Conditioning Electronics

Figure 10 presents the schematic diagram of the electronic circuit to detect the magnetic field. Voltage between V_{o-} and V_{o+} is the sensor output (Figure 10). The MAX4208 is an instrumentation amplifier, providing an adjustable gain with two external resistors (R_8 and R_9+R_{10}). The analog amplified voltage is converted to a digital value by the ADC121S021, a successive-approximation ADC with an internal sample and hold circuit. The digital signal is connected to a Silex SX-560 Module by an SPI serial data link.

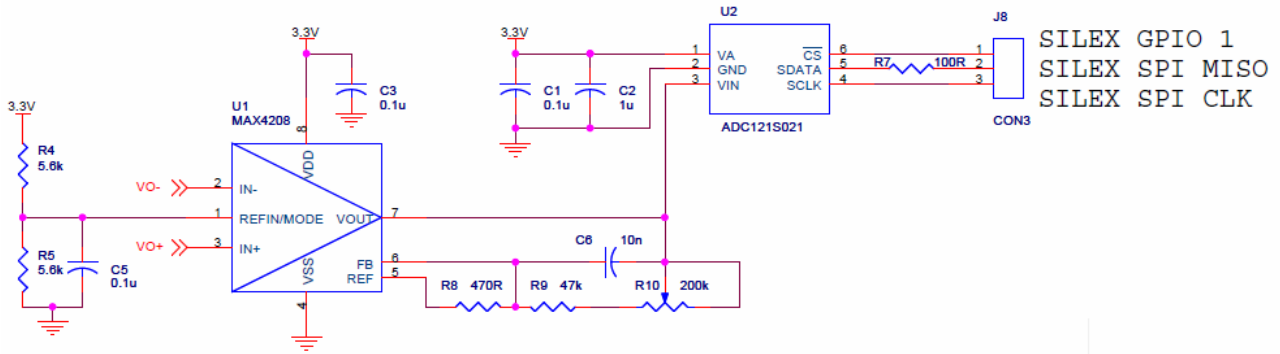


Figure 10 – Magnetic sensor’s conditioning electronics

2.2.1.3 Trip Coil Actuator Circuit

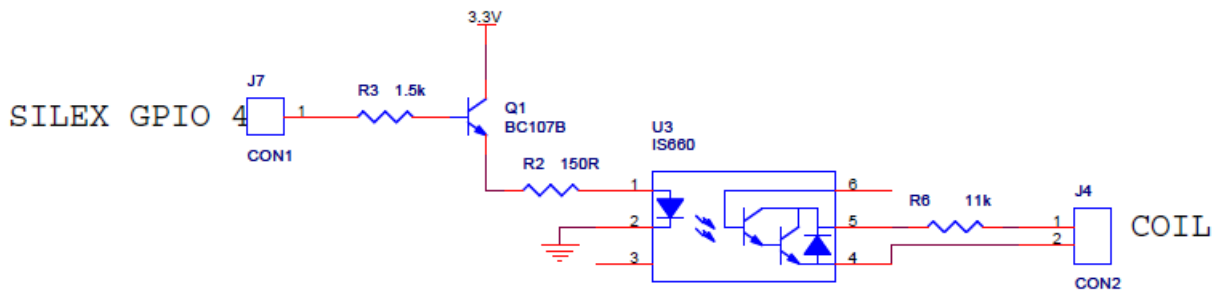


Figure 11 – Trip coil actuator circuit

The circuit of Figure 11 is used to monitor the trip coil, injecting a current through it. The 110 V DC voltage can be simultaneously applied to the coil terminals, as presented on Figure 12. We included an optocoupler (IS660 on Figure 11) to provide electrical isolation.

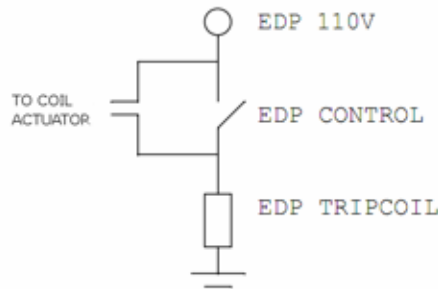


Figure 12 – Trip coil

2.3 Hardware – Temperature Active Monitor

Failures on power transformers affect a series of power lines and all the downstream MV/LV power stations, meaning thousands of homes and businesses. These failures can be monitored, using temperature sensors. Temperature sensors can also detect phase-earth failures on the 15 kV power transformers. In scenario #2 the oil temperature of the power transformers is measured, in scenario #3 the oil temperature of the Neutral Reactance element is monitored to detect phase-earth failures and in scenario #4 de Neutral Resistor Coil box temperature is monitored.

2.3.1 Implementation of Temperature WSN Node

This section describes the implementation of the temperature WSN node.

2.3.1.1 Temperature Sensor

The selected sensor, the LM70, is a silicon temperature sensor. The temperature-sensing element relies on the change of voltage across a p-n junction, essentially a silicon diode. Temperature resolution is 0.25°C while operating over a temperature range of -55°C to +150°C. Depending on the temperature range, accuracy changes from ±2° up to +3.5/-2°C.

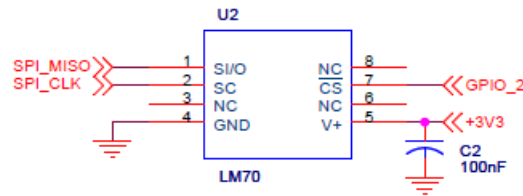


Figure 13 – Temperature sensor

The LM70 is an integrated circuit, including the temperature-sensing element, the signal conditioning electronics, an ADC converter and the SPI serial data link. It connects directly to the Silex SX-560 Module through the SPI link (Figure 13).

2.4 Hardware – Video Surveillance, Intrusion and Hotspot Detector

The MV/LV power transformer scenario aims to improve the physical protection of the MV/LV power transformer installations. This was technologically achieved through a combination of motion detection, video surveillance and hotspot monitoring mechanisms.

2.4.1 Implementation of video surveillance, intrusion and hotspot detector WSN Node

As you can see in Figure 14, this WSN Node is the result of mixing different technologies, all communicating with the Silex SX-560 Module through several types of interfaces, as described above:

- Video surveillance: Logitech's Quickcam Sphere AF Webcam connected through a USB link. To ensure the goal in harsh illumination conditions, it was implemented a small device of illumination composed by a light-emitting diode (LED) device controlled through GPIO;
- Motion detection: UbiSec&Sens passive infrared (PIR) sensor communicating through GPIO;
- Hotspot detector: IRISYS IRI 1011 Thermal Infrared Camera communicating through a RS232 serial data link. Because of the incompatible voltage levels between the camera and the Silex SX-560 Module, a RS232 to CMOS level converter was implemented;

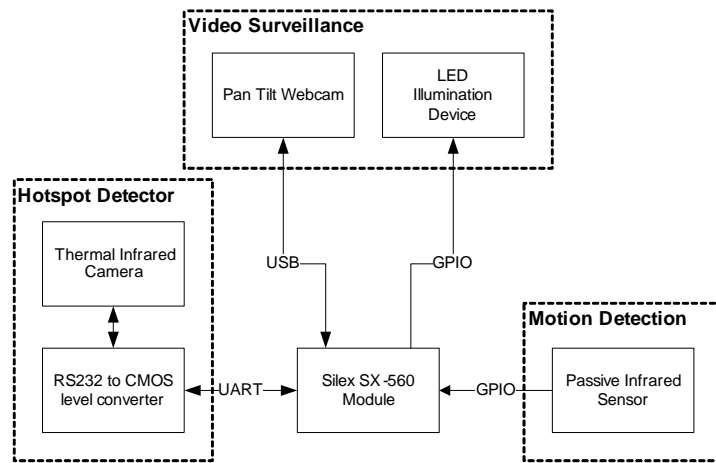


Figure 14 – Video surveillance, intrusion and hotspot detector WSN Node block diagram

2.5 Software

2.5.1 Description of GUI

A Machine-to-Machine (M2) open source system called Mango (<http://mango.serotoninsoftware.com/>) has been used to show the information provided by the wireless sensor network. Mango has been chosen between a list of possible SCADA system because of its active support and for being open source.

By using a web application instead of a desktop application allows the SCADA system to run in a wide variety of architectures and operating systems. Also, it is rather simple to deploy new versions of it when changes are made. All these changes are instantly received by the clients without having to carry out complex administration tasks.

Although Mango allows the so-called public views, in which no login information is required, we have only developed private views. Only logged users may use the system. The appearance and behavior of each item in the screens is modeled as a JavaScript script and CSS in separate files, to allow changes to be made on the presentation without affecting the rest of the system.

A screen has been developed for each of the substation sections and on the power supply distribution. We will present the different screens in the following figures.

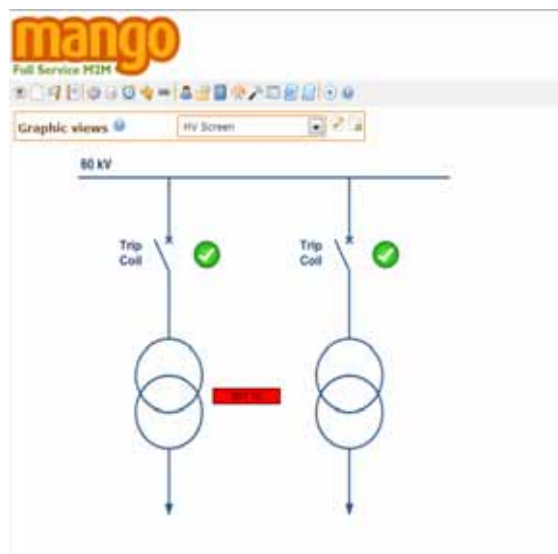


Figure 15 – HV Screen view

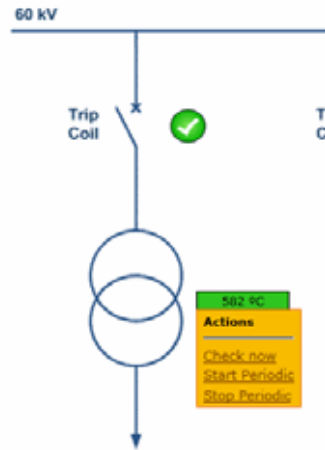


Figure 16 – Detail of allowed actions on HV/MV power transformer

Figure 15 shows the high voltage section where three elements are monitored: two circuit breaker trip-coils, and a HV/MV power transformer. Operations can be executed to request the status of the trip-coils and the temperature of the power transformer.

The actions a user can perform on any element are shown when the mouse is over that element (as depicted in Figure 16). Two main kinds of actions are allowed in the EDP demonstrator: 1) launch on demand operations and 2) launch periodic operations. For periodic operations two different options are offered. The first option activates the periodic request of data and the second one deactivates it.

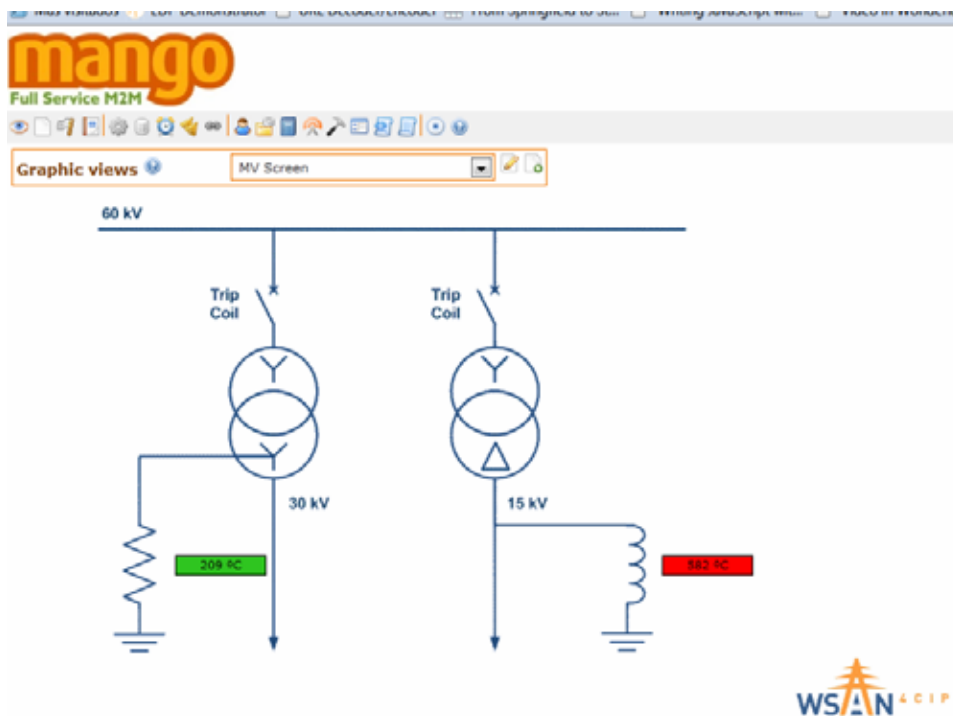


Figure 17 – Medium Voltage screen

Figure 17 shows the medium voltage section, in which two elements are monitored: a neutral resistor and a neutral reactance. The temperature of these two elements can be controlled by sending on-demand or periodic requests to the gateway.



