DELIVERABLE 4.2
Development and performance evaluation of the ENTOMATIC network

Project Acronym	**ENTOMATIC**
Project Reference:	**605073**
Project Title:	**Novel automatic and stand-alone integrated pest management tool for remote count and bioacoustic identification of the Olive Fly (Batrocera oleae) in the field**

Deliverable 4.2 – Development and performance evaluation of the ENTOMATIC network

Revision: **V 5.0**

Authors:

Toni Adame, Sergio Barrachina, Albert Bel, Boris Bellalta (UPF)
Michel Chamoun (MTSYSTEM)
Carmen Capiscol (INOLEO)
Ilyas Potamitis (TEIC)
Frank Spiller (IMMS)
Ward Bryssinckx (AVIA-GIS)

Project co-funded by the European Commission within the ICT Policy Support Programme		
Dissemination Level		
P	**Public**	
C	**Confidential, only for members of the consortium and the Commission Services**	**X**

Revision History

Revision	Date	Author	Organization	Description
1	13/04/2016	Toni Adame	UPF	General structure
2	15/04/2016	Sergio Barrachina	UPF	Contiki considerations
3	19/12/2016	Toni Adame, Sergio Barrachina	UPF	Definition of ENTOMATIC network communication protocols
4	23/03/2017	Toni Adame	UPF	Revision / Update of information
5	31/04/2017	Toni Adame	UPF	Revision / Update of information
*	During the writing time	Michael Chamoun	MTSYSTEM	Assessment, revision and validation
		Carmen Capiscol	INOLEO	
		Ilyas Potamitis	TEIC	
		Frank Spiller	IMMS	
		Ward Bryssinckx	AVIA-GIS	

Statement of originality:

This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

EXECUTIVE SUMMARY

This deliverable (D4.2) is the output of the activities of Task 4.2 “Final design of the ENTOMATIC WSN” and Task 4.3 “Construction and testing of ENTOMATIC WSN” within Work Package 4 (WP4) “Design & Development of WSN”.

The deliverable describes in depth the design and mechanisms employed in the ENTOMATIC Wireless Sensor Network (WSN) responsible for transmitting all data captured from traps to the central gateway. Similarly, the mechanisms employed to retransmit this data from the gateway to the data receiver server are also explained.

Some of the techniques discussed have been developed as part of the previous ENTOMATIC Task T4.1 “Conceptual design of the ENTOMATIC WSN“, whereas some of them are entirely novel. In both cases, techniques and mechanisms included in the system fulfill the ENTOMATIC technical requirements regarding communications, compiled in D1.2 “Update of system specifications”.

The first part of the document analyses the state-of-the-art and describes the proposed solution for each one of the communication layers existing in the WSN; namely, physical, medium access control (MAC), network, transport, and application. Additional chapters offer more information regarding connectivity of sensors and energy issues.

The second part of this deliverable includes the definition of the protocol proposed to establish GPRS communication between the gateway and the data receiver server. Due to the use of a well-known and consolidated protocol such as GPRS, this section is focused on the routines and the schedule necessary to perform periodic GPRS transmissions.

In addition, specific subsections include the results corresponding to performance tests executed on the different elements of the ENTOMATIC communication system. These preliminary tests, which have performed both in a controlled testbed and in real scenarios, show the alignment of the proposed solution with the stated requirements of the project.

Once validated the communication protocol of the ENTOMATIC project, the next step in this field is to integrate these mechanism with the reception, processing and encapsulation of the information provided from the optoelectronic sensor into the established data transmission frames together with the already tested and validated information obtained from the temperature, humidity and luminance sensors.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

ACK	Acknowledgment
ADC	Analog Digital Converter
CCA	Clear Channel Assessment
CoAP	Constrained Application Protocol
CPU	Central Processing Unit
CSMA	Carrier sense multiple access
CSR	Cycle Stability Ratio
e2eACK	End-to-End Acknowledgement
FTP	File Transfer Protocol
GPIO	General-Purpose Input Output
GPRS	General Packet Radio Service
GSM	Global System for Mobile communications
GW	Gateway
HTTP	Hypertext Transfer Protocol
HW	Hardware
ID	Identifier
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of Things
IP	Internet Protocol
ISM	Industrial, Scientific and Medical
I2C	Inter-Integrated Circuit
LiPo	Lithium Polymer
LOS	Line-Of-Sight
LPM	Low Power Mode
MAC	Medium Access Control
MCU	MicroController Unit
NiMH	Nickel–Metal Hydride
NLOS	Non-Line-Of-Sight
OS	Operating System
OSI	Open Systems Interconnection
PCB	Printed Circuit Board
PDR	Packet Delivery Ratio
QoS	Quality of Service
R	Ring
RDC	Radio Duty Cycling
RF	Radio Frequency
RSSI	Received Signal Strength Indication
RTOS	Real Time Operating System
RTT	Round Trip Time
RX	Receiving
SACK	Selective Acknowledgement
SIM	Subscriber Identity Module
SoC	System on Chip
SPI	Serial Peripheral Interface
STA	Station (sensor node)
SW	Software
TCP	Transmission Control Protocol
TDMA	Time Division Multiple Access
TX	Transmission
UART	Universal Asynchronous Receiver-Transmitter
UDP	User Datagram Protocol
WSN	Wireless Sensor Network

1 INTRODUCTION

The current deliverable includes full specifications of the communication protocols and communication modules at the traps and the Gateway of the ENTOMATIC network, so that they fulfil the stated requirements from previous documents.

As shown in Figure 1.1, the whole ENTOMATIC network can be split into two differentiated parts:

1. An acquisition network, based on wireless sensors and used for the communication between the traps and the gateway. While the gateway is placed in a central position, a series of rings formed by scattered communication devices are deployed over the monitored area.
2. A cellular network responsible of transmitting the gathered information from the gateway to the ENTOMATIC data receiver server. In this case, the single requirement for the gateway is being provided with cellular coverage.

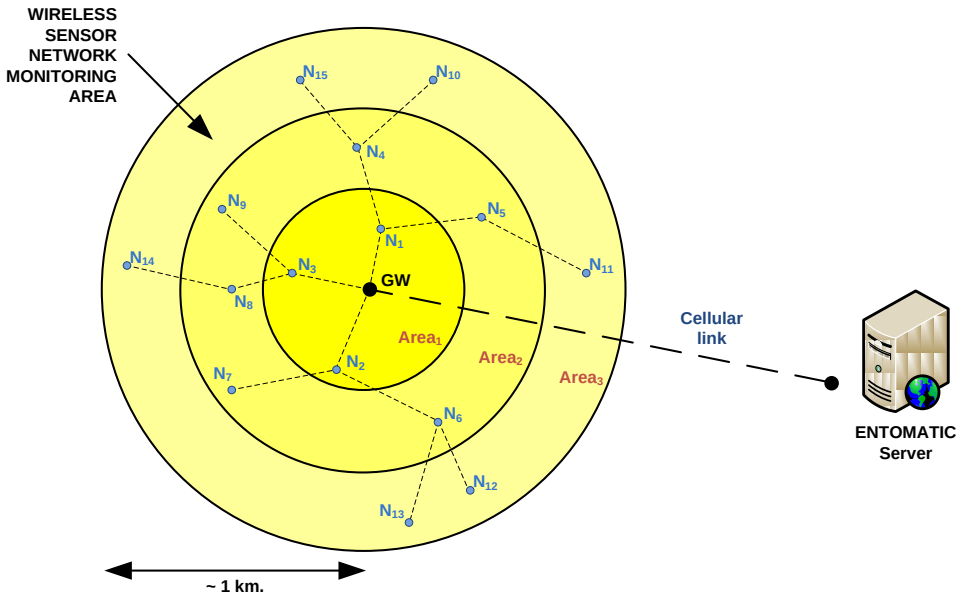


Figure 1.1: Conceptual design of the ENTOMATIC communication network

By following this network approach, the current deliverable has also been split into two main chapters, each of them devoted to a different sub-network. Thus, Chapter 2 describes the ENTOMATIC WSN and Chapter 3 includes all the information regarding the cellular link between the gateway and the central server.

Both chapters follow the same structure, which is summarized in the following lines:

- General considerations: Main subjects to have into account prior to designing the network
- Hardware platform: Detailed information of the hardware employed in the devices conforming the network
- Software platform: Definition of the main routines and mechanisms involved in the communication processes of the network
- Other: Additional aspects affecting the operation of the described modules
- Performance tests: Definition and results obtained from tests (simulated or based on real devices) of the different employed technologies

While the cellular link between the gateway and the ENTOMATIC server uses standard technologies (GPRS and encapsulated HTTP requests and responses), a full protocol stack for communication within the WSN has been designed, programmed, implemented and tested. This group of communication layers (MAC, network and transport) work in combination with the physical IEEE 802.15.4 foundations provided by the hardware employed in the network (see Figure 1.2).

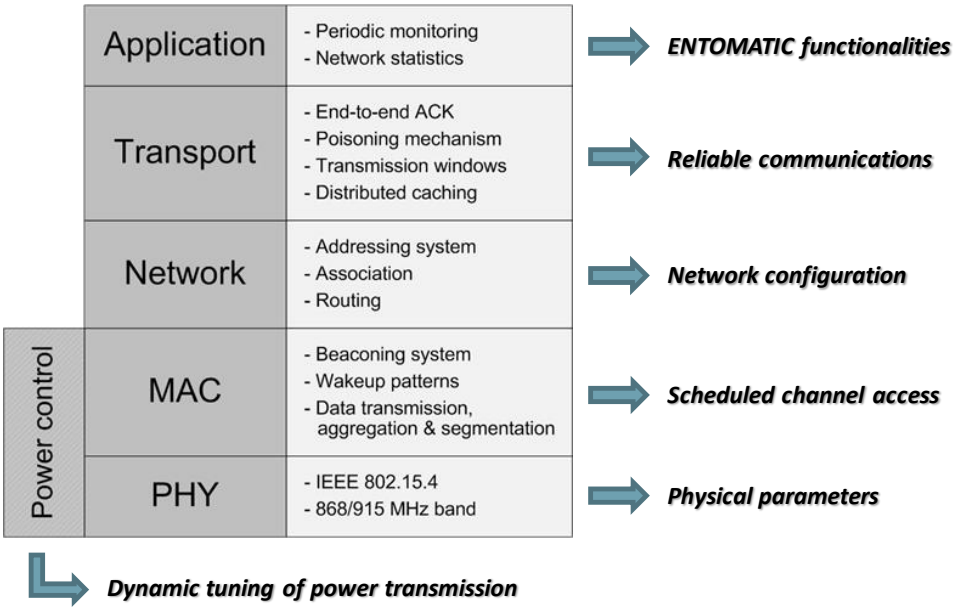


Figure 1.2: WSN Communication protocol stack designed and implemented for the ENTOMATIC project

And it is precisely inside these communication protocols where the engine and the main features of the ENTOMATIC communication system are located. Provided with different purposes and subsystems, all these layers share the same goal of ensuring the maximum network reliability while consuming the minimum energy. In this way, a thorough analysis of the currently different available solutions for each of the aforementioned layers is included in this document along with a comprehensive explanation of the subsystems and mechanisms employed in the implemented solution.

And lastly, both networks (WSN and cellular) have been subjected to several tests in order to validate their proper operation and obtain some metrics regarding the system's performance. In addition, several configuration sets have been tested in order to find out the best operation modes of the system.

2 ENTOMATIC WSN

2.1 GENERAL CONSIDERATIONS

2.1.1 NETWORK TOPOLOGY

2.1.1.1 Logical Topology

The network global logical topology chosen for the ENTOMATIC project is a two-tier hierarchical topology, where upper tier nodes are used to process and send the collected information to the central servers while lower tier nodes are used to perform the sensing tasks in the proximity of the target.

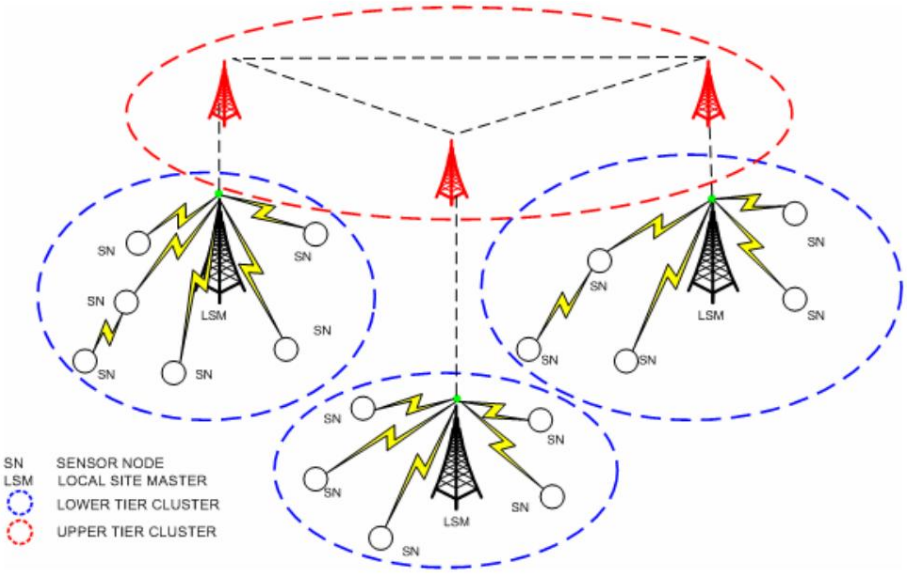


Figure 2.1: Illustration of Two-tier network topology [1]

As for the way the lower tier nodes transmit its data to the gateway within a WSN, there are different topology options: point-to-point, bus, star, ring or circular, mesh, tree, hybrid, or daisy chain. In the current project, the lower tier nodes will use a ***mesh*** network, where messages can be transmitted through different paths from a source to a destination.

Mesh networks provide high reliability, as messages can still be successfully transmitted through alternative routes in case any of the links or the nodes experience problems in terms of processing or connection.

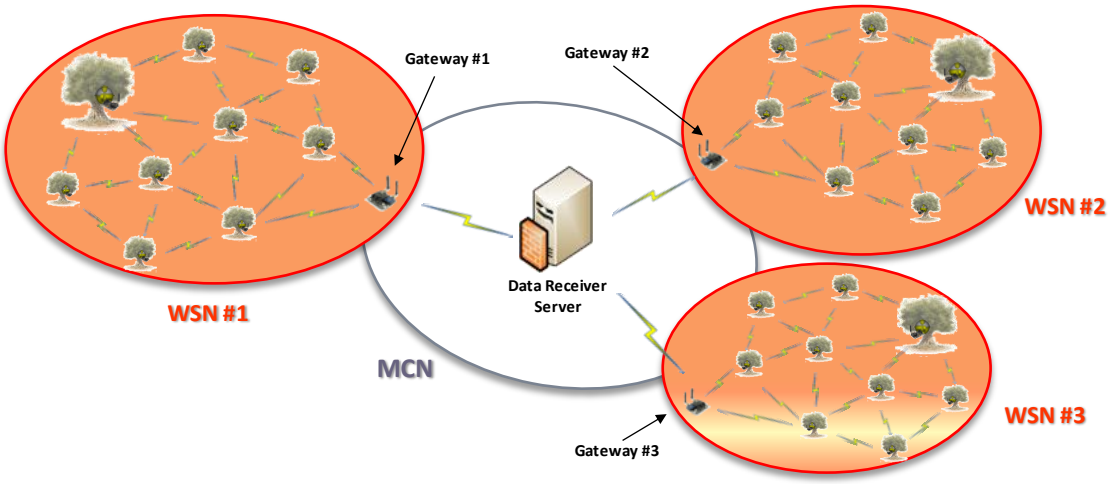


Figure 2.2: ENTOMATIC Logical topology

Lastly, the link between the upper tier nodes and the central servers will be performed through a point-to-point connection by using a cellular technology.

2.1.1.2 Physical Topology

The physical network topology refers to the physical layout of the machines, network devices and cabling. Due to the fact that the ENTOMATIC network is completely wireless, the physical topology is only referred to the physical arrangements of the different network components.

Figure 2.3: Possible deployment of ad-hoc wireless embedded network for precision agriculture [2]

As shown in Figure 2.3, nodes scattered throughout a field assemble together, establishing a routing topology and transmitting data back to a collection point. This will be the same in the ENTOMATIC communication, with a gateway located in a place with a good source of energy (like the country house in the image above) and some nodes located in the selected area to monitor the presence of olive flies.

The different nodes scattered on the field will be classified depending on their distance to the gateway, so that the full coverage of the network can be split into *rings* (or areas), each of them containing some of these nodes. In addition, if it were necessary, the placement of some *relay stations* (relay STA) has been considered to give network coverage to areas where properly service cannot be ensured due to propagation problems or high density of nodes.

The rings in which the full coverage of the network is split are also used to determine the number of hops a message must take before reaching its final destination, i.e., the gateway. Thus, as it can be seen

in Figure 2.4, nodes are only able to communicate with other nodes of their same, their lower or their upper ring, as long as they can be reachable with their power transmission level.

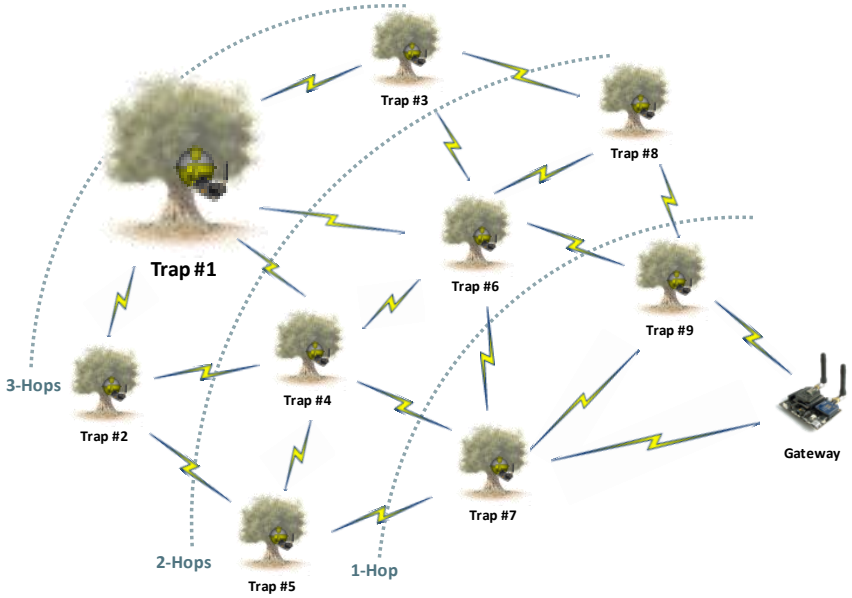


Figure 2.4: Detail of the physical topology for the ENTOMATIC project

In fact, it is advisable that radio transceivers have programmable transmit power control so that only the minimum required power is used when transmitting data. This would also reduce interference between nodes.

2.1.2 SYSTEM ASSUMPTIONS

- The gateway is always plugged to an energy source (or, at least, it is provided with greater batteries than other nodes) and transmits at maximum power
- The coverage area of the network is determined by the GW emitting at maximum power. Thus, any STA inside the coverage area of the GW is supposed to be able to communicate to the GW via single-hop
- There is a maximum of 30 STAs associated per GW
- Each STA has a different identifier (ID)
- In order to avoid congestion issues, an intermediate node can be used as relay only for a pre-determined number of stations
- A power regulation mechanism is implemented, so that each pair of stations forming a link are constantly tuning their corresponding transmission power level in order to save energy whenever it is possible
- Due to the implementation of headers and padding mechanisms in the physical layer, actual packet size transmitted over the air can be higher than net payload produced by stations
- Application is not considered as delay-critical

2.2 HARDWARE PLATFORM

The hardware used as sensor nodes (also known as ‘motes’) has a USB interface to be easily connected to a computer to be configured or to display received data in a hyper Linux or Windows. Motes endow sensors with processing and communication abilities, so that they do not only take environmental data, but also send the information to the base station. Typical components of these devices are:

- A CPU
- Flash memory

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

For this project, 2 different families of motes have been considered, the first one belonging to the frequency band 2.4 GHz (Crossbow TelosB) and the second one belonging to the band of 868 MHz (Zolertia Re-Mote). Although the system operation of both is very similar, the great advantage of Zolertia Re-Mote (besides being a much more current technology - October 2015), is the use of a lower frequency to transmit messages, allowing the system to reach much greater distances with fairly the same energy expenditure.

Although the Crossbow TelosB platform is much more old than the Zolertia Re-Mote, we chose this technology for performing the first platform tests due to its high availability during the first months of the execution period (Zolertia Re-Mote was released on October 2015 and we received the first units on December 2015). As it will be explained in following sections, Crossbow TelosB became very useful to validate most of our new communication mechanisms.

Figure 2.5: Crossbow TelosB [1]

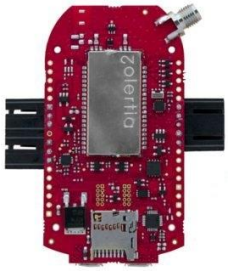


Figure 2.6: Zolertia Re-Mote [2]

2.2.1 CROSSBOW TELOSB

Telos is an ultra low power wireless module for use in sensor networks, monitoring applications, and rapid application prototyping. Telos leverages industry standards like USB and IEEE 802.15.4 to interoperate seamlessly with other devices.

By using industry standards, integrating humidity, temperature, and light sensors, and providing flexible interconnection with peripherals, Telos enables a wide range of mesh network applications. Telos Revision B is a drop-in replacement for Moteiv's successful Revision A design. Revision B includes increased performance, functionality, and expansion.

With Contiki OS or Tiny OS support out-of-the-box, Telos leverages emerging wireless protocols and the open source software movement. Telos is part of a line of modules featuring on-board sensors to increase robustness while decreasing cost and package size.

[1] Crossbow TelosB datasheet: http://www.willow.co.uk/TelosB_Datasheet.pdf
[2] Zolertia Re-Mote datasheet: http://zolertia.io/product/hardware/re-mote

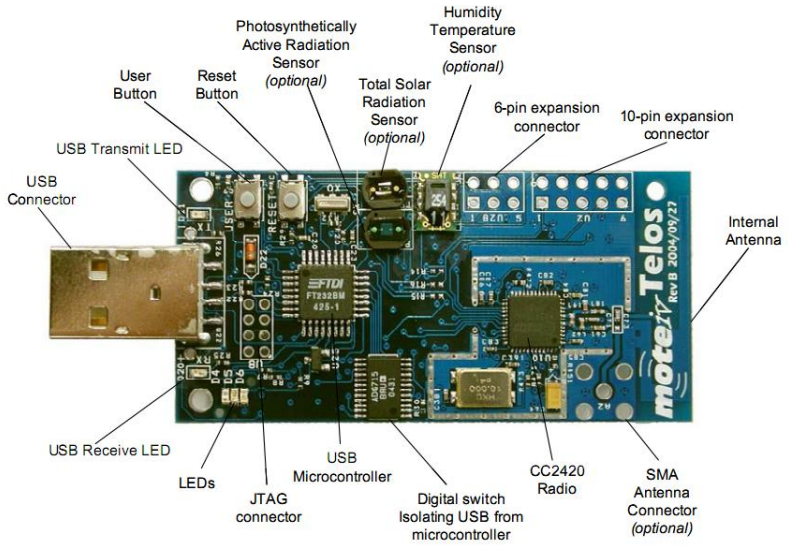


Figure 2.7: Front view of the Crossbow TelosB mote

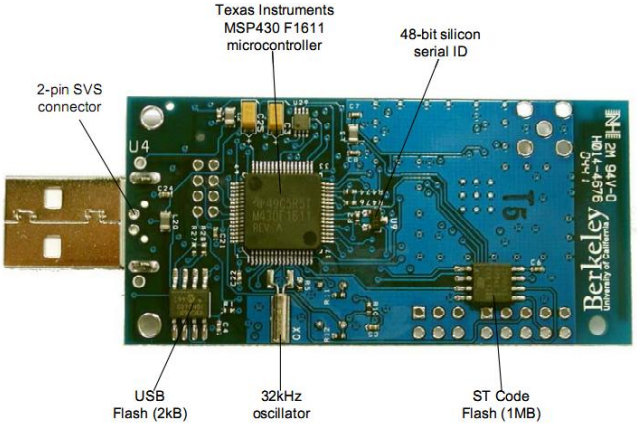


Figure 2.8: Back view of the Crossbow TelosB mote

- **Key Features**
  - 250kbps 2.4GHz IEEE 802.15.4 Chipcon Wireless Transceiver
  - Interoperability with other IEEE 802.15.4 devices
  - 8MHz Texas Instruments MSP430 microcontroller (10k RAM, 48k Flash)
  - Integrated ADC, DAC, Supply Voltage Supervisor, and DMA Controller
  - Integrated onboard antenna with 50m range indoors / 125m range outdoors
  - Integrated Humidity, Temperature, and Light sensors
  - Ultra low current consumption
  - Fast wakeup from sleep (<6µs)
  - Hardware link-layer encryption and authentication
  - Programming and data collection via USB
  - 16-pin expansion support and optional SMA antenna connector
  - TinyOS support : mesh networking and communication implementation

2.2.2 ZOLERTIA RE-MOTE

Re-Mote is a powerful development board to build real IoT projects and solutions. It can be considered as an Ultra-Low Power Wireless platform for 2.4 GHz and 863-950 MHz IEEE 802.15.4, 6LoWPAN, and ZigBee Applications.

Re-Mote IoT hardware board was developed jointly with universities and industrial partners from different countries in the context of of the European research project ***RERUM (REliable, Resilient and secUre IoT for sMart city applications)*** (https://ict-rerum.eu/) [3] to create powerful IoT hardware for Smart cities, logistics, lighting and industrial project. Fully compatible with main IoT operation systems, RE-Mote is the perfect hardware platform to create and work in real IoT.

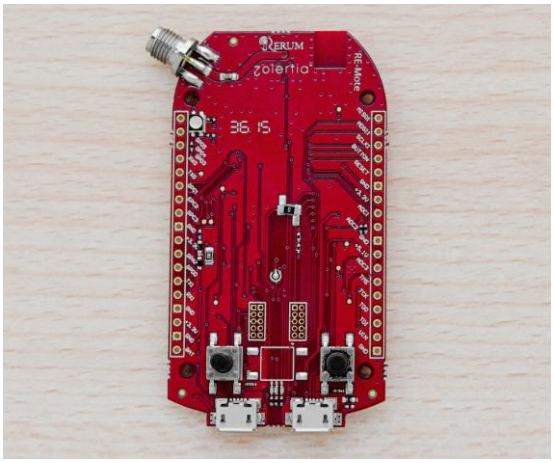


Figure 2.9: Zolertia Re-Mote front view

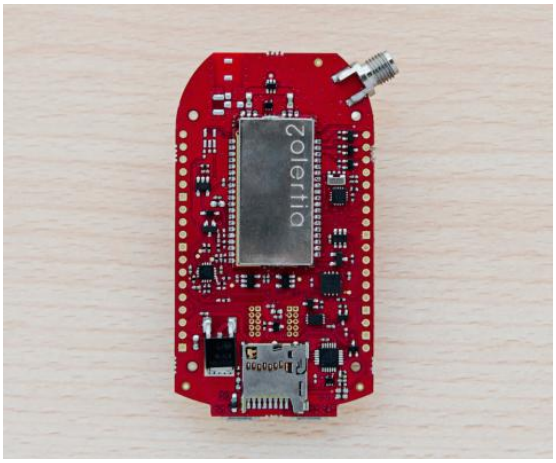


Figure 2.10: Zolertia Re-Mote back view

- **Microcontroller**
  - Powerful ARM® Cortex® -M3
  - 32-MHz Clock Speed
  - 512KB In-SystemProgrammable Flash
  - 32KB of RAM (16KB With Retention)
  - USB 2.0 Full-Speed Device (12 Mbps)
  - Security Hardware acceleration (AES-128/256, SHA2, ECC-128/256, RSA Hardware Acceleration Engine for Secure Key Exchange)
  - 1 x I2C, 1 x SPI, 1 x UART, µDMA, 12-Bit ADC with configurable resolution
  - General Purpose Pin (GPIO) 4/20mA

- **Features**
  - Micro-SD support
  - Shutdown Mode (190nA)
  - Real Time Clock Calendar (RTCC)
  - External Watchdog Timer and battery monitor (optional)
  - Built-in LiPo Battery Charger
  - RF switch to programatically drive either 2.4 GHz or Sub-1GHz RF interface to RP-SMA connector

- **RF 2.4 GHz**
  - ISM 2.4 GHz IEEE 802.15.4 & Zigbee/Thread compliant
  - 2394-2507 MHz frequency range with 1MHz/5MHz programable steps
  - Sensitivity -97 dBm, ACR 44 dB, up to 7 dBm transmission power
  - Data Rate 250 Kbps with DSSS Modulation

- **RF Sub-1 GHz**
  - ISM 863-868, 915-, 920-, 950 MHz ISM/SRD Band, IEEE 802.15.4g compliant
  - Sensitivity -109 dBm (50Kbps), down to -123 dBm (1.2 Kbps), ACR up to 60 dB, up to +16 dBm transmission power
  - Channels from 12.5 KHz - 1.66 MHz
  - Supported modulations 2-FSK, 2-GFSK, 4-FSK, 4-GFSK, MSK, OOK
  - Supports Data Rate Up to 1.25 Mbps in Transmit and Receive Modes

- **Operational values**

Table 2.1: Zolertia Re-Mote operational values

Parameter	Minimum	Average	Maximum	Unit
Operation supply voltage	3.4	4.7	16	V
General purpose pin voltage output	0	3.3	-	V
General purpose pin current output	0	4	20	mA
Shutdown mode	150	-	-	nA
PM3 power mode	-	0.4	-	µA
PM1 power mode	-	0.6	-	mA
Active mode (2.4 GHz RX, CPU idle)	-	20	-	mA
Active mode (2.4 GHz TX, 0 dBm, CPU idle)	-	24	-	mA
Operation Temperature	-40	25	100	ºC

- **Pin-out distribution and components**
  - Front view

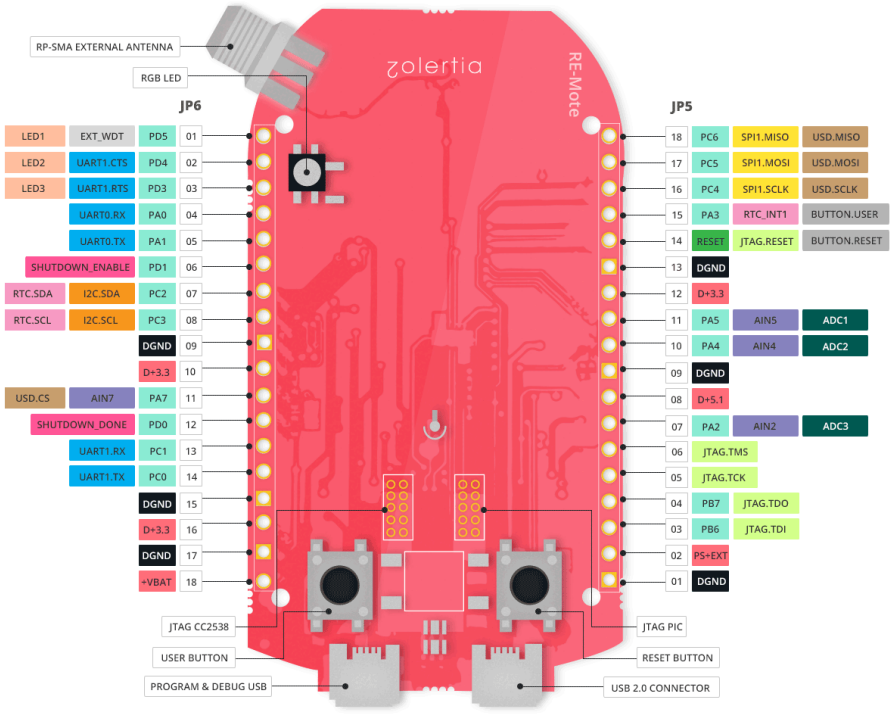


Figure 2.11: Zolertia RE-Mote's pin-out of the front view

  - Back view

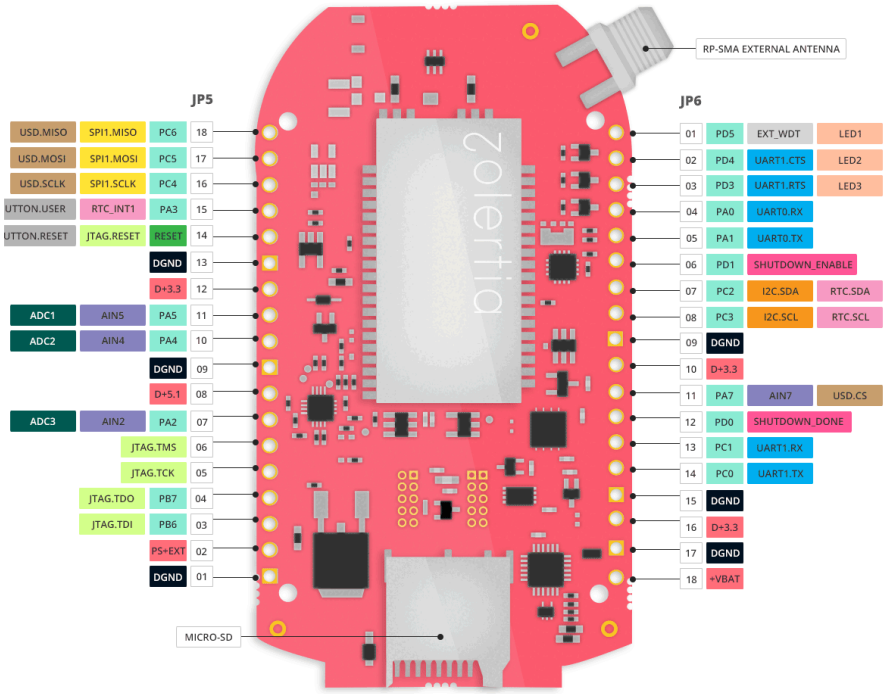


Figure 2.12: Zolertia RE-Mote's pin-out of the back view

- **Compliances**
  - Europe: ETSI EN 300 220, ETSI EN 54-25, EN 300 328, EN 300 440.
  - US: FCC CFR47 Part 15, FCC CFR47 Part 90, 24, 101.
  - Japan: ARIB RCR STD-T30, ARIB STD-T66, ARIB STD-T67, ARIB STD-T108

- **Development tools**
  - Contiki OS: contiki-os.org
  - RIOT OS: www.riot-os.org
  - Thingsquare Platform
  - OpenWSN (upcoming)
  - Eclipse IDE for C/C++
  - Code Composer Studio™
  - IAR Embedded Workbench ® for ARM
  - SmartRF™ Studio
  - SmartRF Flash Programmer
  - Built-in programming support over USB

- **Zolertia resources**
  - Zolertia site: www.zolertia.io
  - Zolertia Online Store: zolertia.io/store
  - Resource Page: www.zolertia.io/resources
  - Zolertia Github repository: github.com/Zolertia
  - Hackster Page: www.hackster.io/zolertia

2.3 SOFTWARE PLATFORM

The Operating System (OS) for a Wireless Sensor Network is typically less complex than general-purpose operating systems. The main tasks performed by a WSN OS are to manage the access to different node resources (processors, memories, timers, disks, network interfaces, etc), and to support for concurrent execution of processes.

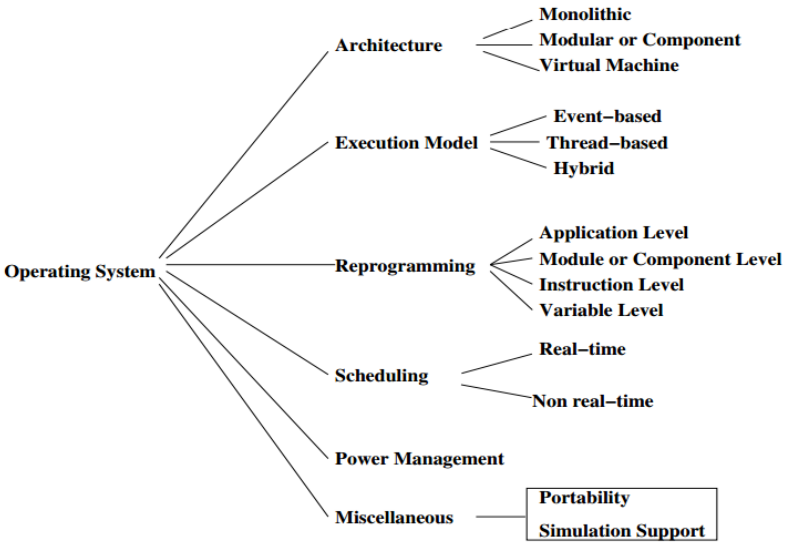


Figure 2.13: Classification framework for WSN Operating Systems [4]

The main features that characterize Operating Systems for Wireless Sensor Networks are [5]:

- **Architecture**
  The organization of an OS constitutes its structure. The architecture of an OS has an influence on the size of the OS kernel as well as on the way it provides services to the application programs. The kernel can either be built in a monolithic fashion, follow a layered approach, or implement the microkernel architecture.

- **Programming Model**
  The programming model supported by an OS has a significant impact on the application development, by defining whether all tasks are executed within the same context and have no segmentation of the memory address space, or every process can run in its own thread and has its own memory stack. The programming model is also linked to the available programming languages for application developers.

- **Scheduling**
  The Central Processing Unit (CPU) scheduling determines the order in which tasks are executed on a CPU. In traditional computer systems, the goal of a scheduler is to minimize latency, to maximize throughput and resource utilization, and to ensure fairness.

  The selection of an appropriate scheduling algorithm for WSNs typically depends on the nature of the application. For applications having real-time requirements, real-time scheduling algorithm must be used. For other applications, non-real-time scheduling algorithms are sufficient.

- **Memory management and Protection**
  Memory management refers to the strategy used to allocate and de-allocate memory for different processes threads.

- **Communication Protocol Support**
  In the OS context, communication refers to inter-process communication within the system as well as with other nodes in the network.

- **Resource Sharing**
  The responsibility of an OS includes allocation and resource sharing, which is of immense importance when multiple programs are concurrently executing. The majority of WSNs OSs

today provide some sort of multithreading, requiring a resource sharing mechanism. This can be performed:

a) In time
   E.g., scheduling of a process/thread on the CPU
b) In space
   E.g., writing data to system memory

2.3.1 OPERATING SYSTEMS FOR WSNS

2.3.1.1 Contiki

Contiki is a multitasking, open source, highly portable operating system written entirely in C and developed by Adam Dunkels and the group of "Networked Embedded Systems" of the Swedish Institute of Computer Science (SICS) [6].

It is designed for embedded systems with limited memory resources, so that a typical configuration for TCP/IP applications can take about 2 kB of RAM and 40KB of ROM (including 802.15.4 physical and link layer, 6LoWPAN with fragmentation and header compression, uIPv6, support for TCP and UDP in the transport layer, and the Contiki operating system itself). Included in Contiki, there is a range of applications such as a HTTP, Constrained Application Protocol (CoAP), FTP, and DHCP servers, as well as other useful programs and tools.

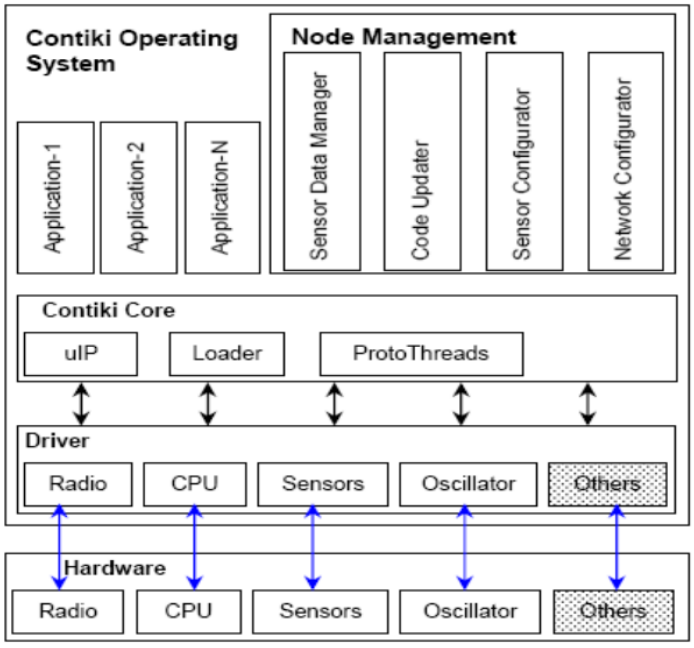


Figure 2.14: Contiki Architecture [5]

A Contiki operating system consists of a kernel, a set of applications, a program loader and a set of processes. A process can be an application program or a service, defining the latter as an agent that provides the functionality needed for more than one application process. All processes, both the application program and services, can be dynamically replaced during the system runtime. Communication between these processes always occurs through kernel. In Contiki, the kernel does not provide the system hardware abstraction layer, but allows device drivers and applications to communicate directly with the hardware.

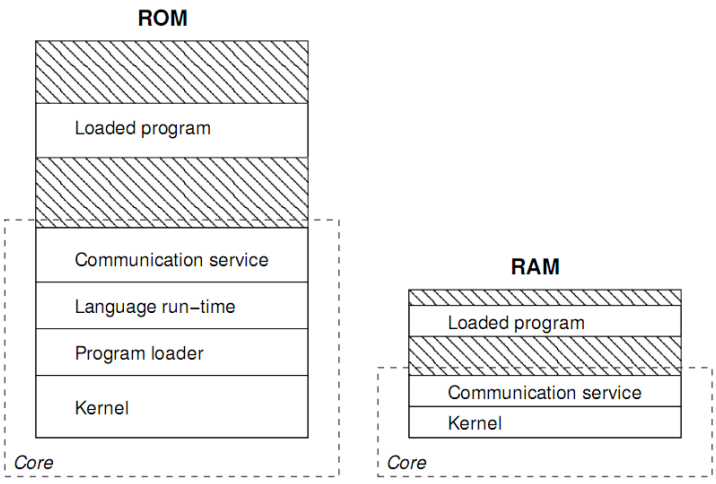


Figure 2.15: ROM and RAM structure in Contiki OS [6]

As shown in Figure 2.15, the Contiki operating system is divided into two distinct parts: the kernel and the loaded programs. The kernel is compiled into a single binary image that is loaded in devices prior to their deployment.

Unlike other operating systems, in which a multi-threading model is used, Contiki's kernel proposes a model based on events. Therefore, it is not necessary to create a data stack for each generated thread or to include locking mechanisms, typical in systems based on multithreading. Thus implies a great saving in memory.

In event-based systems, processes are implemented as routines that are entirely executed. All processes running on Contiki share the same stack, thus optimizing scarce resources. Furthermore, locking mechanisms are normally not needed, since two events routines are never executed concurrently.

However, not all programs are readily express them in the form of state machines, as advocated in the event-based systems. In addition, an event-based system, applications requiring high computing effort monopolize the CPU, disregarding possible external events.

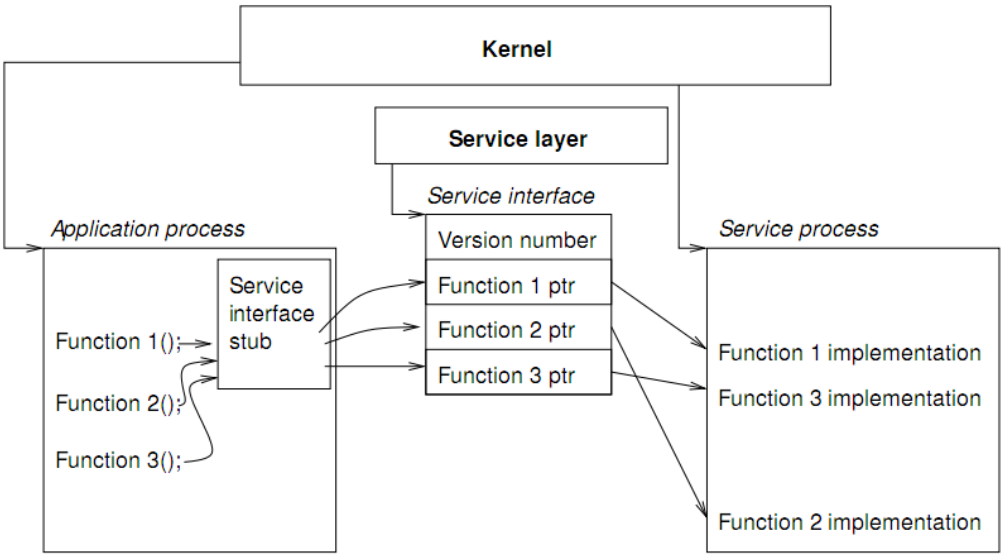


Figure 2.16: An application function calling a service in Contiki OS

The proposal by Contiki solution is based on a hybrid system based on a kernel of events where multi-threading is implemented as an application library that can be linked by programs that require it. The

threads involved in this multi-threading are called *protothreads*; these threads are lighter than traditional and, as already stated, are executed in the same stack, requiring only 2 bytes of memory each. In addition, *protothreads* are implemented in C and do not require specific assembly code for each device.

Table 2.2: Summary of Contiki's features

Feature	Description
Architecture	Modular architecture. Event-driven model (asynchronous and synchronous events) with optional threading facilities to individual processes
Programming Model	Preemptive multithreading by means of protothreads. A protothread is a memory-efficient programming abstraction that shares features of both multithreading and event-driven programming to attain a low memory overhead of each protothread
Scheduling	No scheduling; events are fired to the target application as they occur
Memory management and Protection	Dynamic memory management and dynamic linking of programs
Communication Protocol Support	Three communication stacks: µIP, µIPv6, and Rime
Resource Sharing	Serialized access to all resources
Support for real-time applications	No support for real-time applications
Additional features	Coffee file system, emulation support through Cooja

Contiki contains three communication stacks: µIP, µIPv6, and Rime. µIP is a protocol stack for small 8 bit micro-controllers with the minimum set of features needed for a full TCP/IP usage. µIP is written in C, it can only support one network interface, and it supports TCP, UDP, ICMP, and IP protocols. The µIPv6 stack is the first that meets the requirements to enable communication over the Internet with IPv6.

Rime, meanwhile, is a light communication stack designed for low power radio transmitters. Rime provides a wide range of communication possibilities, from broadcast local area 'best-effort' to flood reliable data in a multi-hop environment.

Table 2.3: Hardware platforms currently available in Contiki OS

Microcontroller / System on Chip	Radio	Platforms	Cooja simulation support
TI CC2538	Integrated / CC1200	Re-Mote	-
RL78	ADF7023	EVAL-ADF7023DB1	-
TI CC2538	Integrated	cc2538dk	-
TI MSP430x	TI CC2420	exp5438, z1	Yes
TI MSP430x	TI CC2520	wismote	Yes
Atmel AVR	Atmel RF230	avr-raven, avr-rcb, avr-zigbit, iris	-
Atmel AVR	TI CC2420	micaz	Yes
Freescale MC1322x	Integrated	redbee-dev, redbee-econotag	-
TI MSP430	TI CC2420	sky	Yes
TI MSP430	TI CC1020	msb430	-
TI MSP430	RFM TR1001	esb	Yes
Atmel Atmega128 RFA1	Integrated	avr-atmega128rfa	-
Microchip pic32mx795f512l	Microchip mrf24j40	seed-eye	-
TI CC2530	Integrated	cc2530dk	-
6502	-	apple2enh, atari, c128, c64	-
Native	-	native, minimal-net, cooja	Yes

Cooja emulator

Cooja is a Java-based emulator designed to simulate sensor networks using the Contiki operating system, one of the most widely used in the field of the Internet of Things [7]. Cooja can emulate heterogeneous WSNs with different types of nodes (both in software and hardware). At this point, it is worth noting the definition of an *emulator,* unlike the one of a *simulator:*

- **Emulator:** A hardware or software system that enables one computer system (called the host) to behave like another computer system (called the guest): e.g. Cooja enabling your laptop to behave like a Z1 mote.
- **Simulator:** A system designed to recreate the operation or behaviour of the guest system. The underlying principles can be the same as the original or different.

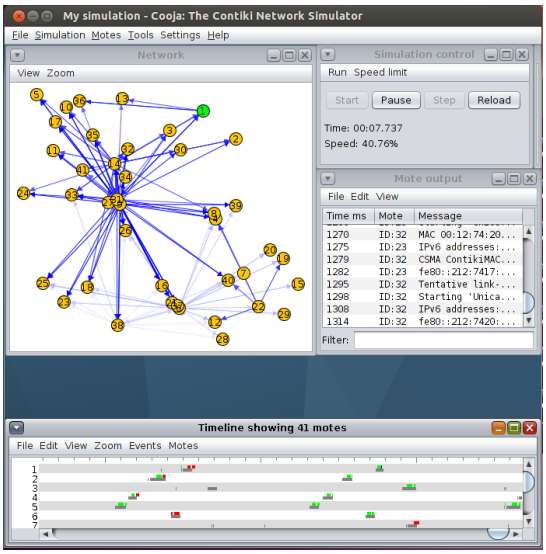


Figure 2.17: COOJA's user interface

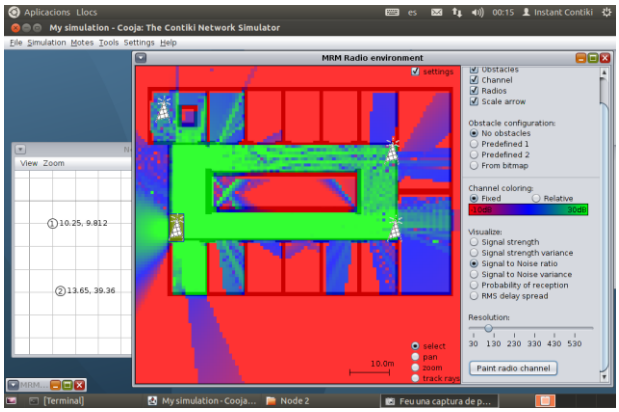


Figure 2.18: COOJA's interface for simulations of physical parameters of transmissions

COOJA is flexible and many parts of the simulator can be easily replaced or extended with additional functionality. Example parts that can be extended include the simulated radio medium, simulated node hardware, and plug-ins for simulated input/output. A simulated node in COOJA has three basic properties: its data memory, the node type, and its hardware peripherals.

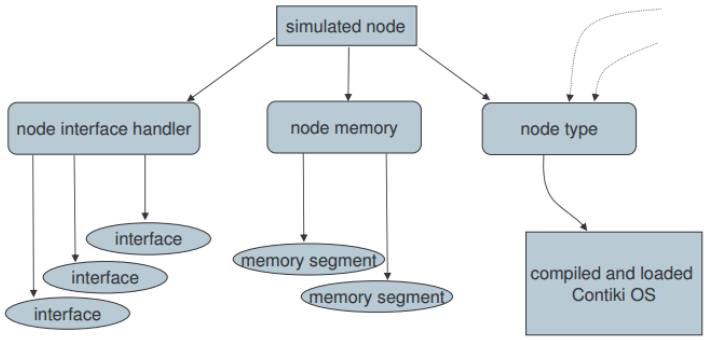


Figure 2.19: A simulated sensor node in Cooja

Simplified, a Cooja simulation consists of a number of nodes being simulated. Each node is classified under its type. The node type is the bridge between the physical node and a loaded Contiki OS executing some specific code. All nodes of the same type are linked to the same loaded Contiki OS. When the simulation is running, all nodes start acting in turn. Once the simulation time is updated and all nodes have acted, then the process is repeated (the simulation loop) [8].

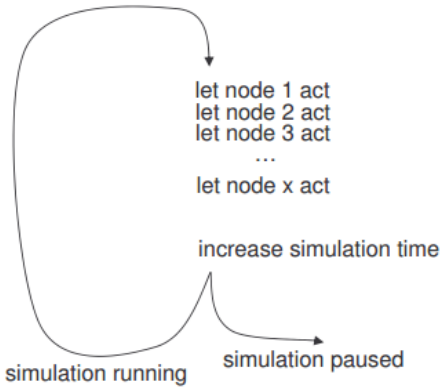


Figure 2.20: Simulation loop of Cooja

The memory consists of one or several memory segments, each with a start address and data. The combination of memory segments defines all parts of an entire simulated Contiki OS. The interfaces act on the memory and simulate node devices such as a clock or a radio transmitter. For instance, when time changes a clock interface updates some specific time variable.

The hardware peripherals of simulated nodes are called interfaces, and enable the Java simulator to detect and trigger events such as incoming radio traffic or a LED being lit. Interfaces also represent properties of simulated nodes such as positions that the actual node is not aware of.

Cross-level simulation

Cooja allows for simultaneous simulations at three different levels, namely the networking (or application) level, the operating system level and the machine code instruction level [9]. Thus, nodes from different levels can coexist and interact on the same simulation allowing, for example, an emulated node could send a package radius based on a Java node.

- **Networking level**
  COOJA supports code development by enabling the user to easily exchange certain simulator modules such as device drivers or radio medium modules. A simulation can be saved and reloaded using other more or less detailed modules, still with the other simulation parameters unaltered. Furthermore, new radio mediums and interfaces such as radio devices can easily be developed in Java and be added to the COOJA simulation environment.

- **Operating system level**
  Cooja simulates the operating system by executing native operating system code as described in the previous section. As the entire Contiki OS, including any user processes, is executed it is also possible to alter Contiki core functionality. This is useful for example to test and evaluate changes in included Contiki libraries.

- **Machine code instruction set level**
  Using COOJA, it is possible to create new nodes with a very different underlying structure than the typical nodes.

Radio models

Each simulation in COOJA uses a radio model that characterizes radio wave propagation. New radio models may be added to the simulation environment. The radio model is chosen when a simulation is created. This enables a user to, for example, develop a network protocol using a simple radio model, and then testing it using a more realistic model, or even a custom made model to test the protocol in very specific network conditions. Often a radio model provides one or several plug-ins in order to configure and view the current simulated network conditions. COOJA supports, except from a completely silent model, a simple model that uses interference and a transmission range parameter that can be changed during a simulation run.

Graphical interface

Another stronghold of this simulator is its powerful graphical interface aimed at obtaining physical parameters of transmissions (received signal strength, SNR, reception probability, etc) simulations. This fact, together with the definition of a MRM (Multi-path Ray-tracer Medium) model, provides maps where the parameters above mentioned can be easily visualized depending on the selected input variables (type of hardware nodes, output signal power, presence and type of obstacles, etc).

2.3.1.2 Tiny OS

TinyOS [10] is a component-based operating system and platform targeting wireless sensor networks. TinyOS is an embedded operating system written in the nesC programming language (a variant of C) as a set of cooperating tasks and processes. nesC does not have any dynamic memory allocation and all program paths are available at compile-time. This is manageable thanks to the structure of the language; it uses modules and interfaces instead of functions.