Figure 18- Medium voltage towers screen

Figure 18 shows the low and medium voltage towers monitored between the EDP substation and the MV/LV power station. All sensors in the tower are able to measure current, temperature and battery level (except for the one labeled MV1 which measures current and battery level). Data can only be requested in a periodical way. The status of the different elements is indicated by means of four different background colors. Grey indicates that the element has not been updated for a certain period of time (configurable in the configuration screen). Green color indicates a normal status. Yellow and red indicates a warning and alarm status respectively.



Figure 19 LV screen

LV Screen (Figure 19) depicts the low voltage section, in which the MV/LV power station is monitored through video vigilance and hot-spot detection. Video (left side of the screen) can be launched on demand. Also it will automatically be shown if intrusion detection is activated and an alarm notification is received by Mango.

In the same way a thermal streaming video (right side of the screen) is shown if hotspot detection is activated and a hotspot alarm is received. Also, the thermal streaming video can be activated on demand.

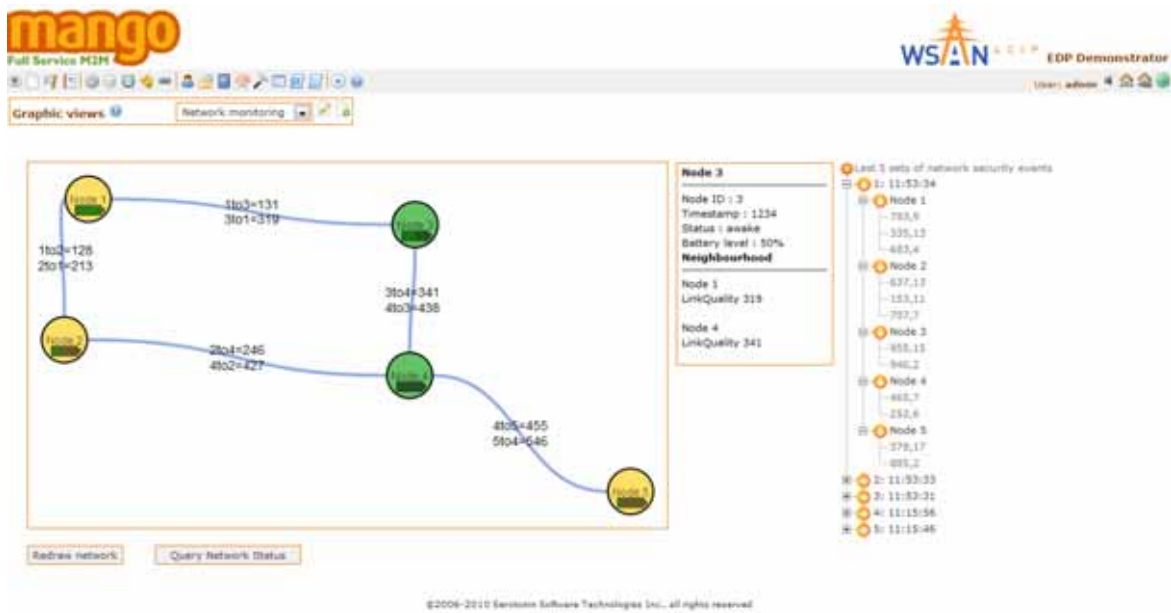


Figure 20 Network monitoring screen

The network monitoring screen (Figure 20) shows a graphical representation of the wireless sensor network and the link between the different nodes. Information about the state of the network, such as node battery level, link quality, node status, etc, is shown when a node is selected with the mouse. Also, the last set of security events is shown in the screen. All the information shown in this screen is periodically requested to the gateway by means of a web service.

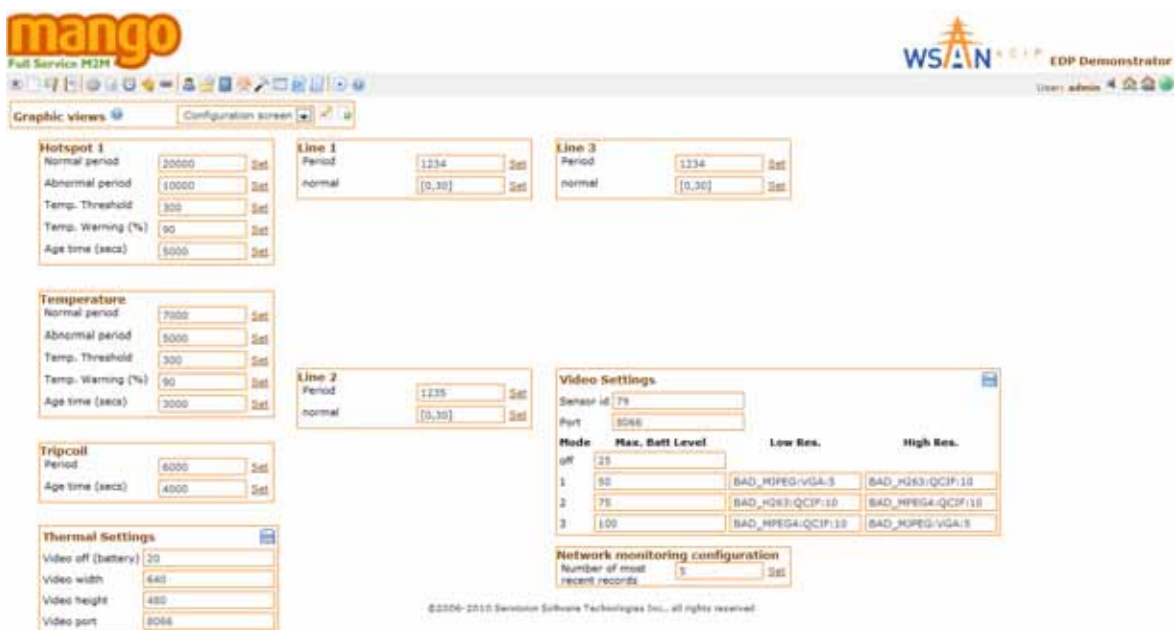


Figure 21 Configuration screen

The configuration screen is shown in Figure 21. All the different configurable parameters from the other screens can be set here. Examples of configurable parameters are: video port, temperature alarm range, period of the different periodical operations, etc.

2.5.2 Description of Gateway SCADA – WSAN

This gateway works as a bridge between the SCADA requests and WSAN responses or events.

The communication between the SCADA and the Gateway is done via web services that may be synchronous requests with communication to WSAN sensors and respective response inside a period of time after which the timeout message is triggered; events as requested by the SCADA (e.g.: temperature readings asynchronously) or events not requested but activated by SCADA triggered by WSAN sensors as a result of a message that must be reported to SCADA. Communication between WSAN to SCADA is also done via UDP packets when video or IR feeds are activated.

The communication between the Gateway and the WSAN sensors are done with text strings separated by semicolons that are null terminated. A message to sensor “ID” may have a prefix IP and the last byte corresponds to the “ID” value. Other fields are parsed at destination depending on which request or response is dealing with. Communication ports are pre-configured. There are also pseudo-broadcast messages transmitted up to a number of times until response for a group of configured sensors that can be applied to the tower current measurement and in this case the most significant bit on “ID” must be set.

Any web method called by the SCADA to the Gateway, initializes (if not done yet) the global service procedures and some other variables based on the request.

There are receive FIFOs and send FIFOs in which the messages are put based on the “ID” of the destination sensor. For example, when a method is invoked by SCADA a formatted text message is put in the send FIFO and dispatched as soon as the global service reaches the send procedure. The receive FIFOs are also filled, in the global service receive procedure, by the “ID” that comes in any message after the response title. It is verified the response title and if it’s an asynchronous message a web method is invoked asynchronously to a specific configured endpoint in the SCADA system, otherwise is a synchronous message and the respective receive FIFO is filled with the message.

2.5.3 Description of WSAN Node

The EDP WSAN Nodes can be of four possible types: Trip-coil WSAN node, Temperature WSAN Node, Power-line WSAN Node, and Power Transformer WSAN node. The first two shall be deployed at the Substation and the other two on the power-line towers and MV/LV Power Transformer, respectively. Each type of EDP WSAN Node is described in a separate subsection. As shall be seen, these different software architectures share a set of core modules and usually differ only on the interfaces required to deal with the specific sensor suites.

2.5.3.1 Trip-coil WSAN Node

The Trip-coil WSAN Node is depicted in Figure 22, which distinguishes between modules running at the application level and the Linux kernel.

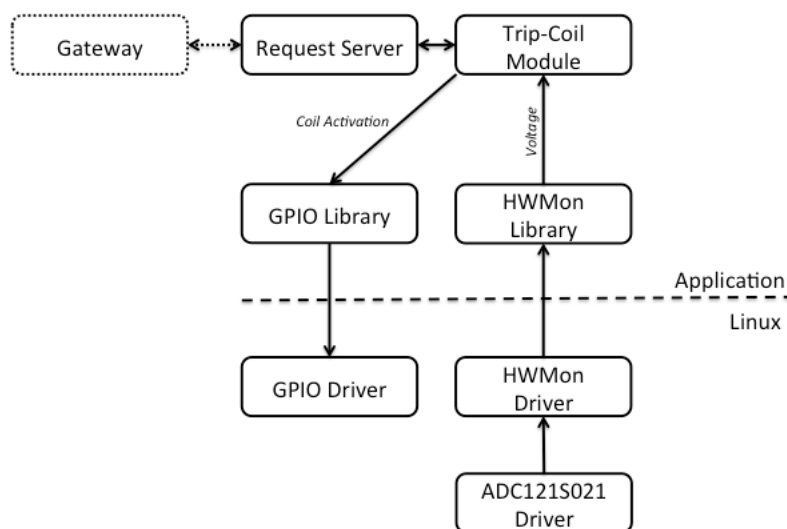


Figure 22 – Trip-coil WSAN Node.

This WSAN node comprises the following modules:

- Trip-Coil module: Implements all the control logic of the WSAN node, collecting, processing and transmitting measurements, as well as controlling the trip-coil actuator. Handling of communications to/from the WSAN Gateway is done through the Request Server module.
- Request Server: Manages all communications with the WSAN Gateway, passing all requests to the Trip-Coil module.
- GPIO Library: Library of functions to allow interaction with the GPIO interface of the SX-560, allowing the activation of the actuator circuit.
- GPIO Driver: Device driver support for the GPIO interface of the SX-560.
- HWMon Library: Library of functions to interact with the Linux HWMon Driver.
- HWMon Driver: High-level device driver that allows reading from the ADC121S021. This corresponds to the digitized voltage reading from the KMZ10C magnetic field sensor.
- ADC121S021 Driver: Low-level device driver for the ADC121S021.

2.5.3.2 Temperature WSAN Node

The Temperature WSAN node is depicted in Figure 23, which distinguishes between modules running at the application level and the Linux kernel.