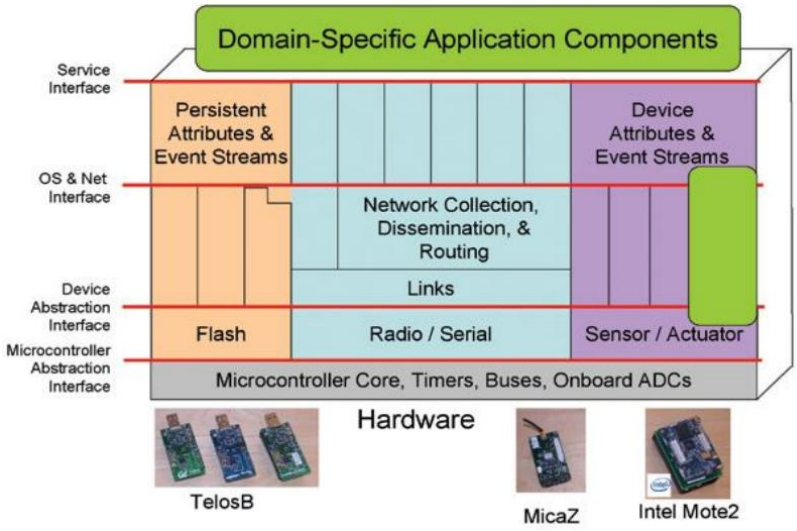


Figure 2.21: Tiny OS architecture

The modules use and provide interfaces and are interconnected with configurations; this procedure makes up the structure of the program. Multitasking is implemented in two ways: trough tasks and events. Tasks, which focus on computation, are non-preemptive, and run until completion. In contrast, events which focus on external events i.e. interrupts, are preemptive, and have separate start and stop functions.

Table 2.4: Hardware platforms currently available in Tiny OS

Hardware	Supported platforms
Platform	EPIC Imote2 Shimmer IRIS Telos Rev B MicaZ Mica2 Mica2dot NXTMOTE - TinyOS on LEGO MINDSTORMS NXT Mulle TinyNode Zolertia Z1 UCMote Mini
Microcontroller	Atmel ATmega128 Texas Instruments MSP430 Intel XScale PXA271

Radio	CC1000 CC1100/CC2500 CC2420 AT86RF212 AT86RF230

TinyOS programs are built out of software components, some of which present hardware abstractions. Components are connected to each other using interfaces. TinyOS provides interfaces and components for common abstractions such as packet communication, routing, sensing, actuation and storage. The OS has full support for both 6LoWPAN and RPL, and also have libraries for CoAP.

Table 2.5: Summary of Tiny OS' features

Feature	Description
Architecture	Monolithic architecture.
Programming Model	Primarily event-driven, support for TinyOS threads has been added
Scheduling	FIFO (First-In-First-Out)
Memory management and Protection	Static Memory Management with memory protection
Communication Protocol Support	Active Message
Resource Sharing	Virtualization and completion events
Support for real-time applications	No support for real-time applications
Additional features	TOSSIM simulator

TOSSIM simulator

TOSSIM simulator [11] is a discrete event simulator for TinyOS sensor networks. Instead of compiling a TinyOS application for a mote, users can compile it into the TOSSIM framework, which runs on a PC. This allows users to debug, test, and analyze algorithms in a controlled and repeatable environment. As TOSSIM runs on a PC, users can examine their TinyOS code using debuggers and other development tools. In the following lines its main characteristics are detailed:

- **Fidelity:** By default, TOSSIM captures TinyOS' behavior at a very low level. It simulates the network at the bit level, simulates each individual ADC capture, and every interrupt in the system.

- **Time:** While TOSSIM precisely times interrupts (allowing things like bit-level radio simulation), it does not model execution time. From TOSSIM's perspective, a piece of code runs instantaneously. This also means that spin locks or task spin locks will never exit: as the code runs instantaneously, the event that would allow the spin to stop will not occur until the code completes (never).

- **Models:** TOSSIM itself does not model the real world. Instead, it provides abstractions of certain real-world phenomena. With tools outside the simulation itself, users can then manipulate these abstractions to implement whatever models they want to use. By making complex models exterior to the simulation, TOSSIM remains flexible to the needs of many users without trying to establish what is "correct." Additionally, it keeps the simulation simple and efficient.

    - Radio: TOSSIM does not model radio propagation; instead, it provides a radio abstraction of directed independent bit errors between two nodes.

    - Power/Energy: TOSSIM does not model power draw or energy consumption. Besides, as TOSSIM does not model CPU execution time, it cannot easily provide accurate information for calculating CPU energy consumption.

- **Building:** TOSSIM builds directly from TinyOS code. To simulate a protocol or system, you must write a TinyOS implementation of it. On one hand, this is often more difficult than an abstract

simulation; on the other, it means you can then take your implementation and run it on actual motes.

- **Imperfections:** Although TOSSIM captures TinyOS behavior at a very low level, it makes several simplifying assumptions. This means that it is very possible that code which runs in a simulation might not run on a real mote.

- **Networking:** Currently, TOSSIM simulates the 40Kbit RFM mica networking stack, including the MAC, encoding, timing, and synchronous acknowledgements. It does not simulate the mica2 ChipCon CC1000 stack.

- **Authority:** Initial experience from real-world deployments has shown that TinyOS networks have very complex and highly variable behavior. While TOSSIM is useful to get a sense of how algorithms perform in comparison to one another, TOSSIM results shouldn't be considered authoritative. TOSSIM should not be considered an end-point of evaluation; instead, it is a system that allows the user to separate out environmental noise to better understand algorithms.

2.3.1.3 FreeRTOS

FreeRTOS is a popular real-time operating system kernel for embedded devices that has been ported to 35 microcontrollers. It is distributed under the GPL (General Public License) with an additional restriction and optional exception. The restriction forbids benchmarking while the exception permits users' proprietary code to remain closed source while maintaining the kernel itself as open source, thereby facilitating the use of FreeRTOS in proprietary applications [12].

- Designed to be small, simple and easy to use.
- Free RTOS kernel - preemptive, cooperative and hybrid configuration options.
- Includes a tickless mode for low power applications.
- Official support for 35 embedded system architectures.
- Supports the ARM Cortex-M3 MPU.
- Very portable, predominantly written in C.
- Supports both real time tasks and co-routines.
- Mutexes with priority inheritance.
- Powerful execution trace functionality.
- Stack overflow detection options.
- No software restriction on the number of real time tasks that can be created.
- No software restriction on the number of task priorities that can be used.
- No restrictions imposed on task priority assignment - more than one real time task can be assigned the same priority.

FreeRTOS is designed to be small and simple. The kernel itself consists of only three or four C files. To make the code readable, easy to port, and maintainable, it is written mostly in C, but there are a few assembly functions included where needed (mostly in architecture-specific scheduler routines).

Table 2.6: Hardware platforms currently available in FreeRTOS

Hardware	Supported processor families	Supported tools
Actel (now Microsemi)	SmartFusion, SmartFusion2 - see Microsemi listing	IAR, Keil, SoftConsole (GCC with Eclipse)
Altera	Cyclone V SoC (ARM Cortex-A9), Nios II	Altera SoC EDS (ARM DS-5 with GCC), Nios II IDE with GCC
Atmel	SAMV7 (ARM Cortex-M7), SAM3 (ARM Cortex-M3), SAM4 (ARM Cortex-M4), SAMD20 (ARM Cortex-M0+), SAMA5 (ARM Cortex-A5), SAM7 (ARM7), SAM9 (ARM9), AT91, AVR, AVR32 UC3	IAR, GCC, Keil, Rowley CrossWorks
Cortus	APS3	Cortus IDE with GCC

Cypress	PSoC 5 ARM Cortex-M3	GCC, ARM Keil and RVDS - all in the PSoC Creator IDE
Freescale	Kinetis ARM Cortex-M4, Coldfire V2, Coldfire V1, other Coldfire families, HCS12, PPC405 & PPC440 (Xilinx implementations) (small and banked memory models), plus contributed ports	Codewarrior, GCC, Eclipse, IAR
Infineon	TriCore, XMC4000 (ARM Cortex-M4F), XMC1000 (ARM Cortex-M0)	GCC, Keil, Tasking, IAR
Fujitsu (Now Spansion)	FM3 ARM Cortex-M3, 32bit (for example MB91460) and 16bit (for example MB96340 16FX)	Softune, IAR, Keil
Luminary Micro / Texas Instruments	All Luminary Micro ARM Cortex-M3 and ARM Cortex-M4 based Stellaris microcontrollers	Keil, IAR, Code Red, CodeSourcery GCC, Rowley CrossWorks
Microchip	PIC32MX, PIC32MZ, CEC13xx, PIC32MZ EF, PIC24, PIC24EP, dsPIC	MPLAB C32, MPLAB C30
Microsemi	SmartFusion, SmartFusion2	IAR, Keil, SoftConsole (GCC with Eclipse)
NEC (now Renesas)	V850 (32bit), 78K0R (16bit)	IAR
NXP	LPC1500 (ARM Cortex-M3), LPC1700 (ARM Cortex-M3), LPC1800 (ARM Cortex-M3), LPC1100 (ARM Cortex-M0), LPC2000 (ARM7), LPC4000 (ARM Cortex-M4F/ ARM Cortex-M0)	GCC, Rowley CrossWorks, IAR, Keil, LPCXpresso IDE, Eclipse
Renesas	RZ/A1 (ARM Cortex-A9), RX700 / RX71M, RX600 / RX64M / RX62N / RX63N, RX200, RX100, SuperH, RL78, H8/S plus contributed ports	GCC, HEW (High Performance Embedded Workbench), IAR Embedded Workbench
Silicon Labs [ex Energy Micro]	EFM32 Gecko (Cortex-M3 and Cortex-M4F), 8051 compatible microcontrollers.	Simplicity Studio (GCC), IAR, SDCC
Spansion	FM3 ARM Cortex-M3, 32bit (for example MB91460) and 16bit (for example MB96340 16FX)	Softune, IAR, Keil
ST	STM32 (ARM Cortex-M0, ARM Cortex-M7, ARM Cortex-M3 and ARM Cortex-M4F), STR7 (ARM7), STR9 (ARM9)	IAR, Atollic TrueStudio, GCC, Keil, Rowley CrossWorks
Texas Instruments	RM48, TMS570, ARM Cortex-M4F MSP432, MSP430, MSP430X, Stellaris (ARM Cortex-M3, ARM Cortex-M4F)	Rowley CrossWorks, IAR, GCC, Code Composer Studio
Xilinx	Zynq, Zynq UltraScale+ MPSoC (64-bit ARM Cortex-A53 and 32-bit ARM Cortex-R5), Microblaze, PPC405 running on a Virtex4 FPGA, PPC440 running on a Virtex5 FPGA.	GCC
Intel/x86	IA32 (32-bit flat memory model), Quark SoC X1000 (32-bit flat memory model), any x86 compatible running in Real mode only, plus a Win32 port	GCC, Visual Studio 2010 Express, MingW, Open Watcom, Borland, Paradigm

FreeRTOS provides methods for multiple threads or tasks, semaphores and software timers. A tick-less mode is provided for low power applications. Thread priorities are supported by having the host program a thread tick method at regular short intervals. The thread tick method switches tasks depending on priority and a round-robin scheduling scheme. The usual interval is 1/1000 of a second to 1/100 of a second, via an interrupt from a hardware timer, but this interval is often changed to suit a particular application.

2.3.1.4 RIOT

RIOT [13] is a real-time multi-threading operating system that explicitly considers devices with minimal resources but eases development across the wide range of devices that are typically found in the Internet of Things. The kernel is written in C but the upper layers support C++ as well. As the project fits with real-time and reliability requirements, the kernel supports hard real-time multi-tasking scheduling.

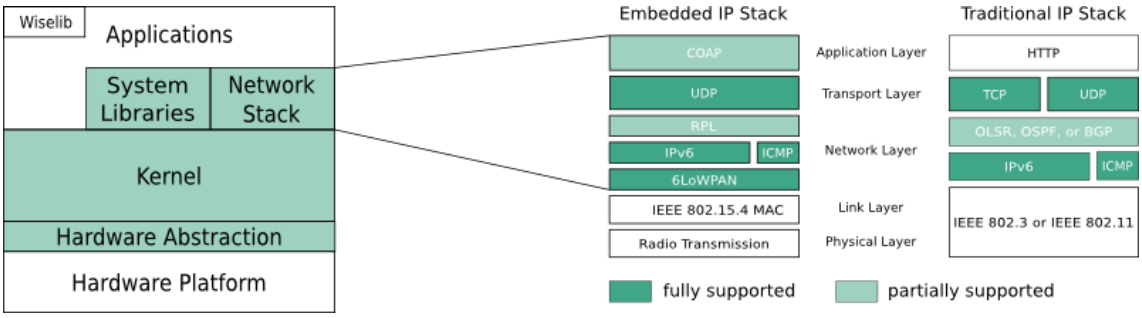


Figure 2.22: RIOT structure

RIOT is based on design objectives including energy-efficiency, reliability, real-time capabilities, small memory footprint, modularity, and uniform API access, independent of the underlying hardware (this API offers partial POSIX compliance). Several libraries (e.g. Wiselib) are already available on RIOT, as well as a full IPv6 network protocol stack including the latest standards of the IETF for connecting constrained systems to the Internet (6LoWPAN, IPv6, RPL, TCP and UDP).

Table 2.7: Hardware platforms currently available in RIOT

Board	Microcontroller	Radio	Family	Vendor	Status
native	Linux/OSX on x86	Ethernet/tap	native	n/a	full support
IoT-LAB M3 node (FIT)	STM32F103RE	Atmel AT86RF231	ARM Cortex-M3	STM	partly supported
Samr21-xpro	ATSAMR21G18A	Atmel AT86RF233	ARM-Cortex-M0	Atmel	full support
arduino-due	SAM3X8E	n/a	ARM Cortex-M3	Atmel	full support
HiKoB Fox	STM32F103RE	Atmel AT86RF231	ARM Cortex-M3	STM	full support
UDOO	SAM3X8E	n/a	ARM Cortex-M3	Atmel	basic support
CC2538DK	CC2538	n/a	ARM Cortex-M3	TI	basic support
OpenMote	CC2538	n/a	ARM Cortex-M3	TI	basic support
stm32f0discovery	STM32F051R8	n/a	ARM Cortex-M0	STM	good support
stm32f3discovery	STM32F303VC	n/a	ARM Cortex-M4	STM	good support
stm32f4discovery	STM32F407VG	n/a	ARM Cortex-M4	STM	good support
pca10000	NRF51822QFAA	BLE	ARM-Cortex-M0	Nordic	radio support missing
pca10005	NRF51822QFAA	BLE	ARM-Cortex-M0	Nordic	radio support missing
Mulle	MK60DN512VLL10	AT86RF212B	ARM Cortex-M4	Freescale	LPM in progress
PhyWAVE Board pba-d-01-kw2x	MKW2x family	internal	ARM Cortex-M4	Freescale	LPM in progress
yunjia-nrf51822	NRF51822QFAA	BLE	ARM-Cortex-M0	Nordic	radio support missing
Arduino Mega2560	ATmega2560	n/a	AVR/ATmega	Atmel	LPM/SPI missing
STM32 Nucleo-F091	STM32F091RC	n/a	ARM Cortex-M4	STM	LPM missing
STM32 Nucleo-F334	STM32F334R8	n/a	ARM Cortex-M4	STM	LPM missing
msb-430	MSP430x16x	cc1020	MSP430	TI	partly supported
msb-430h	MSP430x16x	cc1100	MSP430	TI	basic support
chronos	CC430	cc1100	MSP430	TI	basic support
telosb	MSP430x16x	cc2420	MSP430	TI	basic support
wsn430-v1_3b	MSP430x16x	cc1100	MSP430	TI	basic support
wsn430-v1_4	MSP430x16x	cc2420	MSP430	TI	basic support

Z1	MSP430x16x	cc2420	MSP430	TI	basic support
RE-Mote	CC2538	CC2538/CC1200	ARM Cortex-M3	TI	partly supported
avsextrem	LPC2387	cc1100	ARM7	NXP	baisc support
msba2	LPC2387	cc1100	ARM7	NXP	basic support
pttu	LPC2387	cc1100	ARM7	NXP	?
mbed_lpc1768	LPC1768	ethernet (on die)	ARM Cortex-M3	NXP	partly supported

2.3.2 SELECTION OF AN OPERATING SYSTEM

Among the vast quantity of current OS for WSN, only the four most known have been considered: Contiki, Tiny OS, FreeRTOS, and RIOT. As for the number of publications (in 2010), the percentage of articles related to each operating system included in the main scientific and engineering online databases (IEEE Xplore, ACM Digital Library and Science Direct) are the following: 81% TinyOS, 9% Contiki [14].

The main features of the Operating Systems have been summarized in Table 2.8, including their minimim requirements in terms of RAM and ROM, usage for a basic application, support for programming languages, multi-threading, MCUs without Memory Management Unit (MMU), modularity, real-time behaviour, and availability of a simulator.

Table 2.8: Key characteristics of Contiki, Tiny OS, FreeRTOS, and RIOT

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

Although it has the best scores in the analyzed parameters, RIOT suffers from a lack of availability of a simulator. This makes its election unsuitable for the ENTOMATIC project, as the complexity of the system and the great quantity of nodes per network requires preliminary and comprehensive simulations to complete the design of the different protocols.

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

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

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

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

availability of techniques offered to estimate energy consumption of sensor nodes, such as the *powertrace* [15] application of Contiki.

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

Table 2.9: Latest releases of the considered Operating Systems

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

Lastly, another factor which was considered before choosing the best platform was the degree of software content being up-to-date. As it can be seen in Table 2.9, the oldest current version is the one from Tiny OS, which dates from 2012. The rest of considered WSNs have rather up-to-date functional versions.

After analyzing all these factors, we have opted for the open source operating system Contiki OS [3] that gives developers and researchers flexibility to provide low-capable devices with novel communication mechanisms. Besides, its powerful emulator Cooja gives us a great deal of flexibility to design and test our custom protocols intended to improve the ENTOMATIC network performance as well as to minimize its power consumption.

2.4 PHYSICAL LAYER

The physical layer in the current project is based on the technology provided by the Texas Instruments CC2538 [4] microcontroller and the Texas Instruments CC1200 [5] transceiver embedded in the Zolertia Re-Mote devices [16], compatible with the IEEE 802.15.4 standard [17]. Despite the availability of a 2.4 GHz RF transceiver embedded in the TI CC2538 microcontroller, the current project has only used for communication purposes the 868 MHz RF transceiver included in the Zolertia Re-Mote board.

(At this point, it is worth noting here that Crossbow TelosB have been only used to validate network protocols and no deep analysis on energy consumption has been performed).

2.4.1 TI CC2538 MICROCONTROLLER

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

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

[3] Contiki Operating System main webpage: http://www.contiki-os.org/
[4] Texas Instruments – CC2538 - www.ti.com/lit/gpn/cc2538
[5] Texas Instruments – CC1200 - http://www.ti.com/product/CC1200

reduce the application flash footprint, CC2538 ROM includes a utility function library and a serial boot loader.

In the following lines, the main features of the TI CC2538 transceiver are summarized:

Table 2.10: Absolute maximum ratings of the TI CC2538
(over operating free-air temperature range, unless otherwise noted)

		MIN	MAX	UNIT
Supply voltage	All supply pins must have the same voltage	–0.3	3.9	V
Voltage on any digital pin		–0.3	V_{DD} + 0.3, ≤ 3.9	V
Input RF level			10	dBm
T_{stg}	Storage temperature range	–40	125	°C

Table 2.11: General characteristics of the TI CC2538
(Measured on TI's CC2538 EM reference design with TA = 25 °C and VDD = 3 V, unless otherwise noted.)

PARAMETER	TEST CONDITIONS	MIN	TYP	MAX	UNIT
Wake-Up and Timing					
Power mode 1 → active	Digital regulator on, 16-MHz RCOSC and 32-MHz crystal oscillator off. Start-up of 16-MHz RCOSC		4		µs
Power mode 2 or 3 → active	Digital regulator off, 16-MHz RCOSC and 32-MHz crystal oscillator off. Start-up of regulator and 16-MHz RCOSC		136		µs
Active → TX or RX	Initially running on 16-MHz RCOSC, with 32-MHz XOSC off		0.5		ms
	With 32-MHz XOSC initially on			192	µs
RX/TX and TX/RX turnaround				192	µs
USB PLL start-up time	With 32-MHz XOSC initially on		32		µs
Radio Part					
RF frequency range	Programmable in 1-MHz steps, 5 MHz between channels for compliance with (1)	2394		2507	MHz
Radio baud rate	As defined by (1)		250		kbps
Radio chip rate	As defined by (1)		2		MChip/s
Flash Memory					
Flash erase cycles				20	k Cycles
Flash page size			2		KB

2.4.2 TI CC1200 RF TRANSCEIVER

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

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

In the following lines, the main features of the TI CC1200 transceiver are summarized:

Table 2.12: Main features of the Texas Instruments CC1200 transceiver

RF Performance and analog features	
High-performance, single-chip transceiver	
Excellent receiver sensitivity	-123 dBm at 1.2 kbps
	-109 dBm at 50 kbps
Blocking performance	86 dB at 10 MHz
Adjacent channel selectivity	Up to 60 dB at 12.5-kHz channel
Very low phase noise	-114 dBc/Hz at 10-kHz Offset (169 Mhz)

Programmable Output Power from -12 dBm Up to +16 dBm With 0.4-dB Step Size	
Automatic Output Power Ramping	
Supported Modulation Formats: 2-FSK, 2-GFSK, 4-FSK, 4-GFSK, MSK, OOK	
Supports Data Rate Up to 1.25 Mbps in Transmit and Receive	
Low current consumption	
Enhanced Wake-On-Radio (eWOR) Functionality for Automatic Low-Power Receive Polling	
Power Down: 0.12 µA (0.5 µA With eWOR Timer Active)	
RX	0.5 mA in RX Sniff Mode
RX	19 mA Peak Current in Low-Power Mode
RX	23 mA Peak Current in High-Performance Mode
TX	46 mA at +14 dBm
Other	
Data FIFOs: Separate 128-Byte RX and TX	
Support for Seamless Integration With the CC1190 Device for Increased Range Providing up to 3-dB Improvement in RX Sensitivity and up to +27 dBm TX Output Power	
Digital features	
WaveMatch: Advanced Digital Signal Processing for Improved Sync Detect Performance	
Security: Hardware AES128 Accelerator	
Data FIFOs: Separate 128-Byte RX and TX	
Includes Functions for Antenna Diversity Support	
Support for Retransmission	
Support for Auto-Acknowledge of Received Packets	
Automatic Clear Channel Assessment (CCA) for Listen-Before-Talk (LBT) Systems	
Built-in Coding Gain Support for Increased Range and Robustness	
Digital RSSI Measurement	
Improved OOK Shaping for Less Occupied Bandwidth, Enabling Higher Output Power While Meeting Regulatory Requirements	
Dedicated Packet Handling for 802.15.4g	
CRC 16/32	
FEC, Dual Sync Detection (FEC and non-FEC Packets)	
Whitening	
General	
RoHS-Compliant 5-mm x 5-mm No-Lead QFN 32-Pin Package (RHB)	
Pin-Compatible With the CC1120 Device	
Regulations – Suitable for Systems Targeting Compliance With	
Europe: ETSI EN 300 220, EN 54-25	
US: FCC CFR47 Part 15, FCC CFR47 Part 90	
Japan: ARIB STD-T30, T67, T108	

2.4.3 POWER CONSUMPTION AND PERFORMANCE ANALYSIS

Battery lifetime of nodes is probably the major limitation of WSNs, which in turn depends on the balance between power consumption and energy capacity. This makes energy efficiency a critical issue to be pursued both at node and network level. Similarly, power harvesting techniques can mitigate the problem at node level.

Typical assumptions include considering the radio interface as the main contributor to power consumption. As a consequence, great attention has been given in the literature to protocol optimization, aimed for instance at minimizing the amount of data transmissions throughout the network, and the maximization of node low-power residence time. However, modeling and accurately measuring power consumption in a node should also have into account the activation of other node functional blocks, in addition to the well-known RF interface.

All the studies related to power consumption in the ENTOMATIC project will take into account 4 different power consumption modes:

- **CPU (Processing):** Related to the operation of the microcontroller, peripherals and sensors
- **LPM (Low Power Mode):** Related to the operation of the microcontroller when not being active
- **TX (Transmitting):** Related to the operation of the RF transceiver when transmitting data packets
- **RX (Receiving):** Related to the operation of the RF transceiver when receiving data packets

Table 2.13: Power consumption modes considered in the current project and their source

MODE	SOURCE
CPU	TI CC2538 Microcontroller
LPM (Low Power Mode)	
TX (Transmitting)	TI CC1200 RF Transceiver
RX (Receiving)[6]	
SL (Sleeping)	

Hence, energy consumed by a station is obtained by multiplying the time that it is expected to be in each of its operation modes by their corresponding power consumption:

$$E = t_{CPU} \cdot P_{CPU} + t_{LPM} \cdot P_{LPM} + t_{TX} \cdot P_{TX} + t_{RX} \cdot P_{RX} + t_{SL} \cdot P_{SL}$$

$$E = t_{CPU} \cdot V_{DD} \cdot I_{CPU} + t_{LPM} \cdot V_{DD} \cdot I_{LPM} + t_{TX} \cdot V_{DD} \cdot I_{TX} + t_{RX} \cdot V_{DD} \cdot I_{RX} + t_{SL} \cdot V_{DD} \cdot I_{SL} = \\ = V_{DD} \cdot (t_{CPU} \cdot I_{CPU} + t_{LPM} \cdot I_{LPM} + t_{TX} \cdot I_{TX} + t_{RX} \cdot I_{RX} + t_{SL} \cdot I_{SL})$$

Assuming $V_{DD} = 3V.$, it is still necessary to determine the value of the current associated to each operational state. This information can be found in the datasheet of the TI CC2538 microcontroller and the TI CC1200 RF transceiver. To sum up, Table 2.14 contains the main electrical characteristics of the TI CC2538 microcontroller, while power consumption per operational state in the TI CC1200 is shown in Table 2.15 for the so-called high-performance mode and in the Table 2.16 for the so-called low-power mode.

As for the time a station remains in each state, a proper method to compute these values will be presented in later sections, as they strongly depend on the communication protocols of upper layers.

[6] For the sake of simplicity, energy consumption of an STA's radio module in receiving mode when it is actually receiving a packet or just listening to the channel has been considered equal.

Table 2.14: Electrical characteristics of CC2538 with T_A = 25°C, VDD = 3.0 V, and 8-MHz system clock, unless otherwise noted

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
I_{core}	Core current consumption	Digital regulator on. 16-MHz RCOSC running. No radio, crystals, or peripherals active. CPU running at 16-MHz with flash access		7		mA
		32-MHz XOSC running. No radio or peripherals active. CPU running at 32-MHz with flash access,.		13		mA
		32-MHz XOSC running, radio in RX mode, –50-dBm input power, no peripherals active, CPU idle		20		mA
		32-MHz XOSC running, radio in RX mode at –100-dBm input power (waiting for signal), no peripherals active, CPU idle		24	27	mA
		32-MHz XOSC running, radio in TX mode, 0-dBm output power, no peripherals active, CPU idle		24		mA
		32-MHz XOSC running, radio in TX mode, 7-dBm output power, no peripherals active, CPU idle		34		mA
		Power mode 1. Digital regulator on; 16-MHz RCOSC and 32-MHz crystal oscillator off; 32.768-kHz XOSC, POR, BOD and sleep timer active; RAM and register retention		0.6		mA
		Power mode 2. Digital regulator off; 16-MHz RCOSC and 32-MHz crystal oscillator off; 32.768-kHz XOSC, POR, and sleep timer active; RAM and register retention		1.3	2	µA
		Power mode 3. Digital regulator off; no clocks; POR active; RAM and register retention		0.4	1	µA
I_{peri}	**Peripheral Current Consumption** (Adds to core current I_{core} for each peripheral unit activated)					
	General-purpose timer	Timer running, 32-MHz XOSC used		120		µA
	SPI			300		µA
	I2C			0.1		mA
	UART			0.7		mA
	Sleep timer	Including 32.753-kHz RCOSC		0.9		µA
	USB	48-MHz clock running, USB enabled		3.8		mA
	ADC	When converting		1.2		mA
	Flash	Erase		12		mA
		Burst-write peak current		8		mA

Table 2.15: Current consumption values for TI CC1200 in high-performance mode (T_A = 25°C, VDD = 3.0 V, f_c = 869.5 MHz if nothing else stated)

HIGH-PERFORMANCE MODE					
STATIC MODES					
Parameter	**Min**	**Typ**	**Max**	**Unit**	**Condition**
Power down with retention		0.12	1	µA	
		0.5		µA	Low-power RC oscillator running
XOFF mode		180	1	µA	Crystal oscillator / TCXO disabled
IDLE mode		1.5		mA	Clock running, system waiting with no radio activity
TRANSMIT MODES					
Parameter	**Min**	**Typ**	**Max**	**Unit**	**Condition**
TX current consumption +14 dBm		46		mA	
TX current consumption +10 dBm		36		mA	
RECEIVE MODES					
Parameter	**Min**	**Typ**	**Max**	**Unit**	**Condition**
RX Wait for sync 1.2 kbps, 4-byte preamble (50 kHz Channel Filter Bandwidth)		0.5		mA	Using RX sniff mode, where the receiver wakes up at regular intervals looking for an incoming packet. Sniff mode configured to terminate on Carrier Sense, and is measured using RSSI_VALID_COUNT = 1 (0 for 1.2 kbps with 50 kHz Channel Filter Bandwidth),
RX Wait for sync 1.2 kbps, 3-byte preamble (11 kHz Channel Filter Bandwidth)		3.1		mA	
RX Wait for sync 38.4 kbps, 12-byte preamble		3.4		mA	
RX Wait for sync 50 kbps, 24-byte preamble		2.1		mA	

					AGC_WIN_SIZE = 0, and SETTLE_WAIT = 1.
RX Peak Current 1.2kbps		23.5		mA	Peak current consumption during packet reception
Average current consumption Check for data packet every 1 second using Wake on Radio		8		µA	50 kbps, 5-byte preamble, 40-kHz RC oscillator used as sleep timer

Table 2.16: Current consumption values for TI CC1200 in low-power mode[7]
(T_A = 25°C, VDD = 3.0 V, f_c = 869.5 MHz if nothing else stated)

LOW-POWER MODE					
STATIC MODES					
Parameter	**Min**	**Typ**	**Max**	**Unit**	**Condition**
Power down with retention		0.12	1	µA	
		0.5		µA	Low-power RC oscillator running
XOFF mode		180	1	µA	Crystal oscillator / TCXO disabled
IDLE mode		1.5		mA	Clock running, system waiting with no radio activity
TRANSMIT MODES					
Parameter	**Min**	**Typ**	**Max**	**Unit**	**Condition**
TX current consumption +10 dBm		33.6		mA	
RECEIVE MODES					
Parameter	**Min**	**Typ**	**Max**	**Unit**	**Condition**
RX Peak Current Low-power RX mode 1.2kbps		19		mA	Peak current consumption during packet reception at the sensitivity limit

2.4.3.1 TI CC1200 Programmable output power

As it can be seen in Table 2.15 and Table 2.16, the transmitting state is the most consuming operational mode of the TI CC1200 radio module, with 46 mA when transmitting at 14 dBm and 36 mA when transmitting at 10 dBm. However, there are many other possible output powers which can be easily configured.

Table 2.17: Maximum and minimum output power for the TI CC1200

PARAMETER	MIN	TYP	MAX	UNIT	CONDITION
Max output power		+14		dBm	At 915- and 920-MHz
		+15		dBm	At 915- and 920-MHz with VDD = 3.6 V
		+15		dBm	At 868 MHz
		+16		dBm	At 868 MHz with VDD = 3.6 V
		+15		dBm	At 433 MHz
		+16		dBm	At 433 MHz with VDD = 3.6 V
		+15		dBm	At 169 MHz
		+16		dBm	At 169 MHz with VDD = 3.6 V
Min output power		–12		dBm	Within fine step size range
		–38		dBm	Within coarse step size range
Output power step size		0.4		dB	Within fine step size range

Figure 2.25 shows the relationship between the output power setting and the output power at 868 MHz, while Figure 2.26 shows the relationship between those same setting values and the measured current in radio module circuits. Figure 2.23 and Figure 2.24 complement this information by adding the influence of supply voltage and temperature over the output power.

[7] *At the time of writing this report, TI CC1200 low-power mode has not been ported to Contiki OS yet.*

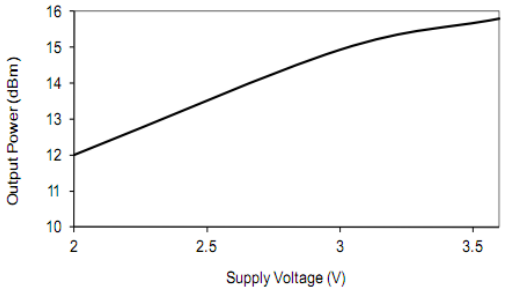


Figure 2.23: Output Power vs Supply Voltage (Maximum Output Power Setting {0x7F})

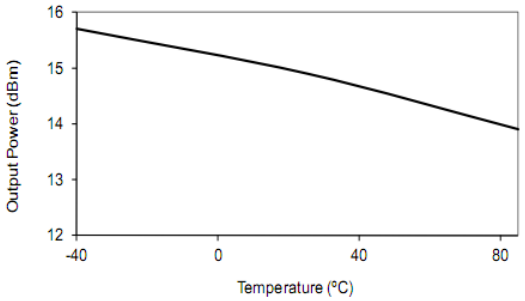


Figure 2.24: Output Power vs Temperature (Maximum Output Power Setting {0x7F})

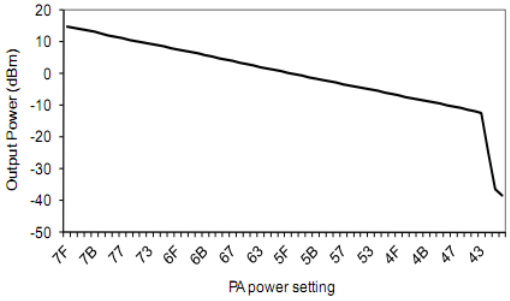


Figure 2.25: Output Power at 868 MHz vs PA Power Setting

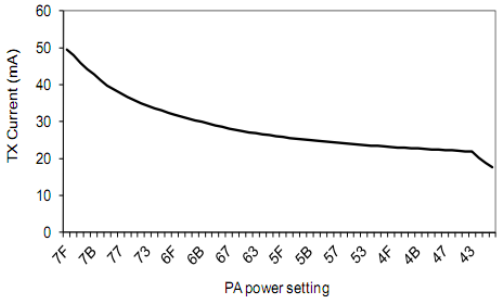


Figure 2.26: TX Current at 868 MHz vs PA Power Setting

In fact, the ENTOMATIC project will take advantage of this variety of possible output powers, so that closer stations will establish connections at the lowest possible output power, thus reducing their battery consumption and increasing their lifetime.

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

```
#define CC1200_CONST_TX_POWER_MIN   -16
#define CC1200_CONST_TX_POWER_MAX   14
```

Due to the disparity of current consumption values among different information sources, any calculation from this document will use the measures obtained from Texas Instruments official tests [18]. In these tests are described how to combine the CC2538 and the CC1200 devices in the same design. The most interesting result is that from Figure 2.27, where it is plotted the real current consumption of the whole system depending on the selected output power.

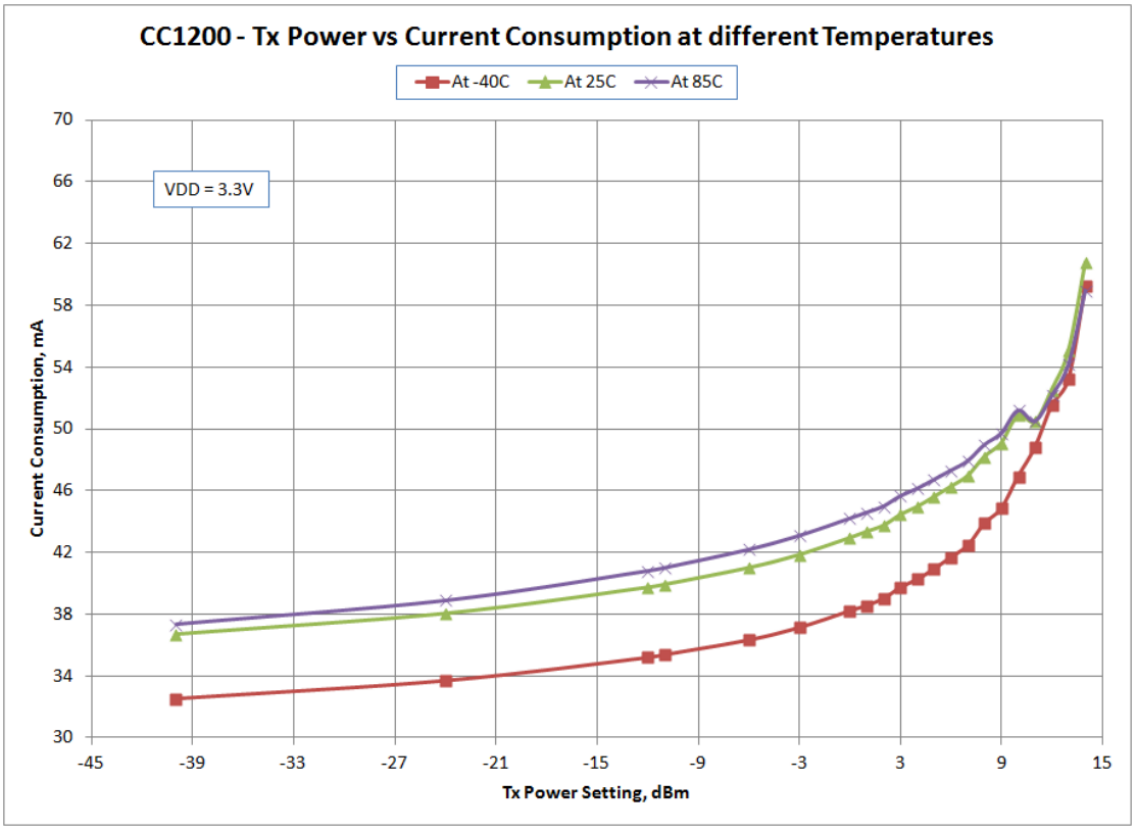


Figure 2.27: CC1200 – TX Current Consumption vs TX Power at Different Temperatures [18]
(T_C = 25°C, VDD = 3.3 V, f_c = 915 MHz if nothing else is stated.
All parameters are measured on the CC2538+CC1200 Combo EM reference design at an Antenna connector)

From Figure 2.27, the following table has been filled with real current consumption values of the TI CC1200 transceiver operating at 25°C:

Table 2.18: Current consumption value of the transmitting operational mode depending on the selected output power

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

2.4.3.2 Receiving performance

Similarly to the information obtained from other operational states, the receiving performance of the TI CC1200 strongly depends on several configuration values. In this case, it has been taken as a reference values corresponding to a data rate of 50 kbps, the only one currently available in the Zolertia RE-Mote devices.

From the TI CC1200 datasheet, TI CC1200 general receive parameters have been extracted (see Table 2.19) as well as specific receiving parameters for the high-performance (Table 2.20) and the low-power mode (Table 2.21).

Table 2.19: TI CC1200 General receive parameters (high-performance mode)
(T_A = 25°C, VDD = 3.0 V, f_c = 869.5 MHz if nothing else stated)

PARAMETER	MIN	TYP	MAX	UNIT	CONDITION
Saturation		+10		dBm	
Digital channel filter programmable bandwidth	9.5		1600	kHz	
IIP3		–14		dBm	At maximum gain
Data rate offset tolerance		±14 ±1600		% ppm	With carrier sense detection enabled With carrier sense detection disabled
Spurious emissions 1–13 GHz (VCO leakage at 3.5 GHz) 30 MHz to 1 GHz		 < –56 < –57		 dBm dBm	Radiated emissions measured according to ETSI EN 300 220, f_c = 869.5 MHz
Optimum source impedance 868-, 915-, and 920-MHz bands 433-MHz band 169-MHz band		 60 + j60 / 30 + j30 100 + j60 / 50+ j30 140 + j40 / 70 + j20		 Ω Ω Ω	(Differential or single-ended RX configurations)

Table 2.20: TI CC1200 RX Performance in 868-, 915-, and 920-MHz Bands (High-Performance Mode)
(T_A = 25°C, VDD = 3.0 V if nothing else stated)

PARAMETER	MIN	TYP	MAX	UNIT	CONDITION
Sensitivity		–122		dBm	1.2 kbps 2-FSK, DEV=4 kHz CHF=11 kHz[1]
		–113		dBm	4.8 kbps OOK
		–108		dBm	32.768 kbps 2-GFSK, DEV=50 kHz CHF=208 kHz[1]
		–110		dBm	38.4 kbps 2-GFSK, DEV=20 kHz CHF=104 kHz[1]
		–109		dBm	50 kbps 2-GFSK, DEV=25 kHz, CHF=104 kHz[1]
		-107		dBm	100-kbps 2-GFSK, DEV=50 kHz, CHF=208 kHz[1]
		–97		dBm	500 kbps 2-GMSK, CHF=833 kHz[1]
		–97		dBm	1 Mbps 4-GFSK, DEV=400 kHz, CHF=1.66 MHz[1]
Blocking and Selectivity 1.2-kbps 2-FSK, 12.5-kHz channel separation, 4-kHz deviation, 11-kHz channel filter		54		dB	± 12.5 kHz (adjacent channel)
		55		dB	± 25 kHz (alternate channel)
		77		dB	± 2 MHz
		82		dB	± 10 MHz
Blocking and Selectivity 32.768-kbps 2-GFSK, 200-kHz channel separation, 50-kHz deviation, 208-kHz channel filter		38		dB	± 200 kHz
		46		dB	± 400 kHz
		66		dB	± 2 MHz
		70		dB	± 10 MHz
Blocking and Selectivity 38.4-kbps 2-GFSK, 100-kHz channel separation, 20-kHz deviation, 104-kHz channel filter		44		dB	+ 100 kHz (adjacent channel)
		44		dB	± 200 kHz (alternate channel)
		64		dB	± 2 MHz
		72		dB	± 10 MHz
Blocking and Selectivity 50-kbps 2-GFSK, 200-kHz channel separation, 25-kHz deviation, 104-kHz channel filter (Same modulation format as 802.15.4g Mandatory Mode)		41		dB	± 200 kHz (adjacent channel)
		46		dB	± 400 kHz (alternate channel)
		65		dB	± 2 MHz
		71		dB	± 10 MHz
Blocking and Selectivity 100-kbps 2-GFSK, 50-kHz deviation, 208-kHz channel filter		45		dB	± 400 kHz (adjacent channel)
		54		dB	± 800 kHz (alternate channel)
		63		dB	± 2 MHz
		68		dB	± 10 MHz
Blocking and Selectivity 500-kbps GMSK, 833-kHz channel filter		42		dB	+ 1 MHz (adjacent channel)
		42		dB	± 2 MHz (alternate channel)
		57		dB	± 10 MHz
Blocking and Selectivity 1-Mbps 4-GFSK, 400-kHz deviation, 1.6-MHz channel filter		46		dB	± 2 MHz (adjacent channel)
		52		dB	± 4 MHz (alternate channel)
		59		dB	± 10 MHz
Image rejection (Image compensation enabled)		56		dB	1.2 kbps, DEV=4 kHz, CHF=10 kHz, image at –125 kHz

Table 2.21: TI CC1200 RX Performance in Low-Power Mode[8]
(T_A = 25°C, VDD = 3.0 V, f_c = 869.5 MHz if nothing else stated)

PARAMETER	MIN	TYP	MAX	UNIT	CONDITION
Sensitivity		–110		dBm	1.2 kbps 2-FSK, DEV=4 kHz CHF=11 kHz[1]
		–96		dBm	50 kbps 2-GFSK, DEV=25 kHz, CHF=119 kHz[1]
Blocking and Selectivity 50 kbps 2-GFSK, 200-kHz channel separation, 25-kHz deviation, 104-kHz channel filter (Same modulation format as 802.15.4g Mandatory Mode)		41		dB	+ 200 kHz (adjacent channel)
		45		dB	+ 400 kHz (alternate channel)
		62		dB	± 2 MHz
		60		dB	± 10 MHz
Saturation		+10		dBm	

2.4.3.3 Wake-up-related timings

As it will be seen in following sections, the main method to save energy for an STA will be to stay asleep the most possible time. Thus, STAs will remain in sleep mode not only when not interacting with the rest of the network, but also they will combine *active* and *sleeping* periods even when exchanging data packets with other STAs or the gateway.

[8] *At the time of writing this report, TI CC1200 low-power mode has not been ported to Contiki OS yet.*

That's the reason why time delays due to the process of changing the operational state of an STA are so important and must be kept in accordance with the rest of mechanisms from upper layers. Values from Table 2.22 show very quick transitions between different active states (always below 500 µs), which means minimum impact on the rest of the system's procedures.

Table 2.22: Wake-up and timing
(T_A = 25°C, VDD = 3.0 V, f_c = 869.5 MHz if nothing else stated)

PARAMETER	MIN	TYP	MAX	UNIT	CONDITION
Powerdown to IDLE		0.24		ms	Depends on crystal
IDLE to RX/TX		133		µs	Calibration disabled
		369		µs	Calibration enabled
RX/TX turnaround		43		µs	
RX-to-RX turnaround		369		µs	With PLL calibration
		0		µs	Without PLL calibration
TX-to-TX turnaround		369		µs	With PLL calibration
		0		µs	Without PLL calibration
RX/TX to IDLE time		237		µs	Calibrate when leaving RX/TX enabled
		0		µs	Calibrate when leaving RX/TX disabled
Frequency synthesizer calibration		314		µs	When using SCAL strobe
Minimum required number of preamble bytes		0.5		bytes	Required for RF front-end gain settling only. Digital demodulation does not require preamble for settling.
Time from start RX until valid RSSI[1] Including gain settling (function of channel bandwidth. Programmable for trade-off between speed and accuracy)		4.2		ms	12.5-kHz channels
		0.25		ms	120-kHz channels

2.4.3.4 Data rate

Depending on the node's transceiver, one or more transmission power and data rate levels may be available (or programmable), being the maximum data rate dependent on the sensitivity, as the communication range is determined by the link budget, i.e., the difference between the receiver's sensitivity and the transmission power of the transceiver.

This complex relation among the variables impacting on the transceiver consumption hardens the task of identifying in advance which are the best combinations of transmission power and data rate for each of the established routing connections. For example, raising the transmission power would also increase the current consumption, impacting negatively on the energy. However, a higher data rate can be used when less demanding sensitivities are required, and therefore, the time transmitting and receiving could be decreased (e.g., a radio transmitting at 50 kbps remains in TX state approximately twice the time a radio transmitting at 100 kbps), having a positive impact on the energy consumption. Moreover, the transmission power level (and power output) is not usually linear with the corresponding power consumption (see Figure 2.27), which hardens even more identifying the most suitable combination of transmission power and data rate beforehand.

On the one hand, the current porting of the TI CC1200 RF transceiver in Contiki OS only allows the whole platform to transmit at **50 kbps**, which enables a -109 of receiving sensitivity. Although this data rate is fixed due to firmware issues, the corresponding sensitivity is favorable for the ENTOMATIC scenarios as the coverage range satisfies clearly the requirements. Specifically, using the outdoor path loss model defined for macro deployments defined by authors in [19],

$$PL(d) = 8 + 37.6\log_{10}(d) + 21\log_{10}\left(\frac{868}{900}\right)$$

where *PL(d)* is the path loss in dBs at a distance *d*, and *f* is the carrier frequency, the theoretically reachable distance with CC1200 operating at maximum power (14 dB) and 50 kbps data rate is 1,400 meters. Real tests on range are described in Section 2.11.3.

On the other hand, the current porting of the TI CC2420 RF transceiver in Contiki OS only allows the whole platform to transmit at **250 kbps**. Even though the transmission and receiving times are considerably reduced with this transceiver, propagation issues caused by high frequency operation and

worst sensitivity (-95 dBm) are a major challenge due to the fact that some STAs may will not be able to listen to the GW.

2.4.3.5 Range coverage

As stated in previous deliverables, communication problems in olive orchards are one of the risk factors of the current project. To ensure the communication link between different traps (as well as between traps and the gateway) is responsibility of the chosen RF transceiver. One advantage of using lower frequencies (for instance, 868 MHz instead of 2.4 GHz) is that signals have better penetration, meaning they pass through objects such as walls with less attenuation. This effect results in larger range coverage.

The wireless radio channel poses a severe challenge as a medium for reliable high-speed communication. It is not only susceptible to noise, interference, and other channel impediments, but these impediments change over time in unpredictable ways due to user movement [20]. Three mutually independent, multiplicative propagation phenomena can usually be distinguished: large-scale path loss, shadowing and multipath fading.

- **Large-scale path loss**
  The 'large-scale' effects of path losses cause the received power to vary gradually due to signal attenuation determined by the geometry of the path profile in its entirety. This is in contrast to the local propagation mechanisms, which are determined by terrain features in the immediate vicinity of the antennas.

- **Shadowing**
  Shadowing is a 'medium-scale' effect caused by obstacles between the transmitter and receiver that absorb power.

- **Multipath fading**
  Multipath propagation leads to rapid fluctuations of the phase and amplitude of the signal due to the constructive and destructive addition of multipath signal components.

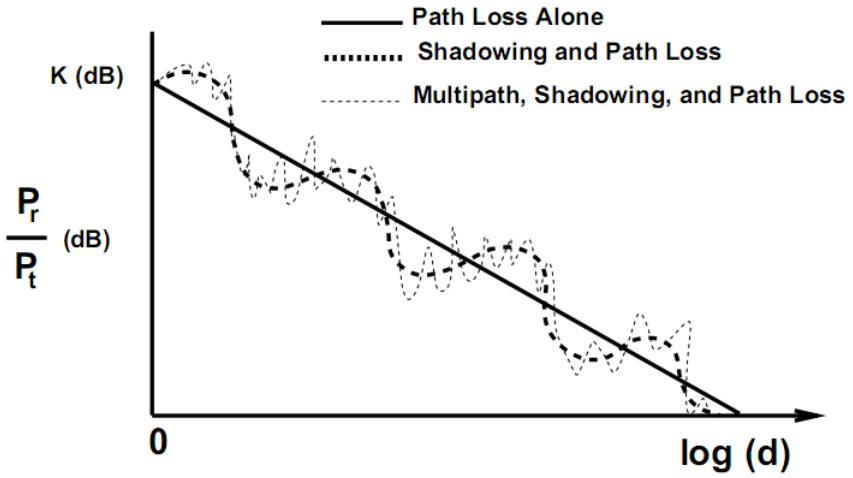


Figure 2.28: Path Loss, Shadowing and Multipath versus Distance [20]

Since variations due to path loss and shadowing occur over relatively large distances, this variation is sometimes referred to as large-scale propagation effects or local mean attenuation. Due to the deployment requirements of the ENTOMATIC traps, where at least one radio transmitter should be installed for every 2 ha of large scale fields of homogeneous lands, only large-scale propagation effects will be considered.

As a mode of illustration, two propagation models are presented in the current document: the free space propagation model and the log-distance path loss model. Lastly, the Texas Instrument range

coverage calculator is presented and used to validate the use of the TI CC1200 RF transceivers in the current project.

Free space propagation model

The free-space propagation model is used to predict the received power level in some location where there exists line-of-sight (LOS) between the transmitter (Tx) and receiver (Rx). This model states that the power decreases as a function of the distance d between Tx and Rx, according to the Friis equation:

$$\mathrm{P_r(d)} = \frac{\mathrm{P_t G_t G_r}}{\mathrm{L}} \cdot \left(\frac{\lambda}{4\pi \mathrm{d}}\right)^2$$

where P_t is the transmitted power, $P_r(d)$ is the received power, G_t is the gain of the transmitting antenna, G_r is the gain of the receiving antenna, d is the Tx-Rx separation in meters, L is a non-propagation loss factor $(\mathrm{L} \geq 1)$, and λ is the signal wavelength in meters.

Path loss represents signal attenuation as a magnitude positive, expressed in dB, and is defined as the difference between the transmitted and the received power, according to the following equation:

$$PL(dB) = -20log\left(\frac{\lambda}{4\pi d}\right)$$

It is worth noting the fact that the previous equations are only valid beyond the Fraunhofer distance, that is, for those distances exceeding:

$$d_f = \frac{2D^2}{\lambda}, \qquad d_f \gg \lambda$$

where D is the greatest linear dimension of the transmitting antenna.

Log-distance path loss model

In many cases, empirical results agree with Friis logarithmic decay of distance. However, the quadratic exponent is not the best option to fit path loss in many actual propagation environments. Path loss are generally expressed as a function of distance through an exponent n, according to:

$$\overline{PL}(d) \propto \left(\frac{d}{d_0}\right)^n$$

where d_0 is a reference distance where conditions of free space are met (typically 1 meter) and d is the separation between Tx and Rx. The following table provides a series of typical values for n depending on the communication environment:

Table 2.23: Values of path-loss exponent *(n)* according to the environment

Environment	Path loss exponent *(n)*
Free Space	2
Urban area cellular radio	2.7 to 3.5
Shadowed urban cellular radio	3 to 5
In building line of sight	1.6 to 1.8
Obstructed in building	4 to 6
Obstructed in factories	2 to 3

Range coverage of TI CC1200 RF transceiver

The Texas Instrument range coverage calculator[9] provides a theoretical calculation of the range coverage of different TI RF transceivers in function of different parameters, such as the frequency, the TX and RX antenna locations, the data rate, or even the presence of different materials in the signal

[9] Texas Instruments – Application Report: Achieving optimum radio range (http://www.ti.com/lit/an/swra479/swra479.pdf)

path. It is based on the Friis Equation presented in Section 0 and can also characterize the shadowing effects of up to 3 materials placed between the transmitter and the receiver.

This tool has been used to prove the feasibility of using the TI CC1200 transceiver in the ENTOMATIC project, where at least one radio transmitter should be deployed for every 2 ha of field. As 1 ha = 100 m. x 100 m., in the worst possible configuration two adjacent transmitter would be located at 250 m. of distance. Among all possible transmission powers (from -16 dBm to 14 dBm) and data rates (from 1.2 kbps to 1 Mbps), some representative values have been selected for testing the range coverage calculator.

Indeed, it is a powerful tool that provides a very understandable graphic of the signal performance in function of the distance. Besides, multiple configuration parameters can be tuned in order to make the simulation as real as possible, including for instance some absorbing materials between the transmitter and the receiver, or modifying the height where antennae are located. Figure 2.29, Figure 2.30 and Figure 2.31 show the results provided by the tool when maximum transmission power for a CC1200 transceiver is selected and only data rate is modified.