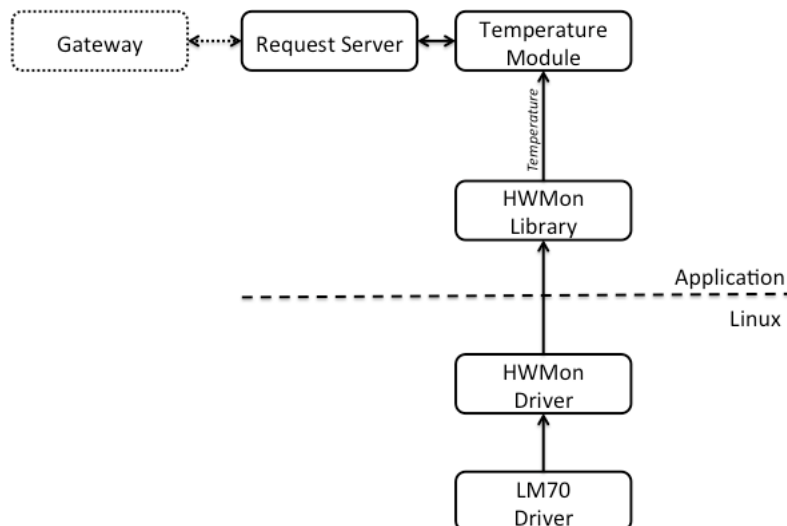


Figure 23 – Temperature WSAN Node.

This WSAN node comprises the following modules:

- Temperature module: Implements all the control logic of the WSAN node, collecting, processing and transmitting measurements. Handling of communication to/from the WSAN Gateway is done through the Request Server module.
- Request Server: Manages all communications with the WSAN Gateway, passing all requests to the Temperature module.
- HWMon Library: Library of functions to interact with the Linux HWMon Driver.
- HWMon Driver: High-level device driver that allows reading from the LM70 temperature sensor.
- LM70 Driver: Low-level device driver for the LM70 temperature sensor, which is read through the SPI interface.

2.5.3.3 Power-line WSAN Node

The Power-line WSAN Node is depicted in Figure 24, which distinguishes between modules running at the application level and the LINUX kernel.

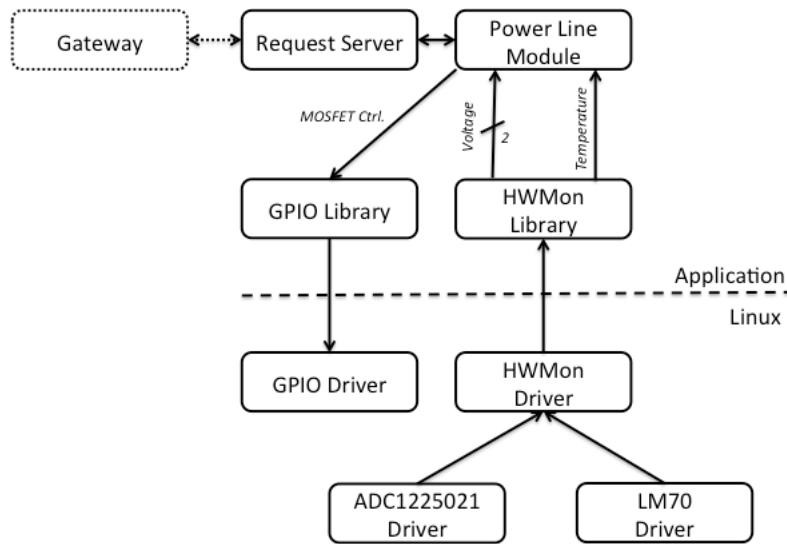


Figure 24 – Power-line WSAN Node.

This WSAN node comprises the following modules:

- Power-line module: Implements all the control logic of the WSAN node, collecting, processing and transmitting measurements. Handling of communication to/from the WSAN Gateway is done through the Request Server module.
- Request Server: Manages all communications with the WSAN Gateway, passing all requests to the Power-line module.
- GPIO Library: Library of functions to allow interaction with the GPIO interface of the SX-560, allowing the NMOS transistor battery recharging to be switched on/off.
- GPIO Driver: Device driver support for the GPIO interface of the SX-560.
- HWMon Library: Library of functions to interact with the Linux HWMon Driver.
- HWMon Driver: High-level device driver that allows reading from the ADC122S021 and LM70 temperature sensor. From the ADC122S021, two digitized voltage values are read, one corresponding to the battery charge and the other to the line current reading.
- ADC122S021 Driver: Low-level device driver for the ADC122S021.
- LM70 Driver: Low-level device driver for the LM70 temperature sensor, which is read through the SPI interface.

2.5.3.4 Power Transformer WSAN Node

The Power Transformer WSAN Node is depicted in Figure 25, which distinguishes between modules running at the application level and the LINUX kernel.

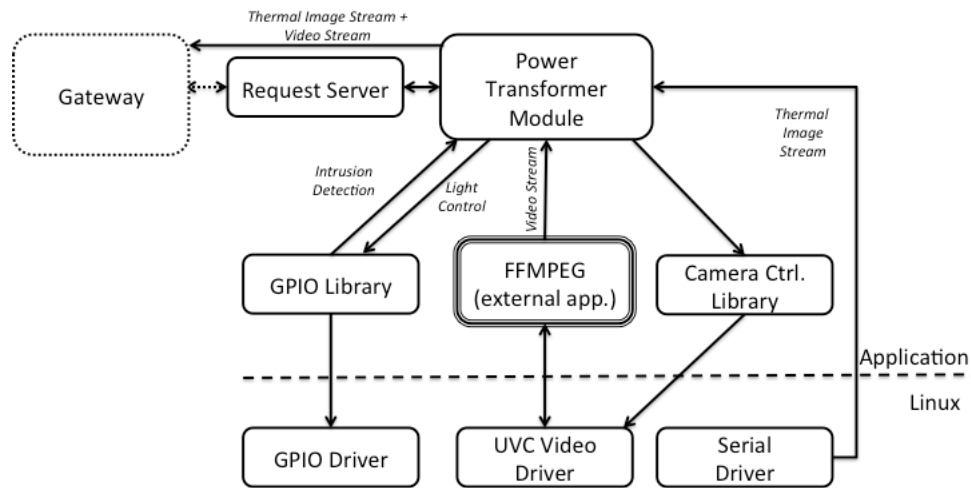


Figure 25 – Power Transformer WSAN Node.

This WSAN node comprises the following modules:

- **Power Transformer module:** Implements all the control logic of the WSAN node, collecting, processing and transmitting surveillance video, thermal video and intrusion alarms, as well as supporting remote PT control of the video surveillance camera and control of the ambient light intensity. Handling of communication to/from the WSAN Gateway is done through the Request Server module, except for the video and thermal image streams, which are sent directly to the Gateway. Regarding the video stream in particular, this module is responsible for translating the video transport from RTP/UDP to DTSN.
- **Request Server:** Manages communications with the WSAN Gateway, passing all requests to the Power-line module.
- **GPIO Library:** Library of functions to allow interaction with the GPIO interface of the SX-560, allowing the reception of intrusion detection alarms from the PIR movement sensor, as well as to control the ambient light intensity.
- **GPIO Driver:** Device driver support for the GPIO interface of the SX-560.
- **FFMPEG:** Third-party application to record, convert and stream audio and video⁴. FFMPEG is free software licensed under the LGPL or GPL.
- **Camera Control Library:** This library allows the PT control of the video surveillance camera.
- **UVC Video Driver:** The USB Video Class Linux device driver supports interaction with compliant video cameras, such as the Logitech's Quickcam Sphere AF Webcam, providing video reception and camera control functionalities.
- **Serial Driver:** The RS232 serial link device driver supports reception of the thermal image stream from the IRISYS IRI 1011 Thermal Infrared Camera.

2.5.4 Description of Secure micro kernel

Existing platforms used inside WSAN cannot provide a secure environment to protect security-critical data and applications against attacks by malicious applications or processes. This problem concerns many different application scenarios in private and business areas, such as smart grid/metering, or even sensor networks for the protection of critical infrastructures.

There is a high interest in protecting sensitive or private data (measured data, e.g., power consumption, alarm systems, other). Each scenario requires the assurance that sensitive information can only be accessed by trustworthy and dependable applications.

⁴ <http://www.ffmpeg.org/>

Enterprises are interested in controlling access to and handling of critical information (e.g., node firmware, cryptographic keys, or measured data) securely, i.e., protecting them from unauthorized usage. Also, employees or users should not be able to circumvent control mechanisms by using available functions for their own purpose, or by exploiting security weaknesses of existing software components.

In general, each party desires to enforce its own security policy on the platform. In the following, we will focus on the protection of specific applications, i.e., the enforcement of a user's security policy. The proposed underlying security architecture provides functionalities that allow secure enforcement of a user's (application's) (local) policies.

The general idea behind this architecture is to establish various compartments on one sensor platform where each compartment can have its own security policy. The policy defines:

- The protection level for the data accessed and processed in a compartment as well as for the applications that run in this compartment, and
- The information flow between individual compartments as well as between the compartments and external parties.

The goal is that each compartment behaves as if it is a single sensor platform separated from other compartments. Furthermore, the underlying architecture should provide channels to the corresponding compartments where the channel properties are specified.

This compartmentalisation allows for instance to separate communication mechanisms (WLAN, ZigBee) with connection to an external network (and thus the possible attack points) from security critical compartments, such as used for encryption or measuring data.

Figure below shows the sensor incorporating the secure microkernel after being deployed at EDP substation.



Figure 26. Sensor nodes running the secure microkernel at the EDP substation.

2.5.4.1 The TURAYA Security Software Layer

TURAYA is a security architecture providing strong isolation and multilaterally secure policy enforcement of legacy applications. As illustrated in Figure 27 - TURAYA-based Trusted Computing Architecture, a TURAYA-based security architecture consists of the following three layers:

- a hardware layer, including conventional components such as memory, CPU, devices, and a Trusted Platform Module (TPM). Due to the fact that current sensor platforms are not equipped with TPMs, a Mobile Trusted Module (MTM) can be used. Possible hardware security anchors for sensor nodes, which can be used with TURAYA, have been analyzed in WSAN4CIP deliverable D2.3,
- the TURAYA security kernel, including a hypervisor layer and a trusted software layer, and
- applications and legacy operating systems that are executed in parallel and isolated from each other with process and memory isolation.

An exemplary system using this general architecture could run 2 different Linux operating systems (L4Linux), one connection to other networks (Secure VPN), and another compartment responsible for additional rules-management (Management).

Breaking down this architecture to the area of wireless sensor networks, these compartments could be

1. Secure Code-Update: over-the-air update of sensor compartments,
2. Secure Storage: provides secure (and isolated) storage for e.g., cryptographic keys or measurement data of the sensors,
3. Secure Measurement: The sensing application performing the measurements (e.g., thermal, movement),
4. Secure Network Connection: The compartment responsible for connecting the sensor node with the mesh network, and
5. Other sensor application: e.g., further handling of measured data

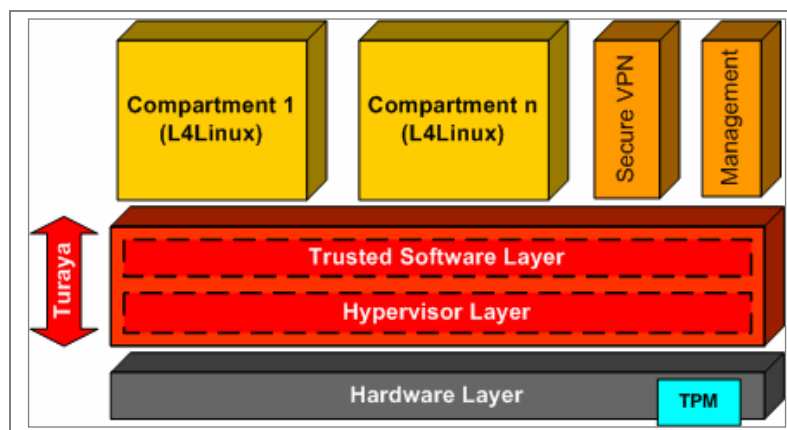


Figure 27 - TURAYA-based Trusted Computing Architecture

2.5.4.2 TURAYA's Hypervisor Layer

The hypervisor layer of the TURAYA security kernel acts as a traditional Virtual Machine Monitor (VMM) by managing the hardware resources of the sensor (memory, CPU, flash, network, ...) and providing basic virtualisation support. Due to the modular concept of the TURAYA architecture, different VMMs and microkernels can be used for the hypervisor layer.

In the following, an instance of the TURAYA security kernel based on the L4 microkernel (cf. Figure 28 - Components of the TURAYA Security Layer) is used. On top of the microkernel, fundamental services dedicated to resources are executed. These include services for the management of processes, memory and interrupts as well as those providing device drivers and enforcing system-wide security policies.