A more complete study is provided in Table 2.24, where values of range coverage for all possible transceiver configurations are shown. As it can be seen, only 3 transceiver configurations (the ones with the lowest transmission powers for the 1 Mbps data rate mode) do not achieve the required distance of 250 m. Besides, as the porting of the CC1200 transceiver functions in Contiki only allows using the 50 kbps data rate mode, even the less powerful transmission is over that value.

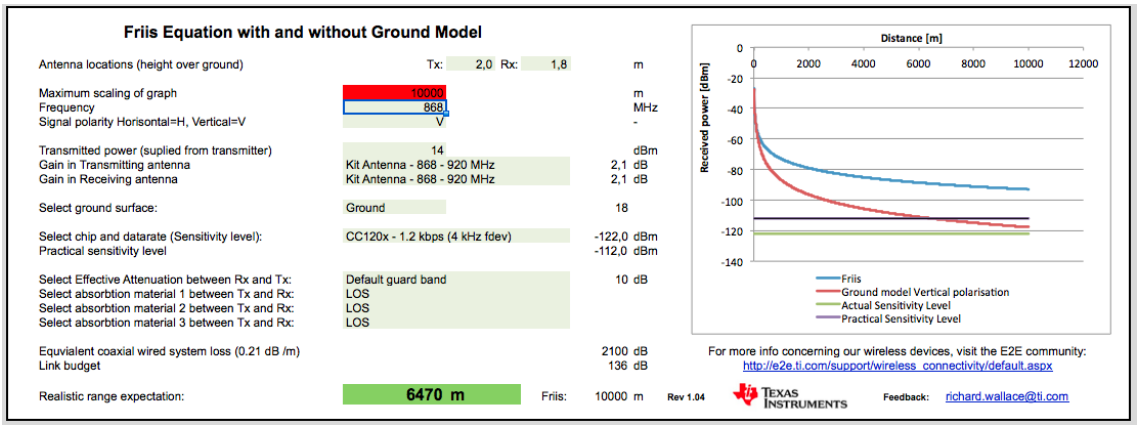


Figure 2.29: Realistic range coverage of TI CC1200 RF transceiver at maximum power transmission (14 dBm) and 1.2 kbps datarate

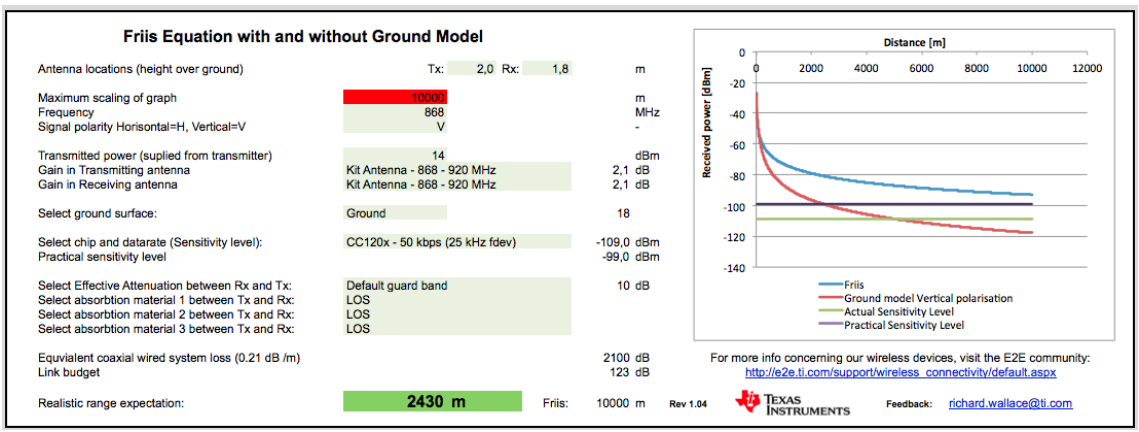


Figure 2.30: Realistic range coverage of TI CC1200 RF transceiver at maximum power transmission (14 dBm) and 50 kbps datarate

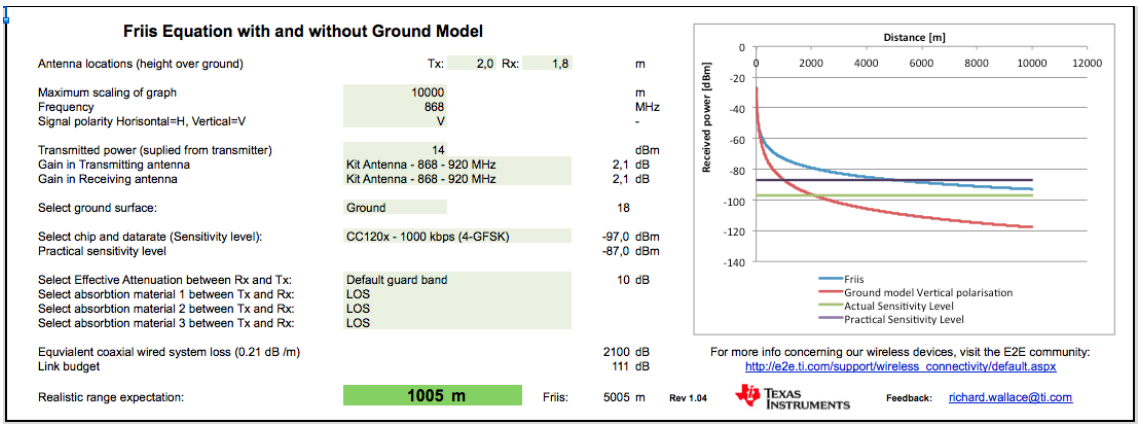


Figure 2.31: Realistic range coverage of TI CC1200 RF transceiver at maximum power transmission (14 dBm) 1 Mbps datarate

Table 2.24: Realistic range expectation of TI CC1200 RF transceiver calculated by the Texas Instrument range coverage calculator

Texas Instruments CC1200 RF transceiver			
Transmitted power (supplied from transmitter)	**1.2 kbps data rate**	**50 kbps data rate**	**1 Mbps data rate**
-16 dBm	700 m.	**285 m.**	**125 m.**
-12 dBm	935 m.	**375 m.**	**165 m.**
-8 dBm	1250 m.	**495 m.**	**215 m.**
-4 dBm	1675 m.	**655 m.**	285 m.
0 dBm	2255 m.	**870 m.**	375 m.
6 dBm	3535 m.	**1345 m.**	565 m.
10 dBm	4780 m.	**1805 m.**	755 m.
14 dBm	6470 m.	**2430 m.**	1005 m.

2.5 MAC LAYER

In previous years, a large number of MAC protocols that are specially designed for WSNs have been defined. To save energy, especially the energy wasted because of idle listening, the most common approach is to put the transceiver into sleep mode for as much time as possible because sleep mode consumes substantially less energy than the other available modes (idle, transmitting or receiving).

In sleep mode, a sensor node is not able to receive/transmit packets from/to the medium. This solution greatly reduces the idle listening energy waste. However, specific mechanisms should be defined to ensure that a sensor node will be awake if a node sends something to it. There are three general categories to address this issue, as follows [21]:

- **Scheduled (Time Division Multiple Access (TDMA)-like) protocols**
  In this category, sensor nodes are assigned to specific slots in a frame to transmit and/or receive; then, the nodes only wake up in those slots and sleep in the rest. A node must know the time slot in which a neighbour will be awake before it can send data to it.

Table 2.25: Characteristics of TDMA-based MAC protocols

Protocol	Pure TDMA	Distributed	Cluster Formation	Mobility	Synchronization	Multihop	Channel Interference	Energy Efficiency
SMACS 1	no	yes	only during network formation	no	not network wide	yes	possible	N/A
SMACS 2	yes	yes	only during network formation	limited	not network wide	yes	possible	radio-off phase
PACT	no	partly	yes	N/A	yes	yes	no	radio-off phase
MAC WITH EDF	yes	no	yes	no	yes	yes	no	N/A
DE-MAC/ ER-MAC	yes	yes	no	no	yes	N/A	no	radio-off phase
TRAMA	no	partly	no	N/A	yes	yes	possible	radio-off phase
TRACE	no	no	yes	N/A	not network wide	no	possible	radio-off phase
CONTENTION-FREE MESSAGE SCHEDULER	yes	partly	no	N/A	yes	no	no	yes
EMACS	yes	yes	no	yes	yes	yes	possible	radio-off phase
GANGS	no	no	yes	N/A	not network wide	N/A	possible	radio-off phase
BMA MAC	yes	no	intra-cluster communication	N/A	yes	no	N/A	radio-off phase
BTODS/ODS	yes	yes	no	N/A	yes	no	possible	radio-off phase
TDMA-W	yes	yes	no	N/A	not necessary	yes	no	radio-off phase

- **Protocols with Common Active Periods**
  In these protocols, nodes define common active/sleep periods. The active periods are used for communication and the sleep ones for saving energy. This approach requires that nodes maintain a certain level of synchronization to keep active/sleep periods common to all nodes. During the active periods, nodes contend for the channel using contention-based approaches, such as pure CSMA, IEEE 802.11 DCF, etc.

  The contention-based approach achieves its highest performance in applications in which traffic is periodic such as monitoring and applications in which keep-alive packets are periodically exchanged to ensure network reliability. However, the use of common active/sleep periods may not be suitable for applications with irregular traffic, because nodes use contention inside active periods, which would be prohibitive when nodes wake up without communicating, and may cause collisions when there is high traffic that cannot be absorbed by the initially envisaged size of the active periods.

Table 2.26: Summary of MAC protocols and their prime roles for scheduled protocols [22]

Function	Protocols
Canonical Solutions	TSMP [27], IEEE 802.15.4 [5]
Centralized Scheduling	Arisha [29], PEDAMACS [30], BitMAC [31], G-MAC [32]
Distributed Scheduling	SMACS [33]
Localization-Based Scheduling	TRAMA [34], FLAMA [35], μMAC [36], EMACs [37], PMAC [38]
Rotating Node Roles	PACT [39], BMA [40]
Handling Node Mobility	MMAC [41], FlexiMAC [42]
Adapting to Traffic Changes	PMAC [43]
Receiver Oriented Slot Assignment	O-MAC [43]
Using Different Frequencies	PicoRadio [45], Wavenis [3], f-MAC [48], Multichannel LMAC [49], MMSN [51], Y-MAC [52], Practical Multichannel MAC [53]
Various Functionalities	LMAC [50], AI-LMAC [54], SS-TDMA [55], RMAC [56]

- **Asynchronous MAC protocols**
  In this type of protocol, no synchronisation is required, because each node goes to sleep and wakes up independently of the others. These protocols implement different mechanisms to ensure that a receiver will be awake to receive the data. The protocols can be divided into preamble sampling and receiver-initiated approaches:

  - **Preamble sampling MAC protocols**
    In preamble sampling, sensor nodes wake up only for checking whether the channel is busy, in which case they remain awake. Next, a sensor node wanting to transmit first sends a long preamble to overlap with the channel sampling time of the receiver and subsequently sends the data.

Category	List of MAC protocols
Basic	Preamble sampling [14], LPL [15], B-MAC [12]
Short preamble burst	ENBMAC [17], BMAC+ [18], SpeckMAC-B [19], DPS-MAC [20], SyncWUF [21], SESP-MAC [22], TICER [23], CSMA-MPS [24], X-MAC [25], AS-MAC [26], Patterned Preamble MAC [27], AREA-MAC [28], 1-hopMAC [30], DPS-MAC' [31], CMAC [32], MX-MAC [33], Preamble sampling with state information [29], SpeckMAC-D [19], MH-MAC [34], MIX-MAC [35], TrawMAC [36], MFP-MAC [37], RA-MAC [38], SEESAW [39]
Taking advantage of sync. info.	WiseMAC [13], CSMA-MPS [24], SyncWUF [21], TrawMAC [36], AS-MAC' [40], SCP-MAC [41], MIX-MAC [35]
Adaptative duty cycle	WiseMAC more bit [13], Stay Awake Promise [43], BEAM [44], AS-MAC [26], AREA-MAC [28], AS-MAC'' [45], extension to B-MAC+ [46], BoostMAC [47], MaxMAC [48], LWT-MAC [49], SCP-MAC [41], AADCC and DDCC [50], ZeroCal [51], X-MAC [25], EA-ALPL [53], Preamble Sampling with State Information [29]

Figure 2.32: List of preamble sampling MAC protocols [23]

- **Receiver-initiated MAC protocols**
  In receiver-initiated protocols, sensor nodes send an advertisement when they wake up. Then, a sensor node with a packet to transmit waits until it receives the advertisement sent by the receiver.

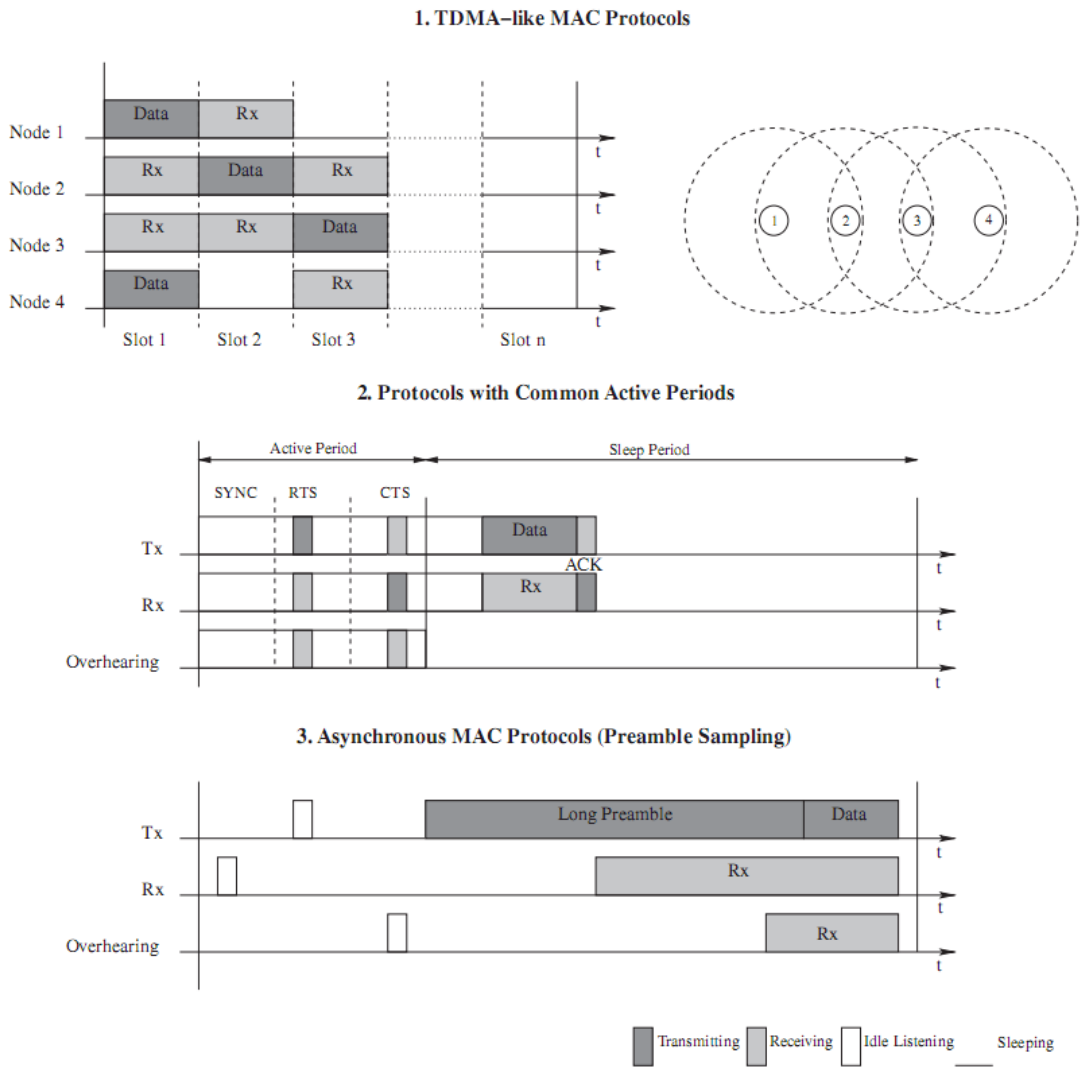


Figure 2.33: Schematic representation of the different MAC categories [23]

Among the different categories of MAC protocols, those which best fit the ENTOMATIC requirements are the ones with Common Active Periods, due to the following reasons:

- They achieve their highest performance in applications in which traffic is periodic such as in monitoring applications
- Nodes can be easily configured with long sleeping periods in order to lengthen their battery lifetime
- During the active periods, nodes can contend for the channel using different contention-based approaches, so that several options can be tested and implemented
- These protocols can be centrally managed by a device with greater resources in terms of energy, memory and processing. Besides, it would be responsible of gathering all data from the monitoring area and sending it to the data receiver server

The last reason, related to the centralized management, has encouraged us to explore in depth the characteristics and operation of the different existing Protocols with Common Active Periods and centralized scheduling, as pointed out in Table 2.26.

2.5.1 MAC PROTOCOLS FOR WIRELESS SENSOR NETWORKS

2.5.1.1 MAC Protocols with Common Active Periods and centralized scheduling

The main objective of this kind of protocols is to extend the lifetime of sensors through topology adjustment, energy aware routing and reduction of MAC active periods. Messages are routed through multi-hop paths to preserve the sensor transmission energy. Message traffic between sensors is arbitrated in time to avoid collision and to allow turning off the unneeded sensors.

Gateway nodes assume responsibility in its coverage area for sensor organization and routing/MAC management. Since the gateway organizes the sensors in the cluster, it can account for energy commitment to data processing, remaining sensor energy, sensor locations and acceptable quality of service such as message latency. It can as well enhance the robustness and effectiveness of the MAC layer because the decision to turn a node receiver off can be more accurate and deterministic than a decision based on local MAC protocol.

In these protocols, the routing approach is also centralized in the gateway node for each coverage area. Suitable routes for sensor data from the active sensor node to the gateway are set according to the optimization of an objective function with several cost factors. Basically, the problem can be viewed as the transpose of a single-source routing algorithm, i.e. single destination routing, where cost factors are classified in the following areas:

- **Energy**
  Communication cost, remaining sensor energy, energy consumption rate, relaying enabling cost, and sensing cost
- **Routing**
  Number of relaying sensors, distance to the sink
- **Delay**
  Transmission delay, queuing cost

The most important examples of protocols running in this category are the following ones:

Arisha

Arisha [24] concentrates all the computing systems in the gateway, so that it gathers topology information about the network topology and assigns slots to all the nodes. Besides, this central device also details the algorithm used to compute the scheduling table where the slot assignment can be done according to two different techniques:

- **Breadth first search (BFS)**
  This technique starts by first assigning slots at the leaf nodes of the tree. Leaf nodes having the same parent are assigned contiguous transmission slots. Other nodes sharing another parent are assigned the subsequent slots, etc. In turn, parents are assigned the subsequent slots according to the same procedure.

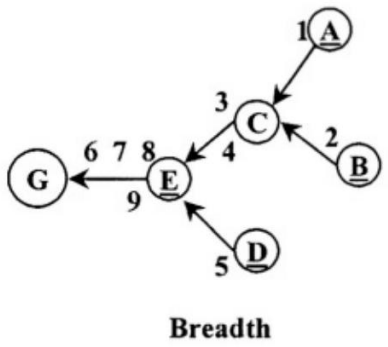


Figure 2.34: Example of breadth first search slot assignment technique

The main advantages are that parents minimize switching times as they keep listening to their children during a contiguous interval and data aggregation can be easily performed. The drawback is that this technique introduces some delay in intermediate nodes, as parents first receive data from all their children before forwarding it.

- **Depth first search (DFS)**
  In depth first search, slot assignment also starts at one leaf node. The next slot is assigned to its parent; the after next slot is assigned to its grandparent; etc; until reaching the sink. After that, the next leaf node follows the same process.

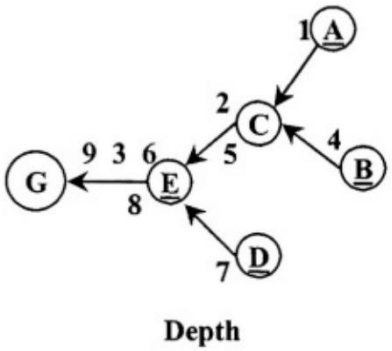


Figure 2.35: Example of depth first search slot assignment technique

This model avoids backlogging packets in intermediate nodes, which reduces end-to-end delay, packet loss thus increasing the throughput. However, it also involves larger energy consumption as intermediate nodes experience more frequent status switching. Lastly, another issue of this technique is scalability, as there is no space re-use of unoverlapped slots.

PEDAMACS

In PEDAMACS [25], the sink gathers information about traffic and topology during the setup phase. Based on this gathered information, the sink calculates a global scheduling and sends it to the entire network, so that uplink communications follows the TDMA scheme established by the sink. Then, a topology learning phase starts, and the sink node floods the network with topology-learning packets. At the end of this phase, a spanning tree is constructed and the sink has the knowledge of the entire topology.

The main assumption of this protocol is that the sink (or gateway) can reach all network nodes in one hop, while the path from a sensor node to the AP comprises several hops. Although only some applications can use a PEDAMACS network, the system is quite flexible and can be generalized in many ways. For instance, the author shows the generalization of PEDAMACS to handle event-driven data generation, existence of more than one AP and the nodes located outside the range of the AP.

For applications where nodes periodically generate packets, the PEDAMACS network provides a lifetime of several years compared to several months and days based on random access schemes with and without sleep cycles, respectively, making sensor network technology economically viable.

BitMAC

Collisions are a source of inefficiency in contention-based MAC protocols that should be reduced to a minimum. BitMAC [26] shows that concurrent multiple access to a communication channel will, however, not necessarily lead to a collision with undesirable effects. Rather, it is possible for a receiver to hear the bitwise “or” of the transmissions of multiple synchronized senders within communication range.

Based on this communication model, BitMAC has the following key features:

- **No collisions:** Although nodes access the channel concurrently, this does not lead to bad effects
- **Determinism:** There are deterministic bounds on the execution time of all protocol elements
- **Robustness:** A large class of temporary and permanent node failures can be tolerated in dense networks
- **Efficiency:** The protocol overhead of BitMAC is low and the channel bandwidth can be utilized efficiently
- **Self-containedless:** BitMAC does not assume any out-of-band mechanisms. Time synchronization is an integral part of the protocol
- **Dense, many-to-one networks:** BitMAC is particularly tailored to dense sensor networks, where sensor nodes report sensor readings to a sink node over multiple hops

The protocol supports uplink communication from the children to a parent, and downlink communication from the parent to one or more children. Time-division multiplexing is used to avoid collisions and to ensure a deterministic behaviour of the protocol. In order to optimize bandwidth utilization, time slots are allocated on demand to nodes that actually need to send data.

The protocol proceeds in rounds with the parent acting as a coordinator. A round starts with the parent broadcasting a beacon message to the children. The beacon contains an indicator whether this round is downlink or uplink communication. If the round is downlink, the beacon message will be followed by the payload data, which will usually contain the address (e.g., ID) of the target node(s). If this is an uplink round, children will transmit send requests to the parent. After receiving these requests, the parent constructs a schedule and broadcasts it to the children. From this schedule, children can deduce their time slot for transmission. During their time slots, children send their payload data to the parent. The parent will then acknowledge successful receipt. If transmission failed, the affected children will try a retransmission in the next round.

To extend the protocol to a multi-hop network, it will use a spanning tree with the sink at the root, where each internal node and its direct children form a star that uses the above presented protocol. As to avoid interference, stars are assigned different communication channels.

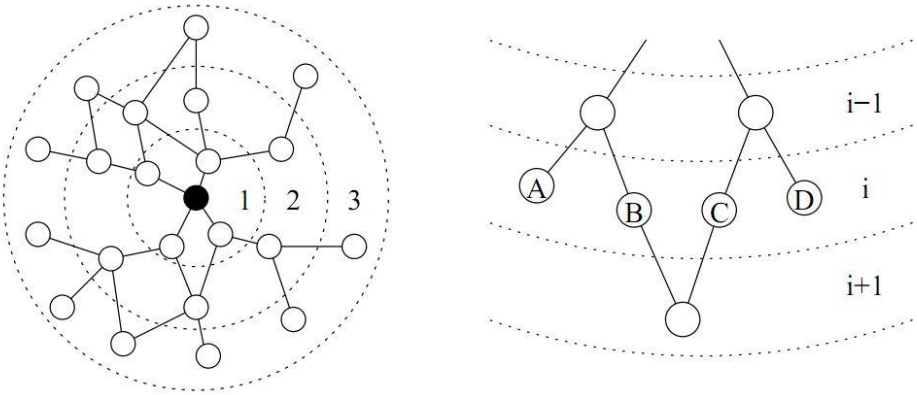


Figure 2.36: Distance from the sink imposes a ring structure on the network

G-MAC

G-MAC [27] implements a new cluster-centric paradigm to effectively distribute cluster energy resources and extend network lifetime. The centralized cluster management function offers significant energy savings by leveraging the advantages of both contention and contention-free protocols. A centralized gateway node collects all transmission requirements during a contention period and then schedules their distributions during a reservation-based, contention-free period. With minimal overhead, the gateway duties are efficiently rotated based upon available resources to distribute the increased network management energy requirements among all of the nodes.

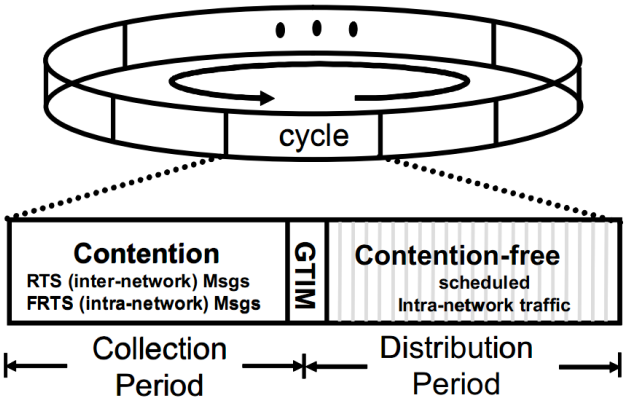


Figure 2.37: G-MAC frame architecture

G-MAC maintains synchronization among nodes and sets up slot owners among nodes having data to be sent to the gateway. Although the system reduces TDMA rigidity by allowing time-critical packets to be sent during the collection phase, it has some drawbacks. For instance, the large overhead experienced by the gateway node, especially during the collection phase that should be large enough to decrease the number of collisions among contenders.

2.5.1.2 Existing MAC protocols in Contiki OS

The network stack implemented in Contiki OS is a little bit different than the usual 5-layers model typically adopted in TCP/IP. In-between the physical and the network layers, where usually is located the MAC, we have 3 different layers: framer, radio duty-cycle (RDC) and medium access control (MAC). The figure below shows the organization of layers in Contiki OS.

Figure 2.38: Stack of communication layers in Contiki OS

2.5.1.3 Framer Layer

Framer layer is not a regular layer implementation; it is actually a collection of auxiliary functions that are called for creating a frame with data to be transmitted and parsing of data being received. Actually, a MAC framer is responsible for constructing and parsing the header in MAC frames. There are two types of framer layers:

- **802.15.4 framer**
  The 802.15.4 framer type creates and parses frames compatible with the standard IEEE 802.15.4 from 2003. The *create* and *parse* functions manipulate all header information

abstractedly through the *packetbuf* structure. It is responsible for reading all information from receive/transmit buffer and inserting/extracting them appropriately into *packetbuf* structure.

- **NullMAC framer**
  The NullMAC framer should be used together with NullMAC (MAC layer). This auxiliary function simply fills 2 fields of the header: receiver address and sender address. This is the very basic type of framer that can be used on Contiki OS.

2.5.1.4 Radio Duty Cycling (RDC)

Radio Duty Cycling (RDC) layer takes care of the sleep period of nodes. This is the most important layer because it is the one responsible for deciding exactly when the packets will be transmitted and it is responsible for making sure that the node is awake when packets are to be received. Examples of RDC layers that are implemented:

- **X-MAC**
  X-MAC [28] is a power-saving MAC protocol in which senders use a sequence of short preambles (strobes) to wake up receivers. Nodes turn off the radio for most of the time to reduce idle listening. They wake up shortly at regular intervals to listen for strobes. When a receiving node wakes up and receives a strobe destined to it, it replies with an acknowledgment indicating that it is awake.

- **ContikiMAC**
  ContikiMAC [29] is a radio duty cycling protocol that uses periodical wake-ups to listen for packet transmissions from neighbors. If a packet transmission is detected during a wake-up, the receiver is kept on to be able to receive the packet. When the packet is successfully received, the receiver sends a link layer acknowledgment.

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

- **NullRDC**
  NullRDC is a minimalistic RDC protocol that simply forwards traffic between the network layer and the radio driver. As such, it does not provide any power-saving mechanism, and keeps the radio always on. This allows for the maximum throughput achievable, while consuming the highest amount of energy. When used with CCA and back-off timers, NullRDC behaves as a traditional CSMA-CA protocol.

2.5.1.5 Medium Access Control (MAC)

Finally, the MAC layer takes care of addressing and retransmission of lost packets. There are only two types of MAC layers available: csma.c and nullmac.c.

- **CSMA**
  CSMA is the default mechanism. The MAC layer receives incoming packets from the RDC layer and uses the RDC layer to transmit packets. If the RDC layer or the radio layer detects a radio collision, the MAC layer may retransmit the packet at a later point in time. The CSMA mechanism is currently the only MAC layer that retransmits packets if a collision is detected. It is also able to keep record of the number of retransmissions, collisions, deferrals, etc.

  Clear Channel Assessment (CCA) [30] is a mechanism used to determine if a wireless channel is currently free. In wireless MAC protocols like the one implemented in the ENTOMATIC project, CCA is used to implement Carrier Sense Multiple Access (CSMA): each node first listens to the medium to detect ongoing transmissions, and transmits its packet only if the channel is free, thus reducing the chance of collisions.

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

- **NullMAC**
  NullMAC is a pass-through protocol that simply calls the appropriate RDC functions.

2.5.2 OVERVIEW OF THE ENTOMATIC MAC LAYER

The specific requirements and hardware constraints of the ENTOMATIC project have made indispensable the design of a new MAC layer based on the foundations of some other MAC protocols with common active periods and centralized scheduling.

The ENTOMATIC MAC layer is a sort of combination of a time division multiple access (TDMA) based MAC layer whose slot assignment is managed by the gateway, and a pure CSMA protocol responsible of managing contention-based medium access by multiple stations within those slots.

In this TDMA scheme, the gateway transmits guide-beacons informing nodes about the time periods in which they should listen to the transmissions of their children and about time periods which should be used for their own transmissions. The main advantages of using such a scheme are:

- Clock synchronization is inherent to the TDMA protocol, so that nodes are constantly set in time by means of listening to the beacons
- Nodes are sleeping most of time; i.e., all time excepting time devoted to listen to gateway's beacons or during their own time slots
- As nodes have their time slots clearly assigned, collision among them are sensibly reduced or even avoided, as it can only occur if the node happens to use the same time slot for transmission as a neighboring hidden node
- Association and routing mechanisms are also fit into this scheme, so that intermediate and already associated nodes do not have to constantly listen to hypothetical network discovery requests
- Changes in the network configuration or even new firmware can be easily transmitted in a coordinated manner

Three different combinations of MAC sublayers existing in Contiki OS have been tested due to the different features of each one of them. While the framer layer and the medium access control have been set to 802.15.4 framer and CSMA, respectively, the differences between the existing RDC sublayers have motivated a deeper study.

Table 2.27: MAC configurations tested for the ENTOMATIC network

	Framer layer	Radio Duty Cycling (RDC)	Medium Access Control (MAC)
Configuration A	802.15.4 framer	Nullmac	CSMA
Configuration B	802.15.4 framer	X-MAC	CSMA
Configuration C	802.15.4 framer	ContikiMAC	CSMA

These three elements (framer, radio duty cycling and medium access control) define the communication between two network elements and ensure the proper delivery of messages between them. However, when dealing with a bunch of station without communication between them and with the aim of reducing redundant transmissions, additional mechanisms are necessary to schedule the normal operation of the deployed stations.

2.5.3 BEACONING SYSTEM

The designed beaconing system has a double function: synchronizing the network devices and scheduling the different actions to be performed. Two main types of beacons are used for this purpose: *primary* and *secondary beacons* (see Figure 2.39).

Both beacons include a timestamp as well as the time until the next *primary beacon*. In addition, the content of a *primary beacon* determines the action to be taken next by the network: for instance, a data transmission phase (*data primary beacon*) or a re-association phase (*re-association primary beacon* [10]). The purpose of *secondary beacons* is to guarantee information redundancy for already associated STAs as well as faster network discovery for non-associated ones. (Frame structures are detailed in Section 2.8.2).

Time between two consecutive *primary beacons* and two consecutive *secondary beacons* is defined as T_p and T_s, respectively. Where $T_p = (k_s + 1) \cdot T_s$, being k_s the number of *secondary beacons* transmitted after every *primary beacon*.

2.5.3.1 Scheduling and temporal evolution

Figure 2.39 shows an example of the general scheduling procedure. In the mentioned example, 4 beacons are spread for redundancy (3 secondaries and 1 primary). The primary beacon triggers the *action cycle*, i.e., the time period including the association and data transmission phases.

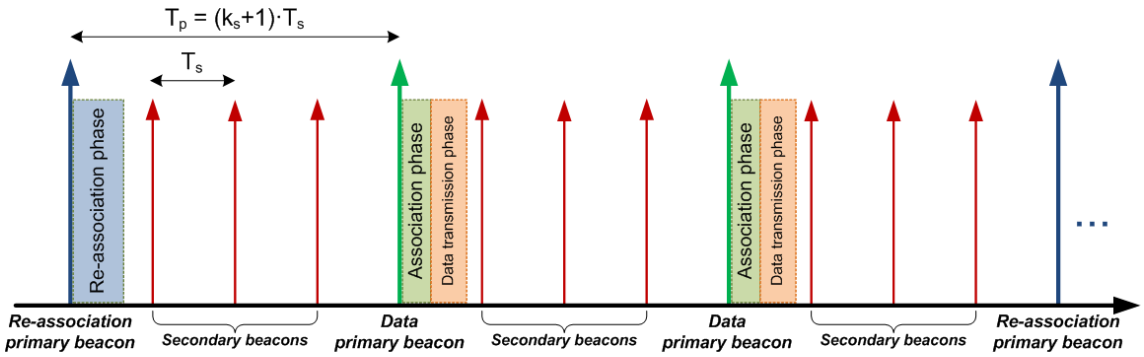


Figure 2.39: ENTOMATIC beacon scheduling

The TDMA-like scheme of the ENTOMATIC system is based on cycles, that is, when the permanent regime is established, STAs sequentially enter in a set of states every cycle. In Figure 2.40, a TDMA scheme for one complete cycle corresponding to a data (or statistics) beacon is shown. It is worth noting that the *action cycle* is triggered by the last (i.e., the primary) beacon sent by the GW. The description of the time variables included in the scheme is depicted in Table 2.28.

At the top of Figure 2.40, the general procedure of the cycle is represented, depicting each of the phases composing it. At the bottom, the transition between different operation states in two selected parent nodes (N_1 from ring 1, and N_2 from ring 2) is shown. A comprehensive description of the different operation states is included in Subsection 2.5.3.2.

[10] More information about the association procedure can be found in Section 2.6.4

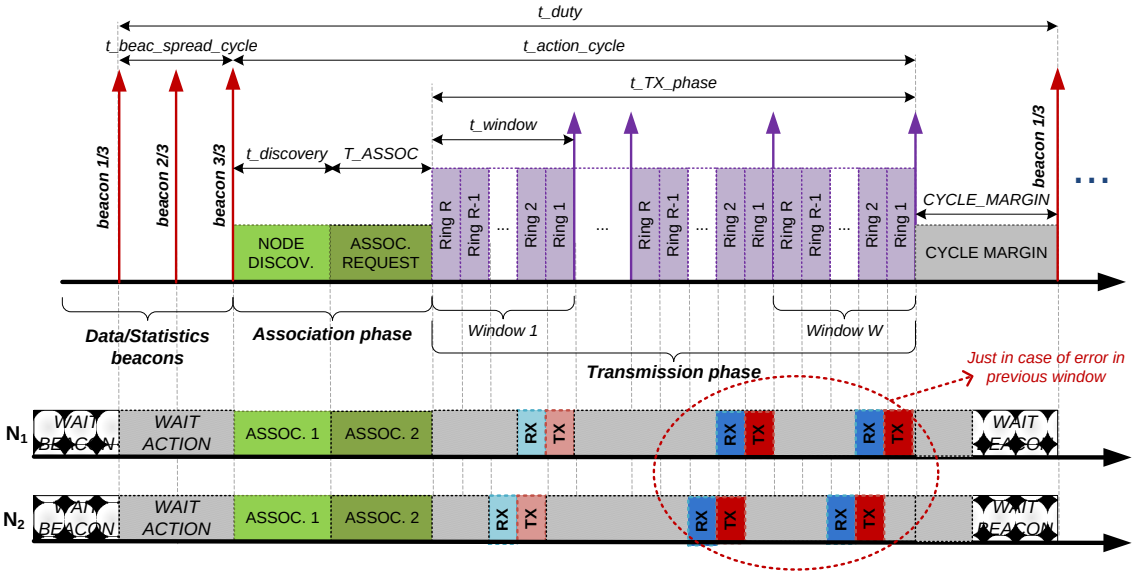


Figure 2.40: Data/Statistics beacon scheduling

The general behaviour of the scheduling system is explained below:

1. The GW generates and sends a primary beacon (asking for new data or statistics) containing relevant parameters regarding the scheduling system. STAs associated to this GW are awake and waiting for listening to new beacons. The STAs, according to the information contained in the beacon, synchronize their clocks, set their state variables, and therefore are able to interact properly with the rest of the WSN.

2. After listening to the primary beacon, the STAs enter in the association phase, which is divided into two states:

    - NODE DISCOVERY: to request or attend discovery request for potential new STAs.

    - ASSOCIATION REQUEST: to provide those new STAs that asked for being associated with a routing path to the gateway.

3. When the association phase finishes, the first transmission window begins. During a transmission window, STAs will listen to new packets from their children their previous ring turn (r - 1), and will transmit their own segments (along with other possible aggregated data payloads) in their own ring turn (r).

4. When a transmission window is finished, all STAs will be awake in order to listen to the e2eACK, which determines whether the data packet from each of the STAs has successfully arrived.

5. If the e2eACK was the last one for the current cycle (i.e., the previous transmission window was the last one), all STAs would enter into the SLEEP state. Some seconds before the first beacon of the next cycle (defined by a time gap variable) they would wake up and enter in the WAIT BEACON state.

   Instead (i.e., the e2eACK was not the last one for the current cycle), if there were more transmission windows pending, STAs that became poisoned[11] during the previous window would perform similarly, listening to children and transmitting their own or aggregated data.

Table 2.28: Scheduling variables

Concept	Variable	Value	Description

[11] See Poisoning system chapter (subsection 2.7.2.2)

Association	*DISC_RANGES_ASSOC*	-	Number of discovery ranges
	T_DISC_RANGE	-	Time of a discovery range
	t_discovery	*DISC_RANGES_ASSOC * T_DISC_RANGE*	Time of node discovery phase
	T_ASSOC	-	Time of association request phase
Transmission	*num_windows* (*W*)	-	Number of TX windows
	T_RING	-	Time of a TX ring
	num_rings (*R*)	-	Number of rings
	GAP_E2EACK	-	Time gap for end-to-end ACK
	t_window	*T_RING * num_rings + GAP_E2EACK*	Time of a TX window
	t_TX_phase	*num_windows * t_window*	Time of the full TX phase
Cycle margin	*CYCLE_MARGIN*	-	Time of the cycle margin
Beacon spreading	*t_action_cycle*	*t_discover + T_ASSOC + t_TX_phase*	Time of the action cycle
	t_duty	*t_action_cycle + CYCLE_MARGIN*	Time of a complete cycle
	t_beac_spread_cycle	*t_duty - t_action_cycle - CYCLE_MARGIN*	Time of beacon spreading

2.5.3.2 Summary of operation states

The overall behavior of the ENTOMATIC code is based on states. A function running periodically, called *periodic()*, determines which is the operation state of the STA at any moment. Then, actions corresponding to the given state are performed. The possible operation states of an STA are:

- **WAIT BACON**: STAs are turned ON and wait for listening to a beacon. The programmed behavior is to wake up 2-3 seconds before the GW sends the beacon to ensure its reception. If the first beacon is not received, STAs will remain awake until receiving another one.
- **WAIT ACTION**: STAs have their radio module OFF until they turn it ON some seconds before receiving the primary beacon, which triggers the start of an action cycle. STAs know the specific moment they will have to be awake for receiving the primary beacon by means of a time field included in the secondary beacons.
- **Association phase states**:
    - **NODE DISCOVERY**: **Non-associated STAs** send a parent node discovery. From the received responses, they determine which is the most suitable parent by means of a parent score metric that depends on several parameters (e.g., the RSSI or the ring of the potential parent). Already **associated STAs** remain awake and response to possible node discovery requests from neighbor STAs.
    - **ASSOCIATION REQUEST**: **Non-associated STAs** send an association request to the best node computed during the NODE DISCOVERY phase. This request is in turn forwarded until reaching the GW, which answers through a broadcast message. This way, nodes forwarding the petition know whether they have a new child in their path. Already **associated STAs**, instead, remain awake whether they have listened to a discovery petition (and answered) or not (just in case that some of their children were asked to provide a route to a new STA).
- **WAIT TX**: This state is activated when the STA belongs to the immediate lower ring. That is, parents should be awake in order to listen to packets coming from their children. If an STA has no children, it will not enter in this state. Similarly, if an STA has received all the packets from its children, it will go to the SLEEP state.
- **TX**: STAs belonging to the current transmission ring will transmit to their parents those packets containing the payloads received from their children as well as the ones generated by them. Once the transmission is finished and the ACK received, these STAs enter in the SLEEP state.

- **WAIT e2eACK**: All the STAs should be awake before the GW sends the end-to-end ACK corresponding to each transmission window.
- **SLEEP**: Radio is OFF and only the periodic function keeps running in order to determine when a change of state is required.

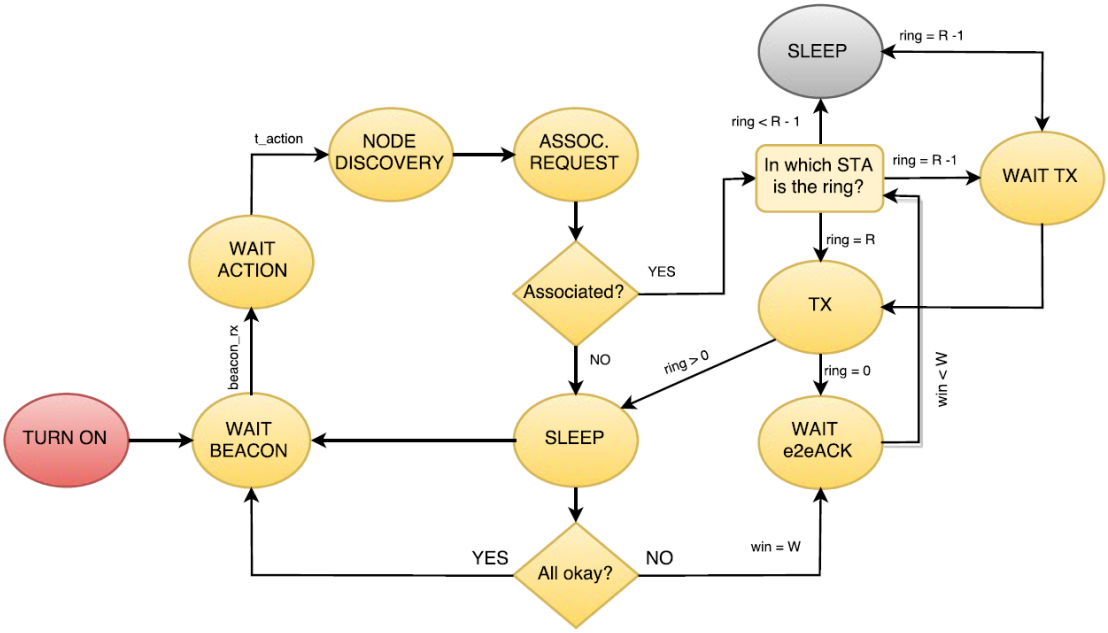


Figure 2.41: General diagram of operation states in ENTOMATIC stations

2.5.4 WAKEUP PATTERNS

A wakeup pattern is a set of instructions generated by the GW which indicates the wakeup plan for sensor nodes over several slot times. Due to the ring structure of the considered network and the different power constraints of GW and STAs, the wakeup pattern depends on the traffic flow [31].

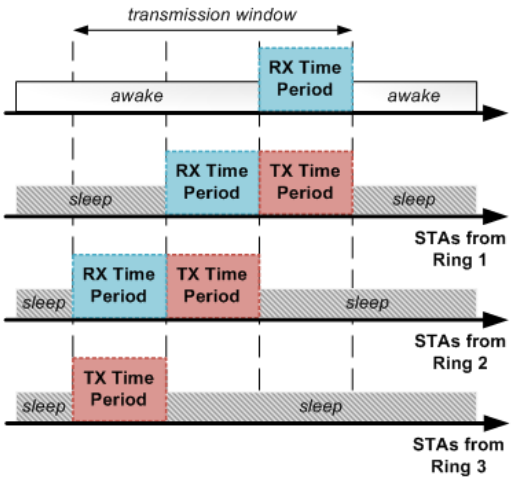


Figure 2.42: Example of a *staggered wake-up pattern* in a 3-ring wireless network

The *periodic wakeup pattern* is suitable for listening to broadcast downlink communications from the GW, as it makes all STAs wake up at the same time. On the other hand, uplink communications follow a *staggered wakeup pattern*, which allocates different active periods to nodes belonging to adjacent rings with partial overlapping (as shown in Figure 2.42). This reduces time STAs are awake during uplink communications and facilitates the implementation of data aggregation mechanisms.

2.5.5 POWER REGULATION MECHANISM

The selection of the most appropriate power transmission level according to the closeness of neighbor nodes is managed through a mechanism based on the RSSI. For this purpose, a safety margin for reliable communications is defined by $RSSI_{min}$ and $RSSI_{max}$.

If an STA is transmitting data packets/ACKs to its parent/child at a power level making the received RSSI higher than $RSSI_{max}$, it will be asked to decrease it for the next transmission. Similarly, if the received RSSI is lower than $RSSI_{min}$, it will be asked to increase it.

In the ENTOMATIC system, recommended values of RSSI are between the following values:

$$RSSI \in [-110, -100]\ dBm$$

Power regulation requests are included in the RSSI control field of data packets and ACK headers. Possible values of this field are: *increase (3)*, *keep (2)*, and *decrease (1)*. After receiving the petitions of its parent and children, an STA determines whether and how to regulate its own power level depending on the following considerations:

- If any STA asks for a raise, increase the power level.
- If all STAs ask for a reduction, decrease the power level. There must be unanimity to not affect other STAs.
- Otherwise, keep the current power level.

Data pkt	Packet id				RSSI control		Agg?	Total segments		Current segment	
	0	0	0	1	x	x	1	1	0	0	1

Figure 2.43: Data packet frame structure with the RSSI control field, responsible of managing the Power Regulation Mechanism

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

2.5.5.1 Crossbow TelosB

The TI CC2420 transceiver of the Crossbow TelosB counts with 8 different power levels ranging from -25 to 0 dBm. These levels are shown in the table below:

Table 2.29: TI CC2420 transceiver power levels

Power level	cc2420_set_txpower() argument	Transmission Power [dBm]
1	3	-25
2	7	-15
3	11	-10
4	15	-7
5	19	-5
6	23	-3
7	27	-1
8	31	0

2.5.5.2 Zolertia Re-Mote

The TI CC1200 transceiver of the Zolertia Remote counts with 31 different power levels ranging from -16 to 14 dBm (in steps of 1 dBm). These levels are shown in the table below:

Table 2.30: TI CC1200 transceiver power levels

Power level	Transmission power [dBm]	Power level	Transmission power [dBm]
0	-16 dBm	**16**	0 dBm
1	-15 dBm	**17**	1 dBm
2	-14 dBm	**18**	2 dBm
3	-13 dBm	**19**	3 dBm
4	-12 dBm	**20**	4 dBm
5	-11 dBm	**21**	5 dBm
6	-10 dBm	**22**	6 dBm
7	-9 dBm	**23**	7 dBm
8	-8 dBm	**24**	8 dBm
9	-7 dBm	**25**	9 dBm
10	-6 dBm	**26**	10 dBm
11	-5 dBm	**27**	11 dBm
12	-4 dBm	**28**	12 dBm
13	-3 dBm	**29**	13 dBm
14	-2 dBm	**30**	14 dBm
15	-1 dBm		

2.6 NETWORK LAYER

In the Open Systems Interconnection (OSI) communications model, the Network Layer knows the address of some neighbours' nodes in the network, packages output with the correct network address information, selects routes and quality of service, and recognizes and forwards to the Transport layer incoming messages for local host domains.

The network layer provides the functional and procedural means of transferring variable-length data sequences from a source to a destination host via one or more networks, while maintaining the quality of service functions. The main functions of the network layer are:

- **Connection model**
  For example, IP is connectionless, in that a datagram can travel from a sender to a recipient without the recipient having to send an acknowledgement. Connection-oriented protocols exist at other, higher layers of the OSI model.

- **Addressing system**
  Every host in the network must have a unique address that determines where it is. This address is normally assigned from a hierarchical system.

- **Packet forwarding**
  The packet forwarding is the relaying of packets from one network segment to another by nodes in a computer network. The main forwarding strategies are the following:
  - Unicast: It is the term used to describe communication where a piece of information is sent from one point to another point. In this case there is just one sender, and one receiver.
  - Broadcast: In this case, information is sent from one point to all other points. There is just one sender, but the information is sent to all connected receivers.

- Multicast: Here the information is sent from one or more points to a set of other points. In this case there is may be one or more senders, and the information is distributed to a set of receivers

Within the service layering semantics of the OSI network architecture, the network layer responds to service requests from the transport layer and issues service requests to the data link layer.

2.6.1 NETWORK LAYERS FOR WIRELESS SENSOR NETWORKS

The network layer in wireless sensor networks has a lot of challenges; namely, the power saving, limited memory and buffers, and a system to uniquely identify a node within the network. In addition, its major function is the routing protocol, whose basic idea is to define a reliable path (and a recovery system to make the system reliable against failures) from every source of packets to its corresponding destination.

For more information about the state-of-the-art regarding network layers for wireless sensor networks, we kindly refer the reader to the subsections 2.6.5: Data transmission, aggregation and segmentation and 2.6.6: Routing systems in Wireless Sensor Networks, where a comprehensive taxonomy of existing mechanisms in each field is provided.

2.6.2 OVERVIEW OF THE ENTOMATIC NETWORK LAYER

Communication in the ENTOMATIC environment follows a centralized scheme, where the gateway has the main role in the network by taking responsibility for managing associations to the network, scheduling data transmissions from stations, acknowledging correct data reception and broadcasting network status.

Stations are passive elements of the network and cannot communicate with themselves; all transmissions are addressed to the gateway, even though these often have to pass through other stations in their way. Conversely, the gateway can make use of its greater transmission power to periodically send broadcast messages addressed simultaneously to all stations, or send unicast messages directly addressed to single stations.

Communication in ENTOMATIC is connectionless, this means that a message can be sent from one end point to another (from an STA to the gateway) without prior arrangement. The device at one end of the communication transmits data addressed to the other, without first ensuring that the recipient is available and ready to receive the data.

In typical connectionless transmissions the service provider usually cannot guarantee that there will be no loss, error insertion, misdelivery, duplication, or out-of-sequence delivery of the packet. However, some upper layer protocols have been implemented in the ENTOMATIC system to solve problems related to losses, error insertion and duplication. In fact, a double mechanism for guaranteeing data transmissions has been developed, so that data acknowledgements are performed, both in link (i.e., between two consecutive hops) and transport layers (i.e., in an end-to-end approach, between the two extremes of communication).

2.6.3 ADDRESSING SYSTEM

The addressing system of the ENTOMATIC network follows the RIME format [32]. In a very similar way to IP addressing, any RIME address can be split into decimal numbers.

In the case of a RIME address, it is split into two 8-bit decimal numbers: the network identifier (A), and the node identifier (B).

$$RIME = (A.B)$$
$$A \in [0{,}255], \qquad B \in [0{,}255]$$

- **A: Network identifier**
  This number uniquely identifies an STA in a network and is randomly and uniformly generated by the GW covering the range from 1 to 127. As an example, on a scenario with two GWs, one could have the address 24.0 and the other one 59.0. Thus, all the STAs associated to a same GW will have the same network identifier (59.1, 59.2, 59.3, etc.).

- **B: Node identifier**
  By system definition, all GWs have a node identifier equal to 0. As for the STAs associated to a GW, they have a node identifier ranging from 1 to the maximum number of STAs allowed by that GW.

By default, when STAs are powered on, take a random and temporary address which is only changed by the definitive network address after they have been admitted by the gateway. Likewise STAs take a random address when receiving a re-association beacon or after realizing they have been removed from the network. To not share the same addressing range, STA temporary addresses can only take A values from 128 to 255, and B values from 0 to 255.

2.6.4 ASSOCIATION SYSTEM

The association mechanism proposed in the ENTOMATIC network is also managed by the gateway. This device is responsible of accepting new stations, giving them a unique network identifier (a network address) and, if necessary, removing them from the stations' register.

The system is able to admit new stations by two different mechanisms: a scheduled, global active one, called ***re-association mechanism***; and a passive, singular one, called ***association mechanism***.

2.6.4.1 Connection test method

Prior to the execution of any association mechanism, the first action performed by an STA as soon as it has been switched on is to check if there is any gateway available within its coverage range. This action is performed thanks to the connection test method, which sends a broadcast message that can only be answered by a gateway.

An available gateway will answer to this petition by sending a unicast message to the STA, including (among other parameters) the time until the next network beacon. When receiving this message, the STA will enter in a sleeping state for the specified time. In case the STA did not receive any response from a GW, it would be automatically switched off and a manual intervention would be required.

The implementation of this method allows STAs to save energy before being connected to the network, as they do not need to wait awake for the next beacon to become associated. In addition, it is useful for in-field installers, as they can know (thanks to a code color implemented with the leds: green -> OK / red -> No OK) if there is an available GW in the surroundings or not.