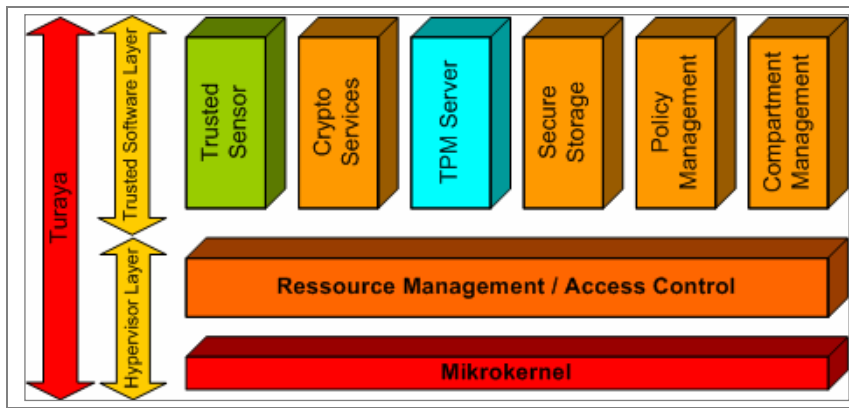


Figure 28 - Components of the TURAYA Security Layer

2.5.4.3 TURAYA trusted sensor node used inside WSAN4CIP

The hardware of the trusted sensor node is an OMAP-3530 based BeagleBoard rev. C4⁵ containing an ARM-Cortex A8 CPU. It has connections for SPI-, I²C- and GPIO-busses in order to connect external sensors to the node as well as USB for WiFi-connection and video sensors. Figure 17 shows the sensor nodes for the EDP-demo in an IP65-housing. Below the BeagleBoard is a custom-made circuit board with connectors to the SPI-bus, where the thermal sensor can be connected to.



Figure 29. TURAYA Trusted Sensor Node with BeagleBoard

As a software basis, the TURAYA Hypervisor Layer uses the OKL4 microkernel, which provides the required resource isolation. On top of the microkernel, a virtualised Linux (OK::Linux) is used. Figure 31 -

⁵ <http://beagleboard.org/>

TURAYA software architecture used inside WSAN4CIP illustrates a possible instantiation of the TURAYA trusted sensor node used inside WSAN4CIP.

Different security modules have been developed for the used sensors, the first proof of concept uses an LM70-thermal sensor. The trusted sensor node is fully compatible to the Silex-SX560 node, see Figure 30.



Figure 30: Thermal-Sensor for the BeagleBoard Trusted Sensor (proof of concept)

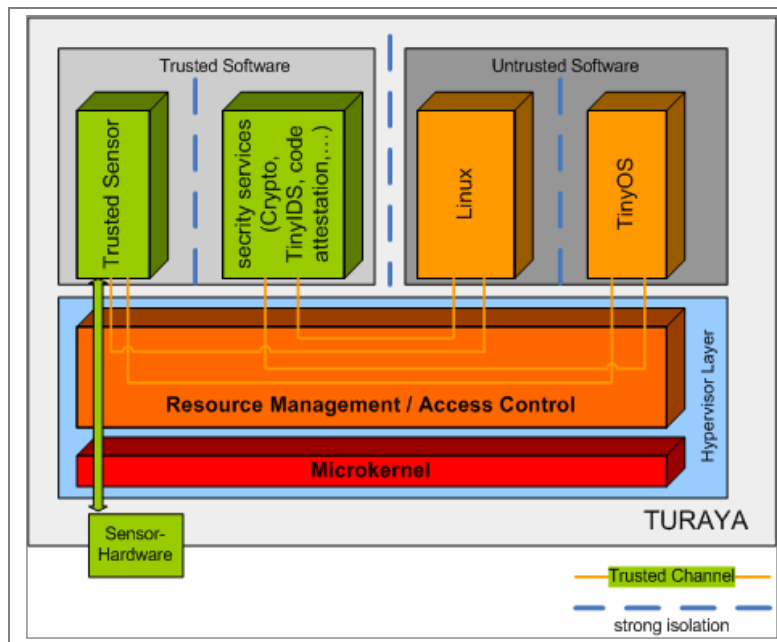


Figure 31 - TURAYA software architecture used inside WSAN4CIP

2.5.5 Description of DTSN

Although DTSN is a transport layer protocol, the decision was made to implement it on top of UDP, as a *daemon* service that other applications can use. The DTSN service shall be assigned a specific UDP port number. In fact, such solution is similar to what is commonly done in RTP/RTCP implementations. The main reason for this decision is to make software development easier (e.g. use of the sockets interface) taking advantage of the fact that the EDP demonstrator is based on IEEE 802.11 and all applications are IP-oriented.

2.5.5.1 Subset of specification to be implemented

The DTSN functionalities to be implemented are the following (see Deliverable 3.2 for detailed specifications):

- End-to-end full reliability service.
- Caching mechanism using uniform cache weights for concurrent flows.
- Optional session timers (for multimedia flows).
- Authentication and integrity check for ACK and NACK messages.

2.5.5.2 Software architecture

The DTSN software architecture is depicted in Figure 32, which shows how a client application connects to the DTSN *daemon* using the API interface, as well as the main modules comprised by the latter.

Communication of the interface primitives between the applications and the DTSN *daemon* is accomplished through UNIX sockets. The API is implemented as a dynamic library, whose public function calls abstract the RPC implementation. Communication between DTSN *daemons* through the network is performed over UDP/IP, using an UDP socket bound with a well defined port number.

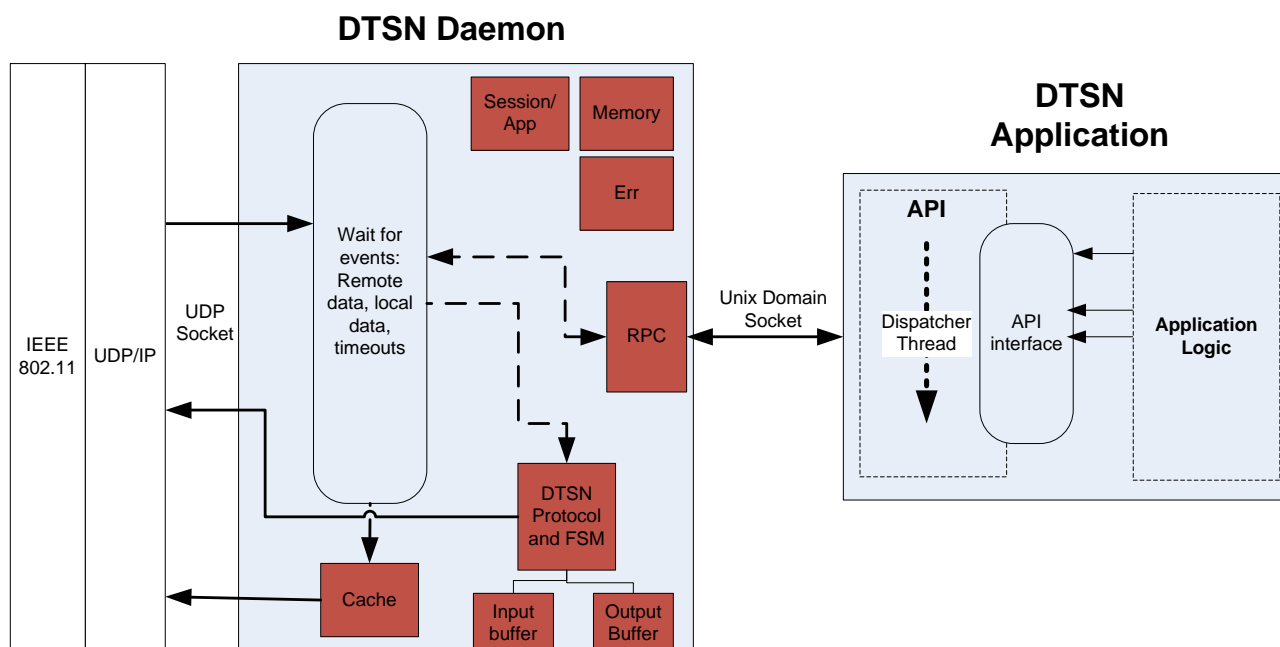


Figure 32. DTSN software architecture.

The functions performed by the different modules are the following:

- **DTSN Protocol and Finite State Machine (FSM):** Implements the source and destination DTSN protocol and FSM.
- **Cache:** Implements the DTSN intermediate node functionality.
- **RPC:** Implements the Remote Procedure Call (RPC) communication between the DTSN client application and the DTSN *daemon*. This RPC module is build on top of a UNIX domain socket.
- **Memory:** memory management with tight control over memory allocations. Memory management is specially important in embedded systems.
- **Err:** Logging facilities.
- **Session/App:** Management of the lists of active client applications and respective sessions.

2.5.5.3 Interfaces provided to the applications and the routing layer

DTSN offers a specific API to the applications, as defined in Deliverable 3.2 [7]. Communication of the interface primitives between the applications and the DTSN *daemon* shall be accomplished through UNIX sockets. The API is implemented as a dynamic library, whose public function calls abstract the RPC implementation.

DTSN source and target entities use UDP sockets currently, the RPL implementation will support this on source and the target. On intermediate nodes the DTSN-RPL interface will be the internal interface defined in Deliverable 3.2.

2.5.6 Description of Secure Routing

The Routing Protocol for Low-power and Lossy Networks (RPL) [8] is designed for wireless sensor networks. In such networks, the links may be unstable and the participating nodes may have power, computational, and storage constraints. Usually in sensor networks, there is one or a few special nodes called base stations. A base station is responsible for collecting the data measured by the sensor nodes and to control these nodes. Therefore, the destination or the source of most of the flows is a base station.

RPL is designed according to this philosophy. There are upward routes directed from sensor nodes to the base station and there are downward routes directed from the base station to the sensor nodes. The upward routes and downward routes are established independently. The support for downward routes is optional, thus, a network can operate according to the RPL standard even if only sensors can send messages to the base station. RPL also supports the communication between two regular sensor nodes by combining upward and downward routes.

The operation of RPL is based on the principles of distance vector routing and on the notion Directed Acyclic Graphs (DAG). In particular, through the exchange of distance information between neighbouring nodes, the network maintains one or more Destination Oriented DAGs (DODAGs), where the destination is the DODAG root, which is typically a base station. Upward and downward routes are selected along the edges of these DODAGs. Point to point communication is also possible in RPL using the upward and downward routes. For a more detailed description of the RPL protocol and the extensions we suggested, see Deliverable D 3.2 [9].

3 FWA prototype

3.1 Description of GUI

A Machine-to-Machine (M2) open source system called Mango (<http://mango.serotoninsoftware.com/>) has been used to show the information provided by the wireless sensor network. Mango has been chosen between a list of possible SCADA system because of its active support and for being open source.

By using a web application instead of a desktop application allows the SCADA system to run in a wide variety of architectures and operating systems. Also, it is rather simple to deploy new versions of it when changes are made. All these changes are instantly received by the clients without having to carry out complex administration tasks.

Although Mango allows the so-called public views, in which no login information is required, we have only developed private views. Only logged users may use the system. The appearance and behavior of each item in the screens is modeled as a javascript script and CSS in separate files, to allow changes to be made on the presentation without affecting the rest of the system.

FWA demonstrator consists of the Rohrbruchsicherungen screen shown in Figure 33. The behavior of FWA demonstrator, unlike EDP, is gateway event-driven and not user event-driven. This means that no user interaction is needed as the screen is automatically updated when new information is received.

There are two different classes of items on the screen: text items, called stacks, and data items. Stacks are composed of three text independent boxes while data items are composed of one box. The background color of each item shows if it is in normal, alarm or highlighted status and is configurable by defining its properties in a CSS file. The right part of next figure shows a detailed view of a text stack which contains an element in normal status (white background), another one in alarm status (red background) and the last one in highlight status (yellow background). The left side of next figure depicts two data boxes showing alarm status and normal status.

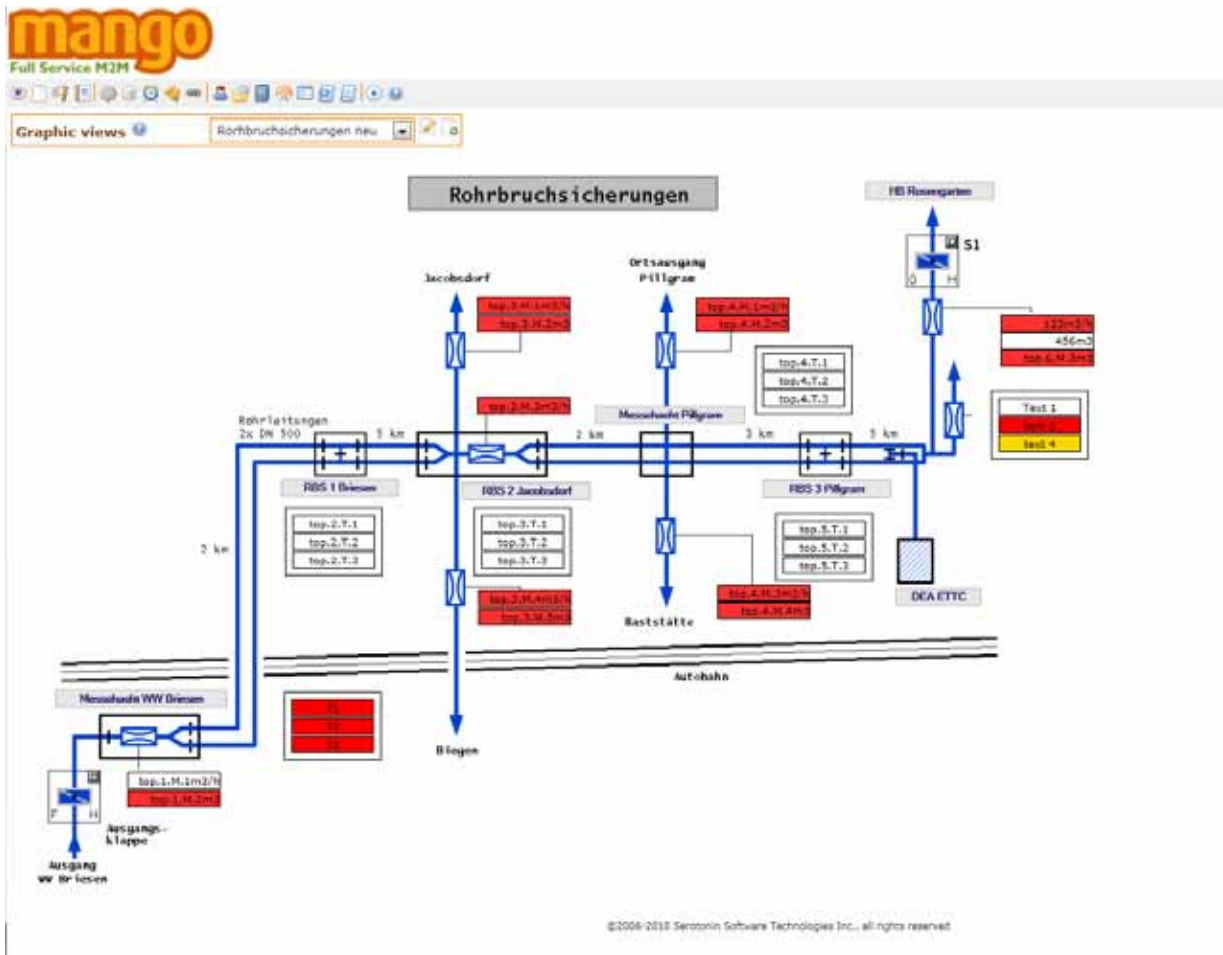


Figure 33 – FWA demonstrator main screen showing some random data values



Figure 34 – Text Stack (left) and two data boxes (right).

3.2 Description of Tiny DSM

This section introduces the tinyDSM middleware that was designed within the project to provide the distributed shared memory abstraction and to support the application development, considering the specific environment it should work within – the wireless sensor network.

This section discusses the set of functions the middleware provides:

- the shared memory abstraction,
- compile time event definition and runtime detection,
- data replication to improve the robustness,
- compile time definition of system behaviour using policy parameters,
- interfaces the middleware provides, and
- methodology how to use it.

3.2.1 The tinyDSM goals

Before specifying the goals it is necessary to think about the application environment – the wireless sensor network. This environment can be classified as a loosely coupled multiprocessor system with a potentially great number of nodes, each equipped with own memory. However, in that kind of system, there are several limitations that need to be taken into account. Regarding the nodes themselves, they have rather constrained resources, i.e., their computation power and the amount of available memory are quite limited. Taking the network as a whole, the connection means between the nodes are limited as well, i.e., the transmission speed and the link reliability are quite poor. The available energy is a limited resource as well, causing a trade-off between the lifetime of the desired system and the allowed power consumed by the nodes. All these constraints combined with the fact that the network is not always static, since nodes may disappear or new nodes may appear, causes the programming of wireless sensor network applications to be a very interesting and challenging task.

Wireless sensor networks are considered to be the key enabling technology for a large variety of innovative applications. But still, the majority of applications is using a wireless sensor network in a passive way, i.e., the sensor network is used to measure and store data that is then requested by the external application by sending queries into the WSN whenever the data is needed. This type of data storage and query technique is quite well researched. Prominent concepts are tinyDB[19], cougar[23], and tinyPEDS[4]. There are several other middleware approaches that are the result of research in the distributed computation and data storage domain [18], [12], [11] and [15], just to mention a few. The tinyDSM approach differs from those by being the first one that aims at providing really DSM like data storage.

In order to use the potential of a wireless sensor network it is necessary to exploit its power that is the ability of cooperative problem solving due to the high parallelism. However, the main advantage of parallel computing in sensor networks is not the increase the total computation power. Even if it may be of some importance for specific solutions, it is rather more interesting to go in the direction of distributed thinking or reasoning based on global or distributed knowledge. In such applications it is important to have the possibility to use some shared memory abstraction.

There is also a class of applications that require the sensor nodes in the network to become active in a certain situation. Health care and homeland security applications are the prominent examples where the sensing extended by detection provides the most advantages. Detection of some predefined situation can change the behaviour of the network or part of it, e.g., causing changing the sampling rate or sending the latest measurements to a predefined sink. Such applications will profit from the ability to be automatically notified about specific changes in the data storage.

The tinyDSM middleware was designed to support both these classes of applications. The goal was to provide a distributed shared memory abstraction with configurable replication and consistency parameters as a basic data storage concept on top of which an event detection mechanism is realized as well. The benefit of this solution is that a certain part of the network can change its behaviour simultaneously due to the common knowledge on the changes in the shared measured parameters or state data. Thus, events can be defined on a more abstract level. And doing this using an event definition language is much easier and faster compared to hardcoding certain behaviour into sensor node applications and wiring it to deployed protocols.

Another goal was to make the application development easier but without taking the complete control from the application designer. She needs to know what happens in the system, at least to a certain extent. It is currently a trend to provide tools that allow anyone to automatically generate application code based only on some simplified description. However, in the area of sensor networks it is not optimal and having at least basic knowledge is absolutely necessary to be able to estimate the results of own requirements. Anyway, a middleware that provides a clear interface and is built using a pre-defined set of functional blocks that are composed together and configured according to the requirements can greatly reduce the application development time, simplify the testing and also provides reusability of code and configuration.

Additionally, in order to support the heterogeneity of possible deployments the tinyDSM middleware was designed to be as hardware and software independent as possible. This goal was achieved by a pure C programming language implementation of the main logic blocks and a set of adapters that allow using the middleware in different hardware and software configurations.

3.2.2 The architecture and interfaces

The architecture of the tinyDSM-based application consists of the following modules (see Figure 35):

- **Application Logic** controls the behaviour of the nodes that build up the global application; it defines the sources of data and behaviour in case of event detections. It can read and write the shared data from the middleware.
- **OS Adaptation Layer** maps the services of the environment or operating system to the functionality needed by the middleware and also specifies the final middleware interface available to the application logic.
- **Event Logic** is responsible for detecting the events and notifying the application in such case.
- **Replication Logic** takes the decision on the replication of data, storage of new data and controls the data locating.
- **Query Logic** is responsible for interpreting incoming messages (queries or requests) and building results into answer messages. It may also allow the use of complex database-like queries issued by the user.
- **Policies** are a virtual component that is actually compiled in the other modules and determines the instantiation of tinyDSM during the implementation phase and configures its behaviour.
- **Memory Manager** controls the physical data storage structures on the node communicating with the hardware platform directly or via the operating system hardware abstraction layer.
- **Communication Interface** handles the communication with other nodes. It may also include security functionality like encryption or integrity tests of the data.
- **Operating System, Protocols and Hardware** constitute the target system where the application is deployed. The specific combination of these can be also referred to as the platform. The hardware influences the physical features that can be provided. The operating system defines the programming language and convention. And the protocols are a library of available modules for the given operating system and hardware.

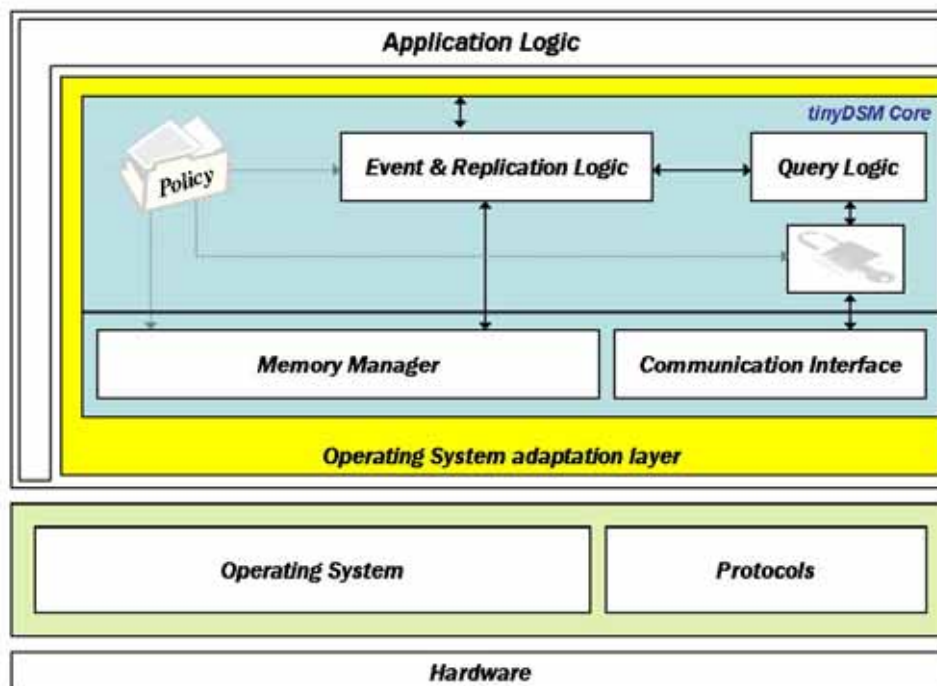


Figure 35: The architecture of a system based on the tinyDSM middleware

The core of the tinyDSM middleware is implemented in C programming language. This solution has the advantage that it can be applied in any environment where the C module can be compiled together with the application code or used as an external library. The most of the operating systems used in the wireless sensor network area, like tinyOS [17], Contiki [13], or Reflex [20] are C++ or C-based, but it is also possible to use tinyDSM in Java programming language, using the Java native interface (JNI).

Different target systems provide services like timers; interrupt handling and communication means differently. Thus, it is necessary to provide an adapter – the OS adaptation layer that adapts these services to the needs of the tinyDSM middleware core. This layer needs to be provided once for each new environment and can be reused for all tinyDSM applications for that platform. Additionally, if the services of the operating system are provided in a hardware independent way, then the adaptation layer can be used on every hardware platform supported by that operating system. This layer enables the tinyDSM middleware to be used in heterogeneous systems and the main logic of the middleware, can stay unchanged.

Since the adaptation layer encapsulates the tinyDSM middleware, it specifies the interface actually provided to the application logic. Depending on programming convention required by the operating system the interfaces of the middleware can be adapted so they can be used by the application logic.

Figure 36 shows the location and interfaces of the operating system adaptation layer. In this Figure the adaptation layer encapsulates the tinyDSM middleware completely separating its original interfaces from both, the application logic and the operating system interfaces.