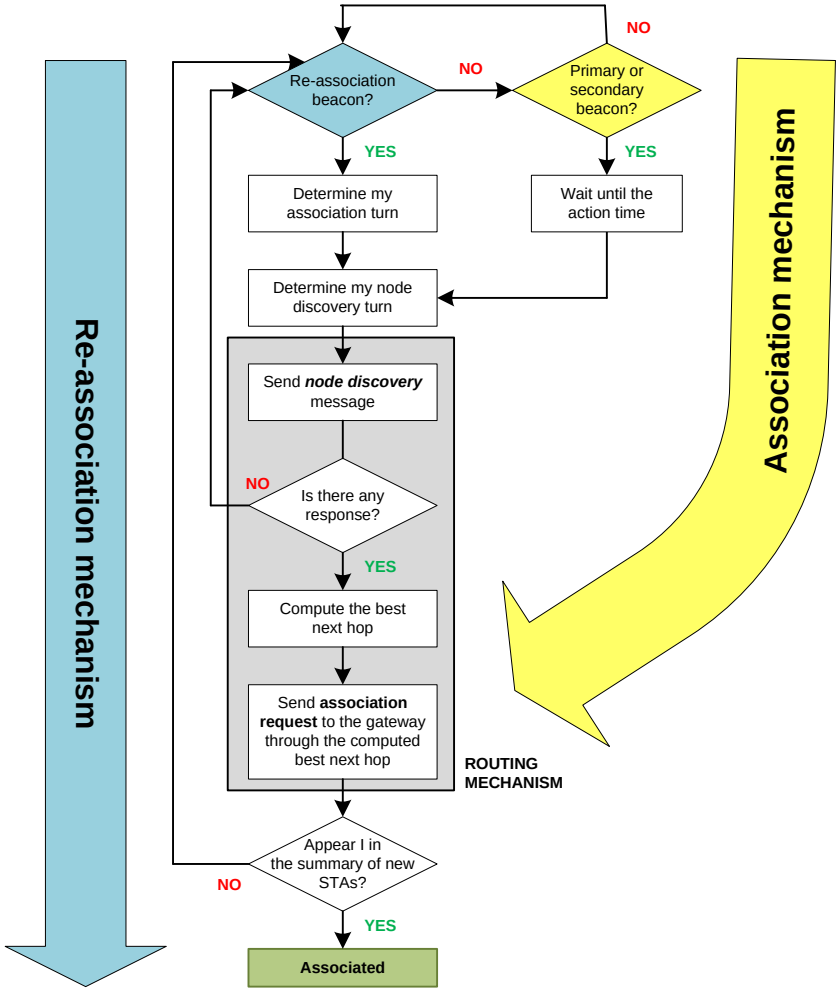


Figure 2.44: Diagram of re-association and association ENTOMATIC mechanism

2.6.4.2 Re-association mechanism

The ***re-association mechanism*** is intended to provide a tool for associating a large amount of stations to the network in a short period of time. Once the gateway is activated, or after a pre-determined number of primary beacons, the gateway broadcasts a special beacon, called ***re-association beacon***. When STAs listen to this beacon, reset their network address (whether they were previously associated or not) and start their particular re-association mechanism.

Their first step is to compute their association turns, because the system is designed to first process the association requests of closer stations. Thus, depending on the RSSI value received in the re-association beacon as well as some other configuration parameters, stations are able to determine their 'turn' to initiate the association process.

Table 2.31: Configuration parameters to determine the association turn

Configuration parameter	Description	Default values (linear method)	Default values (exponential method)	Default values (compressed method)
rssi_max	Maximum RSSI level of a received re-association beacon	-60 dBm	-60 dBm	-60 dBm
ranges	Total number of association turns	10	6	5
length	Value of the first association turn	10 dB	2 dB	5 dB

The configuration parameters for determining the association turn are the maximum RSSI level, the number of turns, and the length of those turns, as shown in Table 2.31. Three different mechanisms have been designed to compute this value: the linear, the exponential, and the compressed method:

- **Linear method**

  In the linear method, the whole range of RSSI values is split into a predetermined number of ranges, whose length is constant.

Table 2.32: Range of received RSSI values for each association turn when using the linear method
(It is worth noting that the first and the last turns also contain the upper and lower values out of the considered margin)

# Turn	Maximum received RSSI	Minimum received RSSI	Turn length (dB)
1	∞ dBm	-70 dBm	-
2	-70 dBm	-80 dBm	10
3	-80 dBm	-90 dBm	10
4	-90 dBm	-100 dBm	10
5	-100 dBm	-110 dBm	10
6	-110 dBm	-120 dBm	10
7	-120 dBm	-130 dBm	10
8	-130 dBm	-140 dBm	10
9	-140 dBm	-150 dBm	10
10	-150 dBm	-∞ dBm	-

Table 2.33: Function of the ENTOMATIC software code loaded in stations corresponding to the process to determine the association turn in the linear method

```
uint8_t calc_reassoc_range_linear
(int16_t rssi_rx, int16_t rssi_max, uint8_t ranges, uint8_t length)
{
        uint8 t i;
        int val;
        int16_t rssi_a, rssi_b;

        if (rssi rx < 0)
        {
                rssi_rx = -rssi_rx;
        }

        i = ranges;
        val = -1;

        rssi_a = rssi_max;
        rssi_b = rssi_a;

        if (rssi rx < rssi a)
        {
                val = 0;
        }
        else
        {
                for (i = 1; i < ranges + 1; i++)
                {
                        rssi_b = rssi_b + length;
                        if ((rssi_rx >= rssi_a) && (rssi_rx < rssi_b))
                        {
                                val = i - 1;
                        }
                        rssi a = rssi b;
                }
        }
        if (val == -1) {
                val = ranges - 1;

        }
```

```
        return (uint8_t) val;
}
```

- **Exponential method**

  In the exponential method, the length in dB of each turn is doubled in every round, so that final lengths are quite higher than the initial ones. With the default values, method the opportunities an STA has for sending a node discovery message are split into 6 different turns, depending on the RSSI received, as shown in the following table.

Table 2.34: Range of received RSSI values for each association turn when using the exponential method
(It is worth noting that the first and the last turns also contain the upper and lower values out of the considered margin)

# Turn	Maximum received RSSI	Minimum received RSSI	Turn length (dB)
1	∞ dBm	-62 dBm	-
2	-62 dBm	-66 dBm	4
3	-66 dBm	-74 dBm	8
4	-74 dBm	-90 dBm	16
5	-90 dBm	-122 dBm	32
6	-122 dBm	-∞ dBm	-

Table 2.35: Function of the ENTOMATIC software code loaded in stations corresponding to the process to determine the association turn in the exponential method

```
uint8_t calc_reassoc_range_complex
(int16_t rssi_rx, int16_t rssi_max, uint8_t ranges, uint8_t length)
{
        uint8 t i, val;
        int16 t rssi a, rssi b;

        if (rssi_rx < 0)
        {
                rssi_rx = -rssi_rx;
        }

        i = ranges;
        val = -1;
        rssi_a = rssi_max;
        rssi_b = rssi_a;

        if (rssi rx < rssi a)
        {
                val = 0;
        }
        else
        {
                for (i = 1; i < ranges + 1; i++)
                {
                        rssi_b = rssi_b + power(2, i - 1) * length;
                        if ((rssi_rx >= rssi_a) && (rssi_rx < rssi_b))
                        {
                                val = i - 1;
                        }
                        rssi_a = rssi_b;
                }
        }
        if (val == -1) {
                val = ranges;
        }
        return val;
}
```

- **Compressed method**

The compressed method is an empirical system aimed to optimize the process of selecting the most appropriate association turn. As its name indicates, it is a *compressed* version of the linear method with only 5 turns.

Table 2.36: Range of received RSSI values for each association turn when using the compressed method
(It is worth noting that the first and the last turns also contain the upper and lower values out of the considered margin)

# Turn	Maximum received RSSI	Minimum received RSSI	Turn length (dB)
1	∞ dBm	-90 dBm	-
2	-90 dBm	-95 dBm	5
3	-95 dBm	-100 dBm	5
4	-100 dBm	-105 dBm	5
5	-105 dBm	-∞ dBm	-

Table 2.37: Function of the ENTOMATIC software code loaded in stations corresponding to the process to determine the association turn in the compressed method

```
uint8_t calc_reassoc_range_compress
(int16_t rssi_rx, int16_t rssi_max, uint8_t ranges, uint8_t length)
{
        uint8 t val;

        if (rssi_rx < 0)
        {
                rssi rx = -rssi rx;
        }

        val = -1;

        if (rssi_rx < 90)
        {
                val = 0;
        }
        else if ((rssi_rx >= 90)&&(rssi_rx < 95))
        {
                val = 1;
        }
        else if ((rssi rx >= 95)&&(rssi rx < 100))
        {
                val = 2;
        }
        else if ((rssi_rx >= 100)&&(rssi_rx < 105))
        {
                val = 3;
        }
        else
        {
                val = 4;
        }

        if (val == -1)
        {
                val = ranges;
        }
        return val;
}
```

Once its association turn is determined, the routing mechanism is initialized. Due to its critical nature, the ENTOMATIC routing mechanism is detailed in section 2.6.7. As for the association process, once the routing mechanism is completed, the gateway receives the association request from the STA (after passing through the different hops of the newly selected routing path) and determines its possible joining to the network. At this point, the gateway basically confirms if the number of already associated stations has not achieved its maximum value and, therefore, a new station can be associated.

Instead of sending a unicast message per each STA informing about its association to the network, the gateway transmits a single summary broadcast message immediately after the end of every association turn. With this message, different goals of the system are achieved:

- STAs waiting for being associated receive a response.
- If that is the case, STAs receive a network address, the confirmation of the next hop address to send any data message, its situation in the network in terms of rings, and its number of children.
- Besides, nodes which from now on will act as parents also receive the confirmation from the gateway.
- Lastly, STAs that have unsuccessfully tried to associate the network in their association turn may use the consecutive turns until they became associated (or the number of turns is over).

From this moment on, the STA remains associated and will periodically send the data collected by its sensors.

2.6.4.3 Association mechanism

The ***association mechanism*** is intended to provide an association solution to those specific nodes that have not been able to associate themselves to the network during the re-association mechanism, have been powered between two re-association beacons, or simply suffer routing problems due to failures in their routing path.

The mechanism follows the same pattern as the re-association one, with the single exception that there is only one association turn located immediately after the last secondary beacon.

2.6.4.4 Disassociation mechanism

Due to the centralized nature of the system, not only associations but also disassociations must be controlled by the network. To do this, a ***disassociation mechanism*** has been also designed, so that newly active STAs may use the network address of inactive ones.

Inactive or erratic STAs shall be removed from the network register and the routing table as soon as possible to ensure the correct reception of data packets from the rest of the network stations as well as to allow new STAs to enter into the system.

By means of the disassociation mechanism, the gateway removes an STA from the network if it does not receive any data packet from it during a pre-determined number of consecutive primary beacons.

$$T_{disassociation} = k \cdot T_{primary\_beacon}$$

Similarly as in the broadcast message used to notify the newly associated STAs to the network, the gateway may attach a list of disassociated STAs within the primary beacons.

Once STAs realize that they have been removed from the network, they can request again for an association by performing the re-association or the association mechanism. Besides, these messages are also useful for parents, as they can check if all their children are still alive. Lastly, this mechanism is very useful for highly variable environments, where network topology suffers continuous changes.

2.6.4.5 Self-disassociation mechanism

Finally, a ***self-disassociation mechanism*** has been included to save energy in case an STA has lost all connection with the gateway of the network. The goal of this mechanism is to avoid repetitive association requests and other energy consumption procedures that could make the STA run out of battery when no connection with the gateway is possible.

All STAs have a timer that is activated after receiving any beacon from the GW. From that moment on, if an STA does not receive any other beacon during a determined period ($T_{disassociation}$), it turns off its

lights, its radio module, and the battery supply. The STA is thus considered 'dead' and it will need to be activated again by manual procedures.

$$T_{disassociation} = l \cdot T_{duty}$$

The $T_{disassociation}$ field is included in every re-association beacon. Its value is set by default as 2*T_{duty}. For further developments, where T_{duty} can be dynamically changed, it is advised to do this only before sending a new re-association beacon. Otherwise, STAs could deactivate themselves by not properly updating their $T_{disassociation}$ value.

2.6.5 DATA TRANSMISSION, AGGREGATION AND SEGMENTATION

In common IT applications, data aggregation refers to the process of compiling information from databases with intent to prepare combined datasets for data processing. In WSNs, data aggregation techniques use different node parameters to select and store data attributes in an aggregated format for further evaluation and usage [33]. In multi-hop WSNs, data aggregation is performed in a distributed way, so that all nodes are responsible for performing these techniques over the received data.

In Figure 2.45 it can be seen an example of data aggregation in WSNs. This network consisting of 4 different clusters has one only sink, which is the final data destination. Nodes with new data to transmit (n_1, n_7, n_{11}, and n_{14}) send a data message to their next hop in the path to the sink. If the receiver also has new data, *aggregates* it with the received and sends all information together to its next hop. Otherwise, it just retransmits the received data to the next hop.

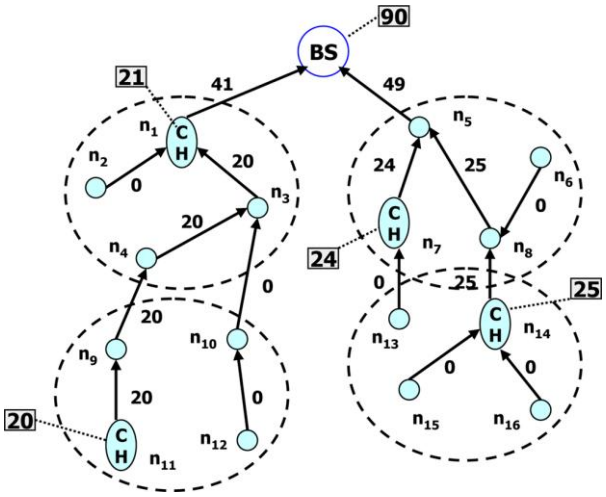


Figure 2.45: Data aggregation from four clusters [34]

Previous research tells us that whether the sources are clustered near each other or located randomly, significant energy gains are possible with data aggregation [35][36]. These gains are greatest when the number of sources is large, and when the sources are located relatively close to each other and far from the sink. The modelling, though, also seems to suggest that aggregation latency could be non-negligible and should be taken into consideration during the design process.

Table 2.38: Taxonomy of data aggregation techniques in hierarchical WSNs

Data aggregation techniques in hierarchical WSNs	Examples of Protocols
Cluster-based	LEACH (Low-Energy Adaptive Clustering Hierarchy)
	EECDA (Energy Efficient Clustering and Data Aggregation)
	CAG (Clustered Aggregation)
Chain-based	PEGASIS (Power-Efficient Gathering in Sensor Information System)

	CHIRON (Chain-Based Hierarchical Routing Protocol)
	COSEN (Chain Oriented Sensor Network for Efficient Data Collection)
Tree-based	TREEPSI (Tree-based Efficient Protocol for Sensor Information)
	TCDGP (Tree-Clustered Data Gathering Protocol)
	PERLA (Power Efficient Routing with Limited Latency)
Grid-based	ATCBG (Aggregation Tree Construction Based on Grid)
	GROUP (Grid-clustering Routing Protocol for Wireless Sensor Networks)

In hierarchical networks, there are four strategies for data aggregation based on network architecture: cluster-based, chain-based, tree-based and grid-based [37][38].

- **Cluster-based**
  In these networks, sensor nodes are organized in the form of clusters and transmit information to a cluster head. There, all data received from sensors is aggregated and then transmitted to the sink. The cluster head can communicate with the sink either through long range transmissions or multi-hoping via other cluster heads. Thus this process results in energy savings and is mainly useful for energy-constrained sensors.

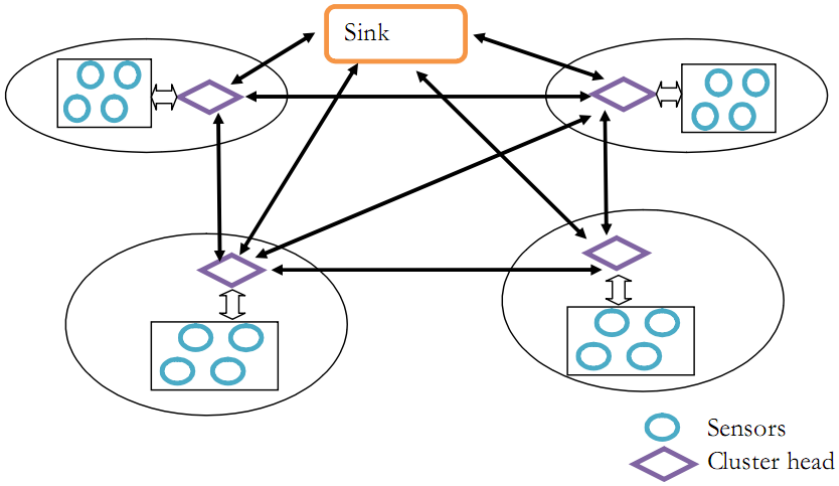


Figure 2.46: Cluster-based data aggregation technique

- **Chain-based**
  If the cluster head is far away from the sensors, they might expend excessive energy in communication. Further improvements in energy efficiency can be obtained if sensors transmit only to close neighbors. The key idea behind chain based data aggregation is that each sensor transmits only to its closest neighbor. In this process, nodes are connected in the form of chains for data transmission to the cluster head.

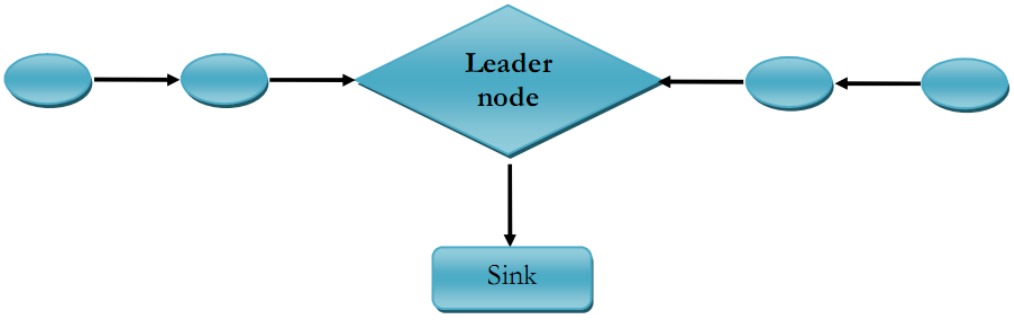


Figure 2.47: Chain-based data aggregation technique

- **Tree-based**
  In a tree based network, sensor nodes are organized into a tree, where data aggregation is performed at intermediate nodes along the tree and a concise representation of the data is transmitted to the root node. Here data flow starts at leaves nodes and ends at the sink. The main aim of the tree based approach is constructing an energy efficient data aggregation tree.

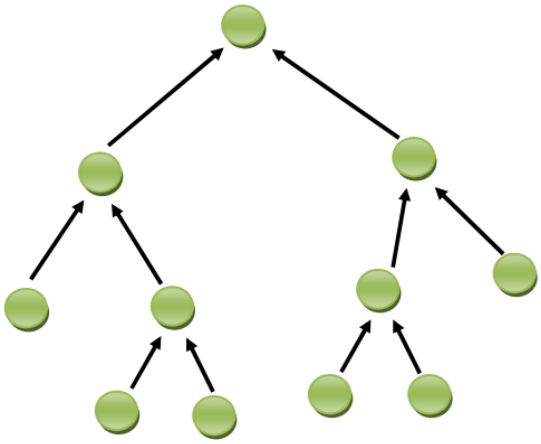


Figure 2.48: Tree-based data aggregation technique

- **Grid-based**
  In this case, the sensor network environment is divided into pre-defined set of grids or regions. Each grid is responsible for observing and reporting events that may occur inside the region to the sink nodes. In addition, one sensor device based on geographical position with respect to either the sink or the center of the grid is chosen as data aggregator. All other sensors inside the gird are aware of this information. During event detection, all other sensors are supposed to send the event information to this data aggregator. The data aggregator after collecting data from other sensors sends only the critical information to the sink node.

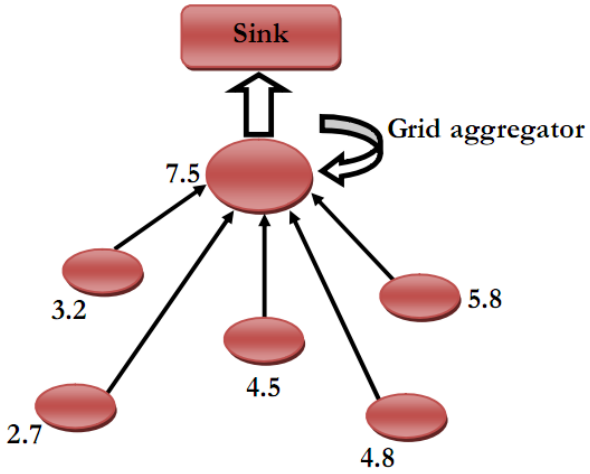


Figure 2.49: Grid-based data aggregation technique

The aggregation is, therefore, the global process of gathering and routing information through a multi-hop network and processing data at intermediate nodes with the objective of reducing power consumption. As for the possible data reduction in processing and management, there are two approaches in data aggregation techniques:

- **With size reduction**

It refers to the process of combining and compressing the data packets received by a node from its neighbors in order to reduce the packet length that is to be transmitted or forwarded towards the sink.

- **Without size reduction**
  It refers to the process of merging data packets received from different neighbors into a single data packet but without processing the value of data.

2.6.5.1 Data Aggregation in ENTOMATIC network

Downlink communications are generally executed through broadcast messages from the GW, so that no data aggregation mechanisms are required. Conversely, uplink communications are unicast and follow a multi-hop path over a tree-based network.

In the latter case, the staggered wakeup pattern fits perfectly with the approach of data aggregation in WSN. Thus nodes attach their own data to that received from their children and all the information is jointly sent to the next hop. If the total amount of data aggregated by an STA exceeds the maximum payload supported by the hardware, it is split into segments[12] sent consecutively.

A selective ACK (SACK) mechanism has been developed, so that before the end of the allocated time slot, the receiver informs the sender of the segments properly received. Upper layers will be responsible for only sending the remaining segments in corresponding time slots from successive transmission windows.

2.6.6 ROUTING SYSTEMS IN WIRELESS SENSOR NETWORKS

Traditional routing protocols used in traditional networks (wired or wireless) are proved to be ineffective in wireless sensor networks. This is mainly due to the strong differences between both kind of networks:

- The number of nodes in a WSN can be several times higher than in a traditional one
- Sensor nodes are densely deployed on the monitored field
- Sensor nodes are error-prone
- Network topology in a WSN varies far more frequently (even depending on the running application)
- Traditional networks are managed by humans, while WSN are autonomous
- Sensor nodes have some constraints in terms of power, processing, memory and battery resources
- There are even WSN whose sensor nodes cannot be identified by means of an ID number, due to the huge number of deployed nodes as well as problems related to the header length

Thus, specific features of WSN have fostered research on routing mechanisms capable of overcoming its current limitations. A first classification of routing protocols WSN may be done depending on how the routing information is transmitted:

- **Proactive protocols**
  Routing information is transmitted periodically, thus updating routes to all possible destinations and making easier communication between nodes anytime.

- **Reactive protocols**
  Conversely, routes are created here only when necessary.

- **Hybrid protocols**
  They are a combination of proactive and reactive protocols.

[12] The amount of data aggregated by an STA (from itself and from its children) is called *packet*. If this packet is split into different parts, each one of these parts is called *segment*.

Another way to classify routing protocols in WSN is the information they use to create the different routes. As stated in [39] there are 7 categories:

- **Location-based protocols**
  In location-based protocols, sensor nodes are addressed by means of their locations. Location information for sensor nodes is required for sensor networks by most of the routing protocols to calculate the distance between two particular nodes so that energy consumption can be estimated. In this section, we present a sample of location-aware routing protocols proposed for WSNs.

- **Data centric protocols**
  Data-centric protocols differ from traditional address-centric protocols in the manner that the data is sent from source sensors to the sink. In address-centric protocols, each source sensor that has the appropriate data responds by sending its data to the sink independently of all other sensors. However, in data-centric protocols, when the source sensors send their data to the sink, intermediate sensors can perform some form of aggregation on the data originating from multiple source sensors and send the aggregated data toward the sink. This process can result in energy savings because of less transmission required to send the data from the sources to the sink.

- **Hierarchical protocols**
  Many research projects in the last few years have explored hierarchical clustering in WSN from different perspectives. Clustering is an energy-efficient communication protocol that can be used by the sensors to report their sensed data to the sink. In this section, we describe a sample of layered protocols in which a network is composed of several clumps (or clusters) of sensors. Each clump is managed by a special node, called cluster head, which is responsible for coordinating the data transmission activities of all sensors in its clump.

- **Mobility-based Protocols**
  Mobility brings new challenges to routing protocols in WSNs. Sink mobility requires energy efficient protocols to guarantee data delivery originated from source sensors toward mobile sinks. In this section we discuss sample mobility-based routing protocols for mobile WSNs.

- **Multipath-based Protocols**
  Considering data transmission between source sensors and the sink, there are two routing paradigms: single-path routing and multipath routing. In single-path routing, each source sensor sends its data to the sink via the shortest path. In multipath routing, each source sensor finds the first k shortest paths to the sink and divides its load evenly among these paths.

- **Heterogeneity-based Protocols**
  In heterogeneity sensor network architecture, there are two types of sensors namely line-powered sensors which have no energy constraint, and the battery-powered sensors having limited lifetime, and hence should use their available energy efficiently by minimizing their potential of data communication and computation.

- **QoS-based Protocols**
  In addition to minimizing energy consumption, it is also important to consider quality of service (QoS) requirements in terms of delay, reliability, and fault tolerance in routing in WSNs. In this section, we review a sample QoS based routing protocols that help find a balance between energy consumption and QoS requirements.

After having analyzed the features of the considered protocols as well as the requirements of the ENTOMATIC network, some considerations about the routing protocol have been taken:

- All network routes will address messages to a single sink: the gateway.

- The routing protocol shall be reactive to the nodes, so that all routing operations must have been previously initiated by the gateway.
- The routing protocol shall contain periodic re-routing rounds, which will let the network adapt to the changing environment. Even if nodes have stable routes, they will look for a better route in case a re-routing round has been initiated.
- Routes shall be stable between two re-routing rounds, unless a node failure has been previously detected. In that case, surrounding nodes affected by the failure will initiate a new routing process.
- Intermediate sensors shall contain effective mechanisms to aggregate data from source sensors. This will lead to energy savings in the data transmissions from the sources to the sink, as in the ***data-centric protocols***.
- The process to compute the best route from a node to the gateway shall take into account a set of values: the RSSI transmitted to the next hop, the RSSI received from the next hop, the number of hops to the gateway, and the number of children of the next hop.
- In case an STA detects quality degradation in its link to the next hop, it shall initiate a re-routing mechanism in a slot specifically allocated from the gateway to this purpose.

2.6.6.1 Data-Centric routing protocols

Data-centric protocols, by implementing new mechanisms of request and data aggregation are responsible of reducing unnecessary energy consumption resulting from redundancy transmissions. Thus, the base station performs data requests based on attributes to certain regions of the network and waits to receive data which are gradually being added by sensors that meet the required characteristics. In the following lines the main data-centric routing protocols for WSN will be presented:

SPIN

SPIN [40][41] is a family of adaptive protocols for WSNs. The fundamental idea is to classify SPIN data obtained by the sensors by high-level descriptors (metadata) and conduct negotiations with the metadata before performing the transmission of collected data.

The operation is as follows: when a node receives new data, it announces it to its neighboring nodes so that, if they are interested, they send a request for the data. At that moment a metadata negotiation starts, thus avoiding the typical problems of other routing systems, such as redundant information, data implosion or overlap.