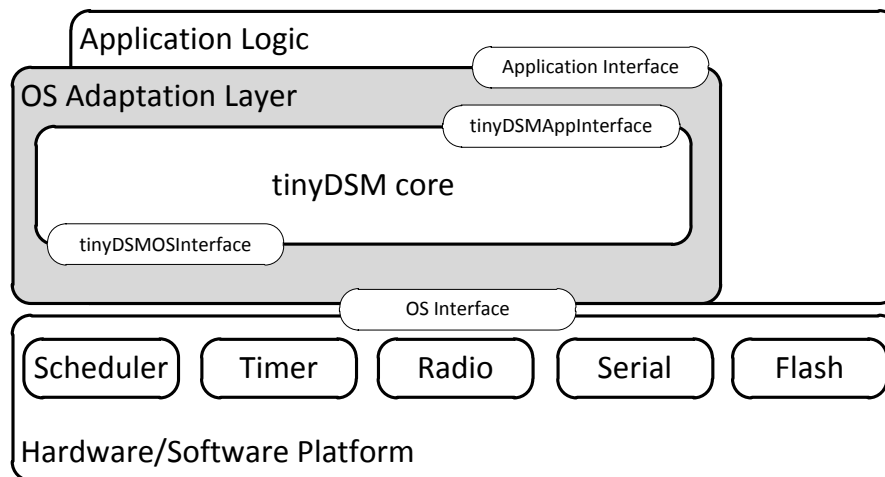


Figure 36: The Operating System adaptation layer – its location and interfaces in a tinyDSM based system

The adaptation layer can be reused between different tinyDSM based applications that use the same software and hardware platform. This is due to the fact that the interfaces of the tinyDSM middleware core are fixed. This reusability improves the application development time.

The main diversification of interfaces and functionality is regarding the phase of application life. The tinyDSM middleware supports the application development in the off-line phase and provides the on-line functionality to the application during run-time. The following paragraphs describe the middleware interface delivered to the running application and then the services of the off-line phase to show how they help the realization of the former ones.

Figure 37 shows the interfaces the middleware provides during the run-time. There is a differentiation between the part of the application that resides locally on the node and the external application, represented, e.g., by end-user devices. The part of the application that resides on the nodes uses the DATA and the EVENT interfaces. The DATA interface provides access to the shared data, allowing reading and writing. Using the EVENT interface the application is notified about the occurrence of pre-defined events. These events definitions are based on values of the shared data items.

For the global application the tinyDSM provides the QUERY interface. Currently it only supports read and write operations similar to these provided by the DATA interface, but it can be extended to answer more complicated queries than those supported by the DATA interface.

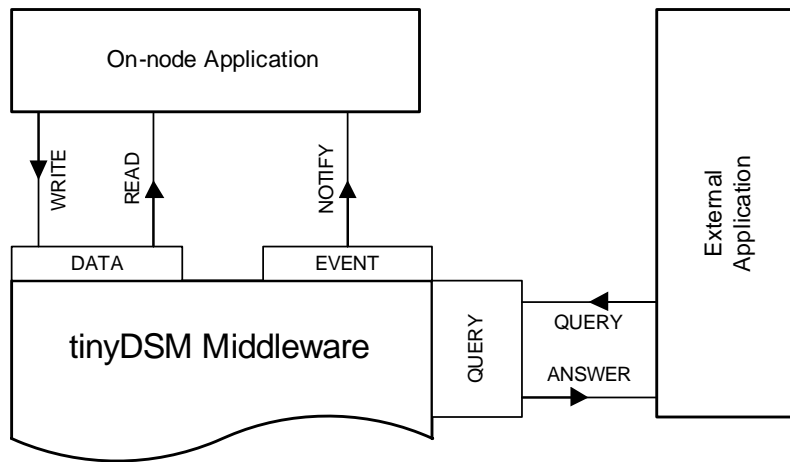


Figure 37: The tinyDSM interfaces

Thus, from the application perspective the middleware provides the following services in the on-line phase:

- WRITING the shared data,
- READING the shared data,
- NOTIFICATION in case a defined state of the data is reached,
- Answering external QUERIES based on the content of the memory.

In order to provide the desired services to the application the tinyDSM middleware needs to be configured properly in the off-line phase. Since there is no single solution that fits all requirements of all possible applications, there is a need to adjust the middleware. In the off-line phase the application developer is able to parameterize the middleware to the application needs. This is realized using a specified set of keywords that are inserted in the application code to define the set of shared variables and to specify the desired behaviour of the distributed shared memory middleware while handling these variables. A pre-compiler tool translates these keywords producing middleware source code that is ready for compilation, after which the application containing the middleware is ready for installing on nodes (see Figure 38).

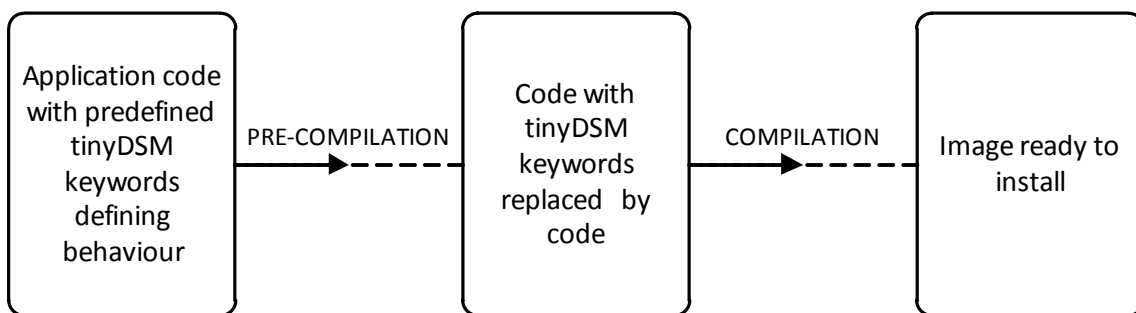


Figure 38: The two-step compilation approach

3.2.3 The tinyDSM data sharing concept

The basic concept behind the tinyDSM middleware is to provide means that allow sensor nodes to share their data in an application defined way following the concept of distributed shared memory as close as possible.

In order to define the shared memory space the application developer specifies a set of shared data items that are foreseen to be used by the application and are to be managed by the tinyDSM. Each data item in this set has a specified type and the desired way it shall be handled defined using the policy parameters. This

definition is common for the network covered by the application and the identities of these data items as well as the handling they require are known to all the nodes in this network. These shared data items are further referred to as shared variables, or simply variables.

A shared variable can be defined either as a global variable or as an array variable, where the latter is the default setting. In case of a global variable there exists only one single entity of this variable in the network and one of the nodes manages it. In contrast, in case of an array variable, there exist as many entities as many nodes are in the network, i.e., each node manages its own entity of that variable. The entities are logically independent shared data items, they store different values and are handled separately, but according to the same rules defined for the variable. These two different ways to define a shared variable allow generating either a single shared data item for the whole network, or an array of shared data items, one for each node, independent of the size of the network. The latter is similar to the definition of a vector proposed in [16].

To help understanding the concept of array and global variables an example of each will be given. An array variable temperature can be used to store the temperature measurements taken by each node. Each entity is independent and there are as many of them as many nodes are in the network. Every node writes its measurements into its own entity. On the other hand, a global variable period can control the period between each measurement. There is only one entity of the variable in the network, relevant for all the nodes.

As mentioned above, each entity of a shared variable has its manager also referred to as owner. The owner is the only node that is allowed to directly write this entity of the variable. When the owner writes to the entity, the written value is further handled in the middleware as the instance of the entity of the variable, or simply, the instance of the variable. An instance is represented by a tuple consisting of address and data fields (see Figure 39). Each instance is unique and the content of the tuple allows distinguishing between any two instances. The VariableID specifies the variable, and thus, the data type and handling. The identity of the node (NodeID) specifies the manager of the entity. The timestamp (or version number) allows distinguishing any two instances of the same entity. These addressing fields are completed by the Value.

VariableID	NodeID	Timestamp	Value
------------	--------	-----------	-------

Figure 39: The structure of an instance of a variable

Thus, the shared memory space consists of a set of instances of the defined variables. It can be seen as a tuple space, but it can be also seen as a distributed database with a single table, whose records are defined by the structure of the above mentioned tuple. An instance can be added to the shared memory space, but it cannot be explicitly removed or changed. It can be discarded due to its expiration or it can be lost if the node that stores it disappears. Multiple instances of an entity can be present in the shared memory space if the configuration for the variable enables history. These instances are chronologically ordered by the timestamp (or version number) and create a log of value changes in the entity of the variable. If no historical storage is defined for that variable, then the previous instance expires as soon as a new one is created.

In order to address the data items in the shared memory space the identity of the variable is used as the primary part of the address. It reduces the set of instances to those containing the values of the chosen variable. For an array variable it needs to be extended by the identity of the owner to point out the entity of interest. For a global variable the middleware will automatically locate the current owner of the single entity. If the current configuration for the variable allows storing historical data and the desired item is not the most recent one, then the timestamp or version number can be specified as well to choose a specific instance (see Figure 40).

Variable		Entity		
		Instance		
VariableID	NodeID	Timestamp	Value	
0	1	1	1	Four instances of two entities of one variable
0	1	2	2	
0	2	1	3	
0	2	2	4	
1	1	1	5	Two instances of one entity
1	1	2	6	
1	2	1	7	
1	2	2	8	
2	1	1	9	
2	1	2	10	
2	2	1	11	
2	2	2	12	

Figure 40: Addressing of the data in the tinyDSM middleware

In order to ease the access to the data in the shared memory space and to increase the robustness of the storage, the instances of the entities can be replicated on nodes other than the owner. The policy chosen for the variable controls the way the replica holders are chosen from the nodes in the vicinity of the owner. The nodes can be obligated to store the replica, or they can for instance decide on a random basis if they want to store it. The decision is taken per entity and is obligatory. If the policy for a variable enables replication, then an owner broadcast the newly created instances of entities of the variable after the write operations. The nodes in the vicinity can use these updates to refresh their replicas of the entity.

The replication can be adjusted regarding several aspects. If it is enabled in the policy, the replication can be configured regarding range (spatial limitation), saturation and frequency. The replication in tinyDSM is based on updates, since due to the small sizes of the shared data items updating is of advantage compared to invalidation. The concept of nodes in the vicinity is similar to those presented by Hood [21] and Abstract Regions [22] – in tinyDSM it is based on the communication means – n-hop broadcast.

The replication of the data allows any of the replica holders to perform a read operation on the replica. This reduces the delay of the read operation and decreases the communication costs in applications where the number of read operations is larger than the number of write operations. Additionally, such a node is able to answer queries for which it has the appropriate data stored without the need of forwarding the query to the owner node. It is also of advantage if the changes of the value of an entity that belongs to one node influence the behaviour of the nodes in its vicinity. An additional important feature of data replication is that it assures the information to be available even if some nodes are exhausted or in sleep mode. Figure 41 shows the idea of spatially limited replication (a) and its advantage in case of read operation (b).

The middleware was designed to provide controlled replication of the data that is stored in it in order to increase the reliability and availability of the data. The primary concept was to allow specifying a area of replication that surrounds the owner of the data and within this area it has to be possible to distribute the copies of the instances of the variables in a random way, to achieve an equal distribution of these.

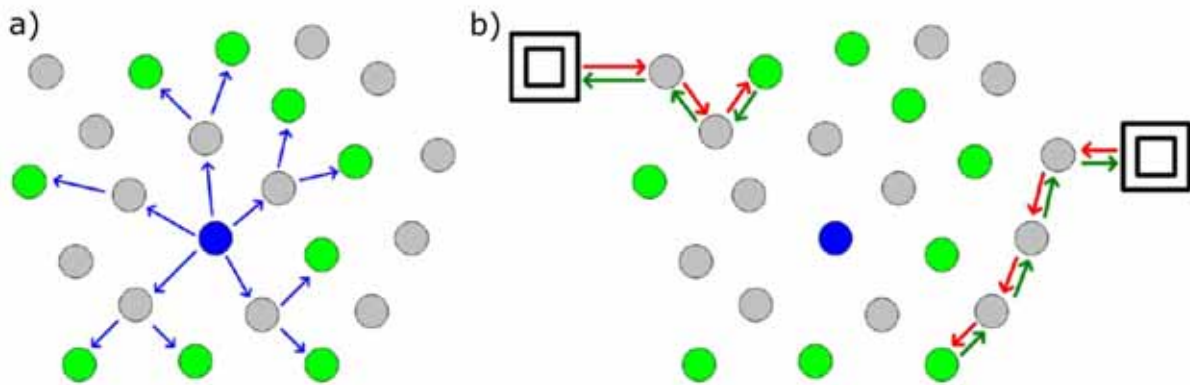


Figure 41: The data replication

As already mentioned the owner is the only node able to directly write its entity of a variable. Any other node willing to write this entity needs to send a write request to the owner. The owner performs then a local write operation and, if requested by the policy, sends an acknowledgement to the requesting node. This solution has been chosen because of the spatial focus of the replication, i.e., the write request may come from any node in the network, even a very distant one, but since the replication area is the vicinity of the owner, this is the best node to initiate the update of the replicas. But an even more important reason is to keep the right order of the write operations needed for consistency model research. This choice may cause the owner to become a bottleneck, but the complexity of the system would suffer from extending the write operation to allow the copies to be writeable.

The tinyDSM supports the multiple readers and single writer (MRSW) access pattern. But the write operation limitation is mitigated by the possibility to issue a write request to the node able to write the given entity. For global variables these limitations can be softened by enabling the migration of ownership, e.g., to cope with dying nodes, or by distributed ownership to distribute the tasks of the owner over some redundant group of nodes (especially interesting in dense networks) or combination of both. These options are especially interesting for global variables, since the ownership bindings for entities of array variables are fixed by definition.

The event mechanism available in the tinyDSM middleware allows monitoring chosen variables. It is based on compile time definitions to reduce the complexity of the evaluation logic. An event is defined as a logic equation with the shared variables in its terms. Such an event equation is evaluated each time a variable present in it changes its value. The result of this evaluation is stored in a special event variable associated to each event. These variables are shared as well, but they are read-only for the application and their type is fixed to Boolean. Except of that, they are like standard shared variables. An event variable can be used in terms of other event definitions, but not in the one it is associated to. Depending on the definition of the event the application is notified either about every evaluation, about the change of the evaluation result or is not notified at all. The last option was designed to support sub-events used in equations of several other events.

Following the concept of event variables that store the result of the evaluation of a defined expression, the basic functionality of shared variables was extended to enable defining a variable that automatically derives its value from an expression with other variables as terms. In this case it is forbidden to create mutually dependent variables, i.e., two variables cannot derive their values from each other. The expression is re-evaluated every time any of its terms changes its value.

The definitions of shared variables and events are provided in a configuration file and are used by the tinyDSM pre-compiler to adapt the middleware to the needs of the application. The desired handling of the instances of a given variable in the system is specified via the chosen policy parameters. The policy parameters can be set one-by-one or as pre-defined policy sets provided in policy file that can be used to simplify the definition and increase its readability. The syntax of the definitions of a shared variable and an event are as follows:

```
distributed [global] [policy parameters] type name[ = value or expression or {expression, trigger}];
event [global] [policy parameters] name = expression or {expression, trigger};
```

Default setting for the kind of variable is an array and if a variable shall be defined as a global one, it has to be explicitly stated in the definition by using the global keyword. Actually, this is one of the policy parameters since it controls the handling of the instances of the variable, but due to its importance and for the sake of clarity it is provided separately in the above definitions.

The specification of the policy parameters is a space separated list. Depending on the kind of the parameter it is either an appearance of the parameter name in case of switching parameters or the name followed by the values that quantify the parameter. A parameter set can be defined as a macro with a defined number of values.

After specifying the other policy parameters or choosing the predefined policy sets that shall be used it is necessary to specify the name of the variable or event. In case of a variable it is also necessary to specify its type, which can be any standard C programming language type. If the defined type is an array or structure the definition results in defining multiple shared variables that represent the fields or elements of the chosen type.

In contrast to an event the entities of a variable can be initialized during the definition. If no initial value is given all the entities of the variable are marked as unset, what is the default case for the entities of an event variable. Such an unset entity cannot be read and remains in this state until it is written for the first time, either by the application or as a result of the evaluation of the expression associated in the definition.

```
enum triggers {
    FTR_NC    = 0x0, // event() not called - default
    FTR_OC    = 0x1, // event() called on change
    FTR_ET    = 0x2, // event() called everytime
    FTR_MASK  = 0x3, // event() Function TRigger mask
    STR_OC    = 0x0, // value set on change - default
    STR_ET    = 0x4, // value set everytime
    STR_MASK  = 0x4  // value Set TRigger mask
};
```

Listing 1: The constants defined for the trigger parameter

The *trigger* parameter specifies the condition that is required after the evaluation of the associated expression to update the value of the variable and, in case of events, to notify the application. The variable can be set after every evaluation of the expression or, per default, only if the result differs from the current value stored in the variable. Similar applies also to notifying the application about the result of the evaluation, but the notification can be also switched off completely, what is the default setting if no trigger parameter is provided. The value of the trigger parameter is set using one or a combination of the defined constants (see Listing 1).

3.3 Description of Secure Routing

In both prototypes, the same routing protocol is used, which is described in page 35.

3.4 Description of Secure Code Update

The specific setting of the demonstrator with wireless sensor nodes along a many kilometres long water pipeline shows the need for an over-the-air (OTA) reprogramming solution, as driving from node to node for a manual code update would be time consuming and expensive. The need for remote reprogramming is created by evolving application requirements and software errors. For an OTA reprogramming solution for a WSAN protecting a critical infrastructure, appropriate security mechanisms are keys to the success and acceptance of a code update mechanism.

The implemented OTA solution uses security mechanisms that were in detail described in D2.3. In this context, resource-awareness, time-efficiency are key goals of the code update solution. The efficient data structure and the use of elliptic curve cryptography contribute to the efficiency of the code update solution.

The fountain code based data transmission that was demonstrated and delivered in D2.5 is not used in this demonstrator. The reason is that Fountain Codes work best in a dense network where nodes have multiple

neighbors. The topology of the FWA demonstrator with nodes in a line at a distance of several km per node causes that nodes have only two neighbors so that Fountain Codes cannot play out their benefits.

4 Specification of system tests

The specification of the demo tests was accomplished according to a modular philosophy. For each demo, tests to be performed at three different levels of integration:

- Individual SW modules (including the GUI and WSAN-GW);
- Integrated WSAN (i.e. the WSAN section up to the WSAN-GW);
- Integrated Demo (GUI+WSAN).

For each test, the documentation includes the following items:

- **1 Test Card:** The test card is identified by a Test ID obeying to the following format, where the fractions in italics represent wildcards:
 - *ModuleName_Number* for individual module tests;
 - *DemoName_WSAN_ScenarioName_Number* for WSAN tests;
 - *DemoName_INT_ScenarioName_Number* for integrated demo tests.

The text card contains a test description and as well as the expected results. If some test card includes both functional and performance tests, functional tests come first.

- **1 Result Card:** The corresponding result card contains a description of measured/observed results.

The test cards for each of the demonstrators are found in Annexes A and B.

4.1 Test and result cards for EDP demonstrator

Test and result cards for the EDP demonstrator are shown in Annex A.

4.2 Test and result cards for FWA demonstrator

Test and result cards for EDP demonstrator are shown in Annex B.

5 Integration plans

This section describes the integration plans and milestones

5.1 Integration activities for EDP prototype

The integration activities of the EDP prototype followed the schedule in Table 3.

Table 3 – Integration plan for the EDP prototype.

Date	Delivery Milestone	Integration Milestone	Responsible Partners
15/02/2011		GUI+ SCADA/WSAN Gateway + + Test Applications	INOV + TEC + UMA
28/02/2011	Secure DTSN		INOV
31/03/2011	Secure RPL		BME
31/03/2011		Secure DTSN+Secure RPL	INOV + BME
15/04/2011		All except Secure μ Kernel	INOV + TEC + UMA + BME
30/04/2011	Secure μ Kernel		SRX
31/05/2011		μ Kernel integration	INOV + SRX
01/05/2011		Start of deployment	INOV + EDP
30/06/2011		Deployment finalized	INOV + EDP

5.2 Integration activities for FWA prototype

The integration activities of the FWA prototype followed the schedule in Table 4.

Table 4 – Integration plan for the FWA demonstrator

Date	Delivery Milestone	Integration Milestone	Responsible Partners
xx/xx/2010	Node hardware + software drivers for tinyOS		IHP
xx/xx/2011	tinyDSM for tinyOS		IHP
xx/xx/2011	tinyIDS for tinyOS		IHP
xx/xx/2011	MAC for tinyOS		LTU
xx/xx/2011	RPL for tinyOS		BME
xx/xx/2011	SCU/DCU for tinyOS		NEC
xx/xx/2011		MAC + RPL	BME + LTU
xx/xx/2011		MAC + node hardware	LTU + IHP
xx/xx/2011		MAC + RPL + node hardware	IHP + LTU + BME
xx/xx/2011		SCADA/WSAN gateway	IHP + UMA
31/07/2011		DCU integration	IHP

01/08/2011		Initial deployment + tests	IHP + FWA
15/08/2011		Final deployment	IHP + FWA

5.2.1 Description of the integration activities of the in lab prototype

The FWA demonstrator consists of several components (see Figure 42). The main components are the hardware platform together with its corresponding software drivers, the MAC protocol, the RPL routing protocol, the tinyDSM middleware, the IDS subsystem and the Secure Code Update (SCU) module. Initially, the main parts of the FWA demonstrator were developed and tested separately by their responsible partners. These tests were performed on configurations different from the final one defined for the FWA demonstrator. The following paragraphs first describe the development progress of the individual modules prior to the integration and then describe the integration procedure and outcome. Finally, the possible solutions to the recognized issues are given.

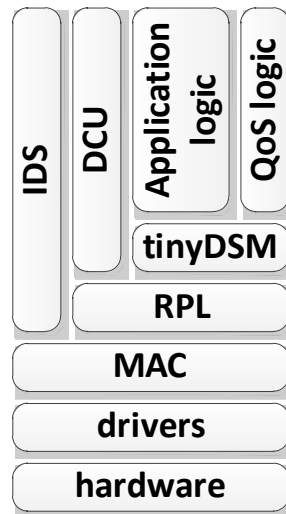


Figure 42: The on-node system architecture of the FWA demonstrator

The hardware platform was developed at the IHP in a way that it is possible to transmit the data with very high power. This was necessary to achieve the desired transmission range. The new platform required the porting of the tinyOS onto it. The porting includes definition of a new platform alias, defining all necessary compilation parameters, as well as providing the software drivers for the hardware components available on the node. The most problematic was the radio driver, since it had to be written from scratch and thus required also an extensive testing. The radio driver provides the software interface required by the MAC protocol, as well as the standard ActiveMessage compliant interface that allows the use of the radio by other applications. The first tests involving three nodes communicating with each other were performed and their results have shown that the communication is working (less than one per cent of messages were lost). These results correspond to the average communication quality available for wireless sensor nodes.

The MAC protocol was developed at LTU. It is a TDMA based medium access protocol. During its development, it was initially tested on the Mulle hardware platform. On this platform the radio hardware module provides the microcontroller with an accurate timing source that is used by the MAC protocol for timing purposes. This configuration performed well during the tests, the protocol fulfilled its tasks, i.e., the access to the medium in the data transmission part of the super frame was controlled by the result of the contention phase. However, in order to allow a clear separation of the phases the protocol relies on an accurate timing source.

The RPL routing protocol developed at the BME was first available for larger nodes, like those used in the EDP demonstrator. Such nodes have more resources available and are more powerful. This version was ported for the tinyOS operating system in order to be applied in the FWA demonstrator on less powerful nodes. This ported version was initially tested in the TOSSIM simulator and finally on a small set of MICAz nodes.

The secure code update (SCU) is representing the class of dynamic code update (DCU) approaches that allow reprogramming the nodes over the air. The approach was successfully tested on the TelosB nodes.

The tinyIDS is an intrusion detection scheme developed specially for wireless sensor nodes. It takes the constrained resources of the nodes into account and allows detecting anomalies in the network traffic as well as direct attacks on the nodes and code injections. It is a set of mechanisms that support the self-protection of the individual nodes and as a result of the network as a whole. The individual mechanisms were tested separately as stated in deliverable D2.4 [24]

The tinyDSM middleware allows sharing the data between nodes and is a required building block for the application logic and the QoS manager. TinyDSM was successfully tested in the tinyOS operating system on MICAz and TelosB nodes in the UbiSec&Sens project. However, the initial version of the middleware introduced in that project was extended with new functions during WSAN4CIP.

The first integration phase involved the integration of the MAC protocol and the hardware platform. During this integration of MAC many tests with multiple nodes communicating have been performed and many improvements have been made to the driver layer. Several of the identified bugs were hard to reproduce. For instance, we have discovered that in some rare cases when the driver layer goes in to TX or RX state and the node receives sync bits at the same moment, it can put node in an unexpected state and make further reception of packets impossible. This issue caused a repeated analysis of the radio driver states and resulted in changes that should help in keeping the states of the driver and the hardware consistent.

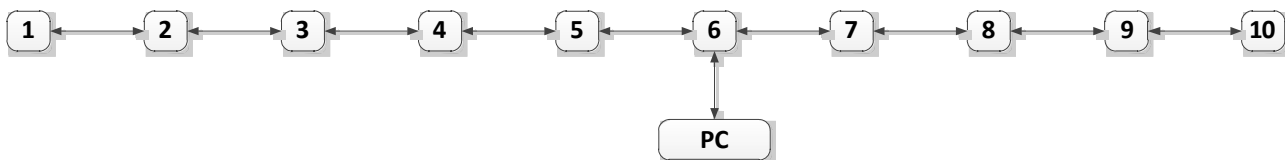


Figure 43: The test network used in the integration phase

During these tests we used an application that is a simplified version of the final application. The test network is shown in Figure 43. The network was consisting of 10 nodes building a line like it is foreseen in the FWA demonstrator and these nodes were equipped with a simple routing protocol based on static routing tables. Thus, the nodes were configured to forward the packets always to a node next in line. Anyway, due to the high transmit power all the nodes were able to receive the packets from other nodes, but these packets were not used and thus constituted to the noise in the channel, emulating a worst case scenario with high traffic channel.

The first tests of the test application were performed with an ALOHA like media access, i.e., the nodes were sending messages without taking care if the medium is free and without any cooperation. The application was generating a measurement once every five seconds on each node. These measurements were then delivered to node number 6, which was connected to the PC. The measurements were there collected and the completeness of the set was examined. First, we made an 8h successful test run with a total of 57600 transmitted measurements. This test included the ALOHA-like MAC, the static routing and the simple application. Even due the high traffic we achieved 88 per cent message delivery ratio. It is worth to notice that we will not have this overhearing and high traffic in the real scenario.

We discovered that the clock source available on the hardware node is not accurate enough for the TDMA MAC protocol. Thus, the nodes in the network were not able to agree on the access pattern during the data transmission phase of the protocol. This is caused by the fact that no accurate high frequency clock source is available and the clock of the microcontroller is generated by a digitally controlled oscillator (DCO) module that has a tolerance of up to several per cent. Such a precision was unacceptable for the TDMA based MAC protocol. Thus, in the initial integration phase the MAC protocol, as specified in WP3 could not be integrated and a simplified CSMA/CD MAC with optional acknowledgements (ACKs), has been applied for the further tests.

In order to test the integration results in the above mentioned network we again used the simple routing protocol and the demonstration application. Again, the application was generating a measurement once every five seconds on each node. And again, these measurements were delivered to node number 6, connected to the PC. For these settings we performed a 3 days long run with a total of over 600000 transmitted measurements. This test included the simplified (CSMA/CD) version of MAC, the static routing and the

simple application. In this test we achieved an average 85 per cent message delivery ratio. MAC with ACKs performed slightly worse, due to even more increased traffic.

The identified issues at that point are the missing accurate timing source on the hardware platform and the inability to work with less accurate sources on the MAC side. The next integration steps can include mechanisms to compensate the inaccuracy on the software side as well as ways to improve the accuracy of the hardware source using the available, more accurate, but slower, clock sources. The former include mechanisms like extending the guard times for the contention and data transmission phases or loosely synchronization of clocks on different nodes. The latter includes mechanisms like recalibration of the DCO using the more accurate clock provided by a 32 kHz quartz available on the hardware platform. The choice of the solution is to be defined and the expected outcome has to be investigated.

The next step involved the integration of the RPL routing protocol. The protocol has been successfully compiled and installed on the nodes. However due to a bug, the data in the transmitted messages was corrupted and the test application was not working. Further investigation revealed that the initial test application for TOSSIM was not checking the data correctness of the sent messages. Thus, the issue was not detected earlier. The observed traffic induced by the RPL suggests that the nodes were able to construct the routing tree, however this could not be validated by a successful data exchange. In a meanwhile this issue has been solved, but additional tests are required to confirm that the new version of the RPL is not affected anymore.

As soon as the protocol stack is integrated and tested it is possible to integrate the application-related part of the FWA demonstrator on top of the protocol stack. The integration will be followed by an extensive test phase to reveal the potential drawbacks of the integrated modules, as well as to investigate the performance of the integrated solution. The foreseen deployment includes the first test deployment with tests in the real environment. This phase includes the performance tests and real life tests of the modules like DCU to enable faster debugging and modifying the code on the nodes in this broadly distributed demonstrator setup. The test deployment is then followed by the long-term deployment of the final demonstrator system to check its durability.

6 Lessons learned

The integration of the FWA demonstrator did not succeed in the planned time. Unfortunately, the modules were developed isolated from each other and from the target hardware, which led to a more complex integration task than what was expected. There are two major problems that led to this situation. The first reason is the initial lack of clear and complete interface definitions between modules resulting in missing required functionality. The example of this issue is the missing accurate timing source on the node hardware. This accurate timing source is required by the TDMA-based MAC protocol in order to keep its timing rigours. A timing source is available on the node hardware, but its accuracy is not sufficient for the MAC module. There are ways to improve the accuracy of the available source, but it requires additional implementation and testing effort. It is necessary to specify the target/required accuracy and verify, if it is possible to achieve this goal on the available hardware.

The first problem is sometimes caused by the initial lack of the knowledge on the required final functionality as the module is developed. And in case the development involves multiple layers (hardware and software) the time required for updating the specifications of all the layers may grow.

The second reason is the narrowed testing procedure that disallows identifying all the bugs included in the modules in the development phase. The modules are usually tested in a simplified environment that hides some specific situations exposing unwanted behaviour. The examples here are the RPL protocol data bug and the problem with the states in the radio software driver. Testing of the latter was performed in a small network (two and three nodes) with limited traffic. During the tests after the integration with the CSMA/CD MAC protocol the traffic generated by the test application was much higher. This high traffic caused the node number 6, who was only receiving messages from other nodes, to hang in an unspecified state disallowing further reception of messages. Further inspection of the state machine revealed that the rare case, where two nodes were sending their messages at almost the same moment caused the receiving node to hang. These messages were corrupted due to collision anyway, but in order to recover from the unspecified state it is necessary to reset the radio module on the node and put it back into a correct one. This is achieved for the node 6 in the test setup by triggering a periodic sending. The node number 6 was simply sending its own measurements. This approach allows periodic resetting of the radio module to avoid the problem. In the final demonstrator setup the node that is the WSAN/SCADA gateway will be also sending the management messages and thus, the situation where a node is only receiving will be rare.

The individual software and hardware modules developed for WSAN nodes are very often specific and optimized or constrained due to limited resources. This specific features and requirements have to be matched by their counterparts or modules that deliver or use the given functionality. Additionally, due to the limited resources, the application development for WSANs is often a try-and-fail approach of combining different available modules. These modules often try to be generic, but this feature comes often by the price of reduced optimization.

What we learned during the integration phase is that modules that are foreseen to work together shall be also developed together or at least in parallel with a close feedback loop. In such an approach the integration phase is much longer and virtually starts at the beginning of the development reducing the surprise factor of the final integration result.

7 Conclusions

This deliverable shows the state of component integration for both demonstrators. As a summary, most components are available for both demonstrators (electrical substation and water treatment).

Results of tests performed on the available components show that their performance in the lab prototypes is very promising. It is expected that real demonstrators can show very interesting results.

References

- [1] [http:// www.wsan4cip.eu/](http://www.wsan4cip.eu/)
- [2] SX-560 Intelligent Programmable WLAN Module, 802.11a/b/g Wireless Solution for OEM Embedded Applications, datasheet
- [3] IEEE Std C57.13-1993, IEEE Standard Requirements for Instrument Transformers Sponsor Transformers Committee of the IEEE Power Engineering Society, June 1993
- [4] Norbert J. Ackermann, Current Transformers: Basics of Operation and In-Service Testing, www.msema.com/CT_Basics.doc
- [5] Calculation of the Current Transformer Accuracy Limit Factor, Application Note application note A/09.11.2004, ABB, November 2004
- [6] Evan Mayerhoff, Corona and its Effects, High Voltage Connection, Inc., www.highvoltageconnection.com
- [7] <http://www.hobut.co.uk/current-transformers/split-core-cts/series-80>
- [8] WSAN4CIP project *Deliverable D1.2 - Specification of WSAN4CIP demonstrators*, 2009.
- [9] RPL: IPv6 Routing Protocol for Low power and Lossy Networks. IETF Draft. Editor: T. Winter
- [10] WSAN4CIP Deliverable D3.2: Specification and analysis of dependable networking mechanisms for the WSAN4CIP application scenarios. Editor: António Grilo
- [11] WSAN4CIP Deliverable D3.2: Specification and analysis of dependable networking mechanisms for the WSAN4CIP application scenarios. Editor: António Grilo
- [12] T. Abdelzaher, B. Blum, Q. Cao, Y. Chen, D. Evans, J. George, S. George, L. Gu, T. He, S. Krishnamurthy, et al. Envirotrack: Towards an environmental computing paradigm for distributed sensor networks. In *Distributed Computing Systems, 2004. Proceedings. 24th International Conference on*, pages 582-589. IEEE, 2004.
- [13] P. Costa, L. Mottola, A.L. Murphy, and G.P. Picco. TeenyLIME: transiently shared tuple space middleware for wireless sensor networks. In *Proceedings of the international workshop on Middleware for sensor networks*, pages 43-48. ACM, 2006.
- [14] A. Dunkels, B. Grönvall, and T. Voigt. Contiki - a lightweight and exible operating system for tiny networked sensors. In *Proceedings of the First IEEE Workshop on Embedded Networked Sensors (Emnets-I)*, Tampa, Florida, USA, November 2004.
- [15] J. Girao, D. Westhoff, E. Mykletun, and T. Araki. Tinypeds: Tiny persistent encrypted data storage in asynchronous wireless sensor networks. *Ad Hoc Networks Journal (Elsevier)*.
- [16] R. Gummedi, O. Gnawali, and R. Govindan. Macro-programming wireless sensor networks using kairós. *Distributed Computing in Sensor Systems*, pages 126-140, 2005.
- [17] T.W. Hnat, T.I. Sookoor, P. Hooimeijer, W. Weimer, and K. Whitehouse. Macrolab: a vector-based macroprogramming framework for cyber-physical systems. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 225-238. ACM, 2008.
- [18] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, et al. Tinyos: An operating system for sensor networks. *Ambient Intelligence*, pages 115-148, 2005.
- [19] L. Luo, T.F. Abdelzaher, T. He, and J.A. Stankovic. Envirosuite: An environmentally immersive programming framework for sensor networks. *ACM Transactions on Embedded Computing Systems (TECS)*, 5(3):543-576, 2006.
- [20] S.R. Madden, M.J. Franklin, J. Hellerstein, and W. Hong. TinyDB: an acquisitional query processing system for sensor networks. *ACM Transactions on Database Systems*, 30(1):122-173, 2005.

-
- [21] K. Walther and J. Nolte. A exible scheduling framework for deeply embedded systems. In Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops - Volume 01, AINAW '07, pages 784-791, Washington, DC, USA, 2007. IEEE Computer Society.
 - [22] M. Welsh and G. Mainland. Programming sensor networks using abstract regions. In Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation-Volume 1, pages 3-3. USENIX Association, 2004.
 - [23] K. Whitehouse, C. Sharp, E. Brewer, and D. Culler. Hood: a neighborhood abstraction for sensor networks. In MobiSys '04: Proceedings of the 2nd international conference on Mobile systems, applications, and services, pages 99-110, New York, NY, USA, 2004. ACM Press.
 - [24] Y. Yao and J. Gehrke. The cougar approach to in-network query processing in sensor networks. SIGMOD record, 31(3):9-18, 2002.
 - [25] WSAN4CIP Deliverable D2.4: Specification of an IDS scheme and secure code attestation protocol for WSAN. Editor: Peter Langendörfer