In addition, SPIN based routing consumes much less energy because the metadata, but are not specified by a standard and are specific for each application, are usually much smaller than the information that is intended to convey.

SPIN nodes use three different communication messages:

- ADV (Announcement of new data)
  When an SPIN node has data to share, announces with ADV message containing metadata such information.
- REQ (Request for data)
  An SPIN node sends a REQ message when it is interested in receiving the announced data.
- DATA (Data message)
  It contains the requested data as well as the metadata header.

As can be seen in the figure, SPIN operation is based on the following steps:

a) The node A has new data and sends a message to all its neighbors ADV (in this case, only B)
b) B is interested in the offered data and sends a request message to A.
c) A sends data to B.
d) B, upon receiving new data, announces it via broadcast to all its neighbors.
e) Interested neighbors ask B for the data.
f) B sends data previously received from A only to the interested nodes.

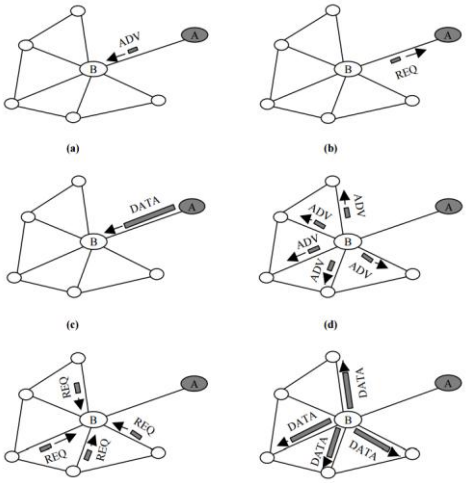


Figure 2.50: SPIN routing protocol operation [40]

One of the advantages of SPIN is that it is very robust to topological changes, since each node only needs to know which its neighbors are. Instead, its main drawback is that it is not always able to deliver information to all interested nodes. The explanation lies in the following figure, since the only interested nodes are not in the coverage area of the source, and those that are its neighbors are not interested.

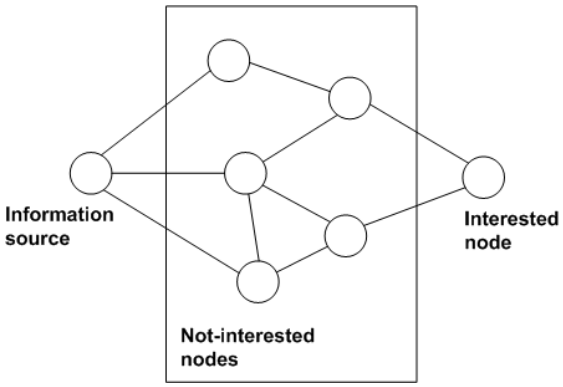


Figure 2.51: Hidden nodes problem in SPIN routing protocol

Therefore, this type of routing is not recommended for safety applications, in which a regular and reliable distribution of data packets is required. Nor for those scenarios where different stations do not maintain direct vision with other network nodes, such as in the case of ENTOMATIC.

However, it can be very useful in any application in which the requests are not regular and the network structure is sufficiently well formed to avoid hidden nodes.

Directed Diffusion

Directed diffusion [42] method is basically a mechanism of dissemination and data aggregation. All data is described by a list of pairs of attributes (similarly to the metadata in SPIN) that, unlike the previous mechanism, are requested by the information receiver or 'sink'.

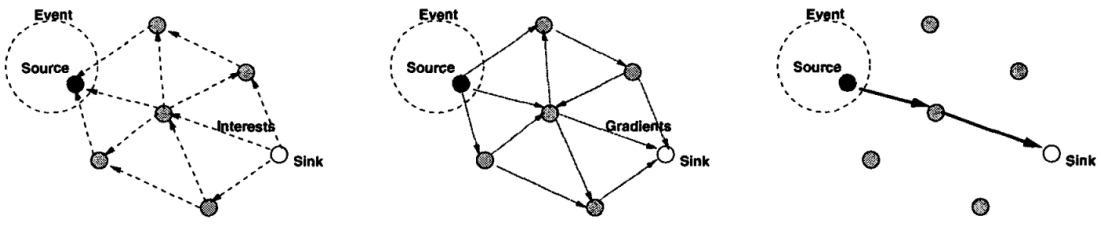


Figure 2.52: A simplified schematic for directed diffusion [42]

Thus, and as requests are spread throughout the network, gradients (or back roads) are created from source. In addition, this protocol allows the aggregation of data by nodes, which save in its memory previous requests, in order to be compared with the received data on its way to the sink node.

This is how the paths between sources and the sink node are created. In turn, sink reinforces these paths when resending through it formal data requests. This is how the reinforced path is created, which will become the main road, through which information will flow. In *directed diffusion* it is also possible to repair roads, because when one of these fails, a new or alternative path can be identified and subsequently reinforced, giving strength to the system.

All nodes that implement the direct broadcast are aware of the application that is running, so they can save energy by choosing the best roads, as well as keeping and processing data received from the network. Data storage effectively increases the efficiency, strength and coordination between network nodes.

The *directed diffusion* applications are many and varied, from finding best path between two IP sensors in a non-IP network, to the spontaneous notice of an important event to a particular section of a network. It is important to note that this type of routing mechanisms are especially recommended for applications in which the same requests are performed frequently, such as in environmental monitoring. Instead, it would not be worth searching the optimal path in other applications that rarely functioned on the basis of such requests.

Many other routing protocols derived from the original mechanism of direct broadcast, among which are: CADR, and ACQUIRE COUGAR.

Rumor routing

Rumor routing [43] protocol is another variation of the *directed diffusion*, used in scenarios where the amount of information to be exchanged is small and it is not possible to apply a criterion of geographic routing. The basic idea is to route typical *directed diffusion* requests to nodes that have observed a particular event, instead of flooding the entire network with messages to obtain information regarding the occurred events.

When a node detects an event, it adds it to a local table and generates an agent (long lasting entity) circulating throughout the network to spread information about the local event: mainly the type of event and the route to reach it.

Thus, when a node creates a request, it is propagated through the network until it reaches a node that has been visited by the 'agent', and has the ability to route the request to the appropriate event (see figure). From that moment, the route is established and the exchange of information starts.

Suitable applications where this routing method can be used are those characterized by having a small number of possible events with a huge number of nodes; i.e., decentralized applications.

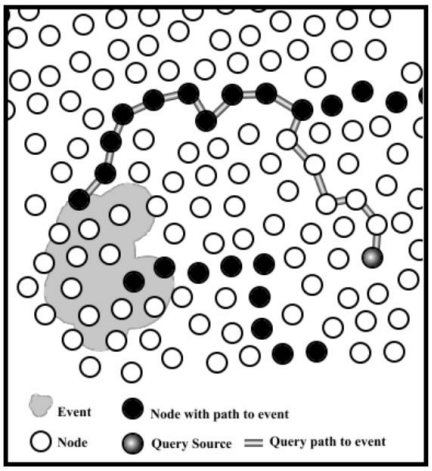


Figure 2.53: *Rumor routing* operation [43]
Query is originated from the query source and searches for a path to the event. As soon as it finds a node on the path, it's routed directly to the event

EAR

The objective of routing protocol EAR (Energy Aware Routing) [44] is to increase the lifetime of the network. For this purpose a set of paths through a suboptimal dependent probability function of energy consumption in each path are selected.

The operation is based on a first discovery of all possible routes between a source and a destination, where the energy cost of these routes is also calculated. Roads with high consumption are discarded, while the rest are introduced into a function whose probability is inversely proportional to the cost of each road.

The drawback of this mechanism is that the entire system must know the exact location of each node to calculate transmission costs, which may be overspending in some applications.

However, for applications where there is a low degree of mobility and where the lifetime of the network is critical, because of the inability to replace the sensors (for instance, applications on inaccessible terrains, war zones monitoring), EAR is the best choice.

2.6.7 OVERVIEW OF THE ENTOMATIC ROUTING PROTOCOL

The ENTOMATIC routing protocol has been designed as an intrinsic part of the association mechanism. Thus, once finished the process of association to a determined network, a newly associated STA will be able to exactly know the path its messages must follow to reach the gateway. While an STA is correctly associated to the network, it uses its previously computed routing path, which is only recomputed after a failure of the own STA or its immediate neighbors (parent or children). Indeed, no routing process is initiated unless it is part of a greater association process.

Another feature of the ENTOMATIC routing protocol is that STAs only know the identity of its immediate next hop (parent) and their previous hops (children). By splitting the association process into turns depending on the closeness to the gateway, STAs organize themselves by forming rings, so that the closest STAs to the gateway are part of the ring 1 and, therefore, send their messages directly in just one hop to the gateway. In turn, STAs of successive rings only have to know the network address of the node from their upper ring which best fit in that first hop, as the rest of the routing path will be determined by the own path of that node.

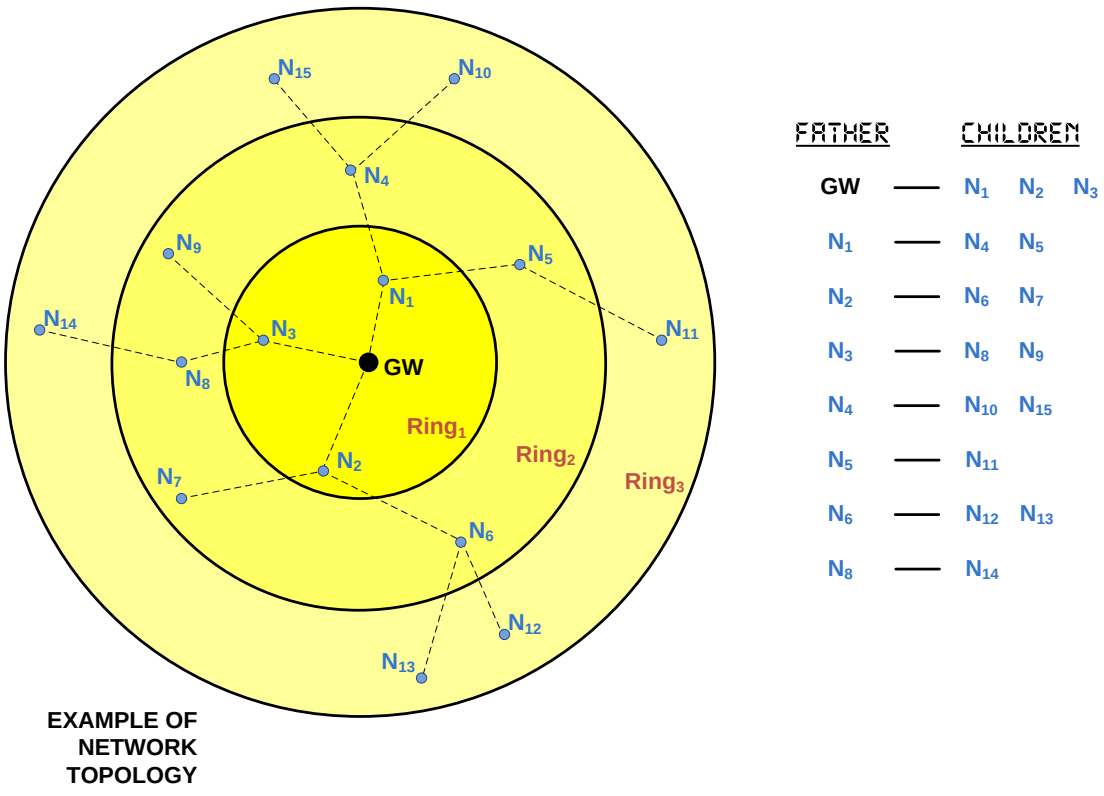


Figure 2.54: Example of ENTOMATIC network topology with 3 rings and 15 stations

As part of the ENTOMATIC network, the routing protocol is also centralized in the gateway, so that it has a global vision of the routing table, and confirmation of parent-child pairs is always notified via a broadcast message which is listened to by all nodes in the network. However, and with the aim of simplifying the computing operations of STAs, only routing information regarding the own STA is computed (i.e., no global updated routing table is maintained by nodes, but only by the gateway).

In spite of its simplicity, the ENTOMATIC routing protocol provides the network with a robust method to compute and, if necessary, recompute the different paths that packets must follow from their source node to their destination (i.e., the gateway). The exchange of routing packets among nodes in the network is minimum, thus minimizing the impact of these packets on energy consumption. In addition, the routing protocol is self-configurable and can work without human assistance.

2.6.7.1 Routing protocol for non-associated stations

By default, any STA turned on will try to associate to the WSN if it has not been associated yet. If an STA has lost the connectivity to the WSN or has been turned on for the first time, it will perform the same steps. These are the following ones:

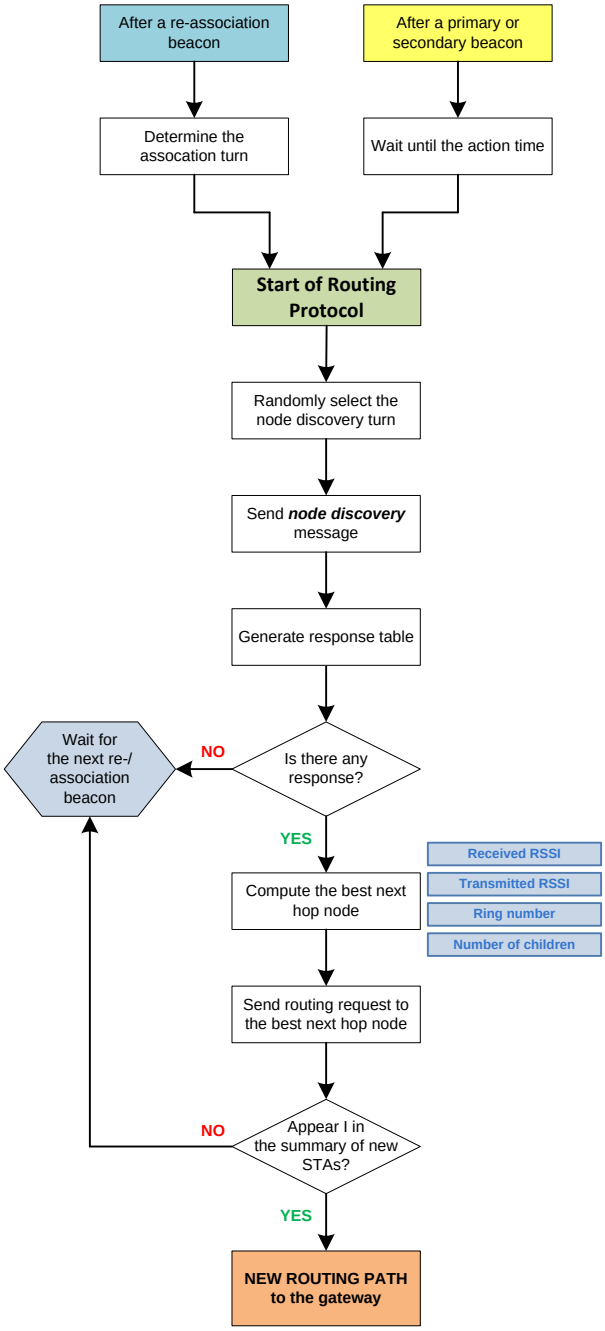


Figure 2.55: ENTOMATIC routing protocol diagram

1. Listen to the channel until receiving a **re-association beacon** or **the last secondary beacon** from a GW. Cases where the STA is outside the transmission area of the GWs are not considered, as downlink transmissions must be guaranteed in the ENTOMATIC project[13].

2. After listening to the **re-association** or **the last secondary beacon**, the STA computes its association turn as detailed in section 2.6.4.2. Within its association turn, the STA randomly selects its own node discovery turn, as a protection method against collisions resulting from requests of other STAs.

[13] We are considering allowing relays to extend the GW coverage, but in fact, the STA that does not reach the GW directly but received the beacon the same way.

3. Once within the corresponding node discovery turn, the STA enters in the **AUX-Discovery** state and sends a discovery request at maximum power via broadcast. All nodes currently associated to the WSN that listen to this request and are able to act as parent answer via unicast at maximum power too. Then, the STA willing to associate will determine which of the possible candidates is the best one; i.e., the one which minimizes the value from the following formula:

$$S(RSSI, r, ch) = w_1 \cdot RSSI_t + w_2 \cdot RSSI_r + w_3 \cdot r + w_4 \cdot c$$

Where $RSSI_t$ is the RSSI received at the parent candidate, $RSSI_r$ is the RSSI received at the STA itself, r is the ring the candidate belongs to, and c is the number of children the candidate already has. The weights w_i are attached in every **re-association beacon** and can be tuned according to the environment requirements.

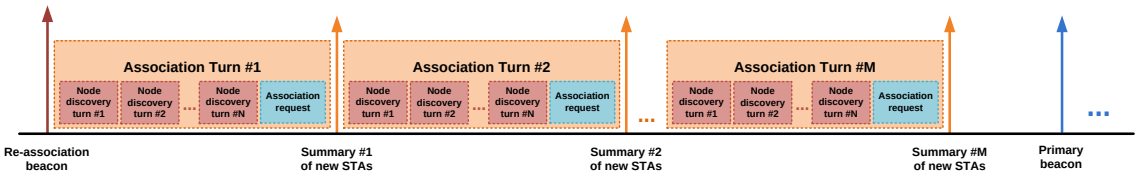


Figure 2.56: Channel scheduling after a re-association beacon

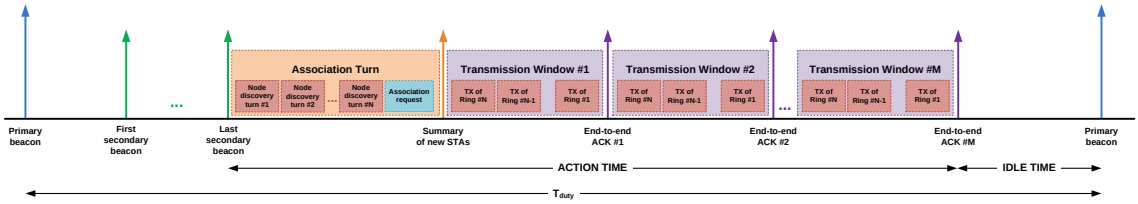


Figure 2.57: Channel scheduling after the last secondary beacon

From the excerpt of the ENTOMATIC code loaded in stations, we can realize that there are two available methods to choose the best next hop. When *ASSOC_MECH* flag is activated, nodes take into account four different metrics from the possible candidates to become its next hop in the path to the gateway: RSSI received, RSSI, transmitted, number of hops to the gateway, and number of children. Otherwise, nodes only take into consideration the received RSSI.

Table 2.39: Excerpt of the ENTOMATIC software code loaded in stations when performing the process to determine the best next hop in the routing path

```
uint8_t best_assoc_resp()
{
[...]

#if ASSOC_MECH
       comp = (-assoc_rx.rssi_rx[i]*weights.p_rssi_rx)
              + (-assoc rx.rssi tx[i]*weights.p rssi tx)
              + (assoc rx.ring[i]*weights.p ring)
              + (assoc_rx.children[i]*weights.p_children);
#else
       if((assoc_rx.ring[i]<min)||
       ((assoc_rx.ring[i]==min)&&(assoc_rx.rssi_rx[i]>rssi_max)))
       {
           min = assoc rx.ring[i];
           rssi_max = assoc_rx.rssi_rx[i];
           ind = i;
       }
#endif

[...]
}
```

As it can be seen in Figure 2.58 and Figure 2.59, the network topology of a typical layout of nodes can change greatly depending on the selected computing method. Thus, when limiting the

maximum number of children per station to 5, the weighted method provides a better distribution of nodes into rings, with the following advantages:

- Fewer transmitted packets, due to the aggregation
- Smaller power transmission modes, since nodes from an upper ring are closer

4. When the **AUX-Discovery** period is ended, all STAs enter in **AUX-Association** state. At this point, the STA willing to be associated, sends via unicast a specific request to its previously determined best next hop (as known as *parent*). This request will be forwarded until reaching the GW, which will send via broadcast a packet confirming the association and providing the new rime address of the STA. This way, both the STA and its new parent are sure that the path is established.

Routing protocol based on received RSSI, transmitted RSSI, number of hops to the gateway and number of children of the intermediate node (weighted method)

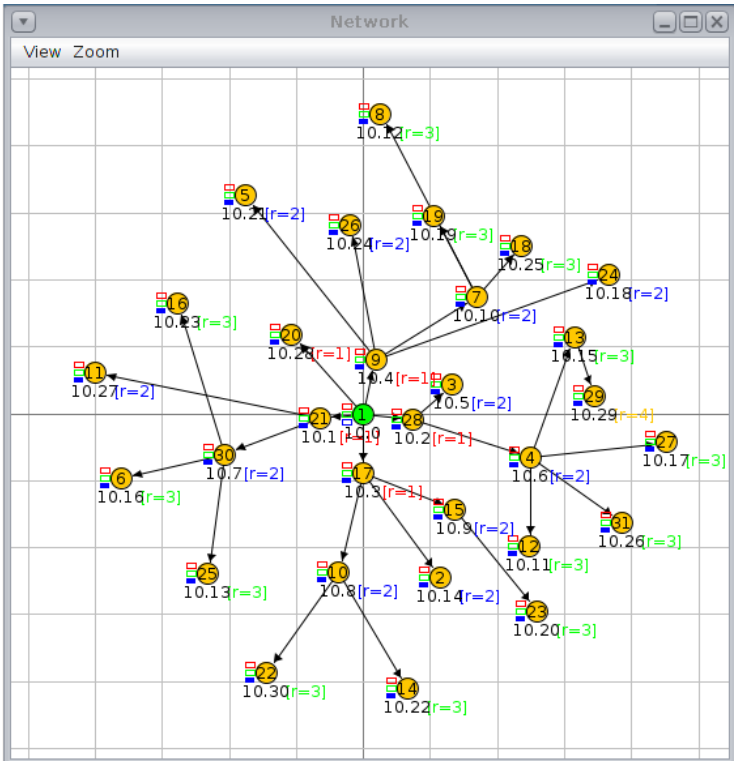


Figure 2.58: Topology of a 30-node network simulated in Cooja using a routing protocol that takes into consideration several parameters of the surrounding nodes

```
00:41.617       ID:1    +++++++++++++  ROUTING TABLE  +++++++++++++
00:41.626       ID:1    [10. 1]          | MAC:00:12:74:15:00:15:15:15  |  Next Hop: 10. 0  |  Ring: 1  |  #Children: 2
00:41.634       ID:1    [10. 2]          | MAC:00:12:74:1c:00:1c:1c:1c  |  Next Hop: 10. 0  |  Ring: 1  |  #Children: 2
00:41.642       ID:1    [10. 3]          | MAC:00:12:74:11:00:11:11:11  |  Next Hop: 10. 0  |  Ring: 1  |  #Children: 3
00:41.650       ID:1    [10. 4]          | MAC:00:12:74:09:00:09:09:09  |  Next Hop: 10. 0  |  Ring: 1  |  #Children: 4
00:41.658       ID:1    [10. 5]          | MAC:00:12:74:03:00:03:03:03  |  Next Hop: 10. 2  |  Ring: 2  |  #Children: 0
00:41.666       ID:1    [10. 6]          | MAC:00:12:74:04:00:04:04:04  |  Next Hop: 10. 2  |  Ring: 2  |  #Children: 4
00:41.674       ID:1    [10. 7]          | MAC:00:12:74:1e:00:1e:1e:1e  |  Next Hop: 10. 1  |  Ring: 2  |  #Children: 3
00:41.682       ID:1    [10. 8]          | MAC:00:12:74:0a:00:0a:0a:0a  |  Next Hop: 10. 3  |  Ring: 2  |  #Children: 2
00:41.690       ID:1    [10. 9]          | MAC:00:12:74:0f:00:0f:0f:0f  |  Next Hop: 10. 3  |  Ring: 2  |  #Children: 1
00:41.698       ID:1    [10.10]          | MAC:00:12:74:07:00:07:07:07  |  Next Hop: 10. 4  |  Ring: 2  |  #Children: 3
00:41.706       ID:1    [10.11]          | MAC:00:12:74:0c:00:0c:0c:0c  |  Next Hop: 10. 6  |  Ring: 3  |  #Children: 0
00:41.714       ID:1    [10.12]          | MAC:00:12:74:08:00:08:08:08  |  Next Hop: 10.10  |  Ring: 3  |  #Children: 0
00:41.722       ID:1    [10.13]          | MAC:00:12:74:19:00:19:19:19  |  Next Hop: 10. 7  |  Ring: 3  |  #Children: 0
00:41.730       ID:1    [10.14]          | MAC:00:12:74:02:00:02:02:02  |  Next Hop: 10. 3  |  Ring: 2  |  #Children: 0
00:41.738       ID:1    [10.15]          | MAC:00:12:74:0d:00:0d:0d:0d  |  Next Hop: 10. 6  |  Ring: 3  |  #Children: 1
00:41.746       ID:1    [10.16]          | MAC:00:12:74:06:00:06:06:06  |  Next Hop: 10. 7  |  Ring: 3  |  #Children: 0
00:41.754       ID:1    [10.17]          | MAC:00:12:74:1b:00:1b:1b:1b  |  Next Hop: 10. 6  |  Ring: 3  |  #Children: 0
00:41.763       ID:1    [10.18]          | MAC:00:12:74:18:00:18:18:18  |  Next Hop: 10. 4  |  Ring: 2  |  #Children: 0
00:41.771       ID:1    [10.19]          | MAC:00:12:74:13:00:13:13:13  |  Next Hop: 10.10  |  Ring: 3  |  #Children: 0
00:41.780       ID:1    [10.20]          | MAC:00:12:74:17:00:17:17:17  |  Next Hop: 10. 9  |  Ring: 3  |  #Children: 0
00:41.788       ID:1    [10.21]          | MAC:00:12:74:05:00:05:05:05  |  Next Hop: 10. 4  |  Ring: 2  |  #Children: 0
00:41.796       ID:1    [10.22]          | MAC:00:12:74:0e:00:0e:0e:0e  |  Next Hop: 10. 8  |  Ring: 3  |  #Children: 0
00:41.804       ID:1    [10.23]          | MAC:00:12:74:10:00:10:10:10  |  Next Hop: 10. 7  |  Ring: 3  |  #Children: 0
00:41.812       ID:1    [10.24]          | MAC:00:12:74:1a:00:1a:1a:1a  |  Next Hop: 10. 4  |  Ring: 2  |  #Children: 0
00:41.821       ID:1    [10.25]          | MAC:00:12:74:12:00:12:12:12  |  Next Hop: 10.10  |  Ring: 3  |  #Children: 0
00:41.829       ID:1    [10.26]          | MAC:00:12:74:1f:00:1f:1f:1f  |  Next Hop: 10. 6  |  Ring: 3  |  #Children: 0
00:41.837       ID:1    [10.27]          | MAC:00:12:74:0b:00:0b:0b:0b  |  Next Hop: 10. 1  |  Ring: 2  |  #Children: 0
00:41.845       ID:1    [10.28]          | MAC:00:12:74:14:00:14:14:14  |  Next Hop: 10. 0  |  Ring: 1  |  #Children: 0
00:41.854       ID:1    [10.29]          | MAC:00:12:74:1d:00:1d:1d:1d  |  Next Hop: 10.15  |  Ring: 4  |  #Children: 0
00:41.862       ID:1    [10.30]          | MAC:00:12:74:16:00:16:16:16  |  Next Hop: 10. 8  |  Ring: 3  |  #Children: 0
00:41.865       ID:1    ++++++++++++++++++++++++++++++++++++++++++++++
```

Figure 2.59: Routing table of the network simulated in Figure 2.58

Routing protocol only based on received RSSI (simplified method)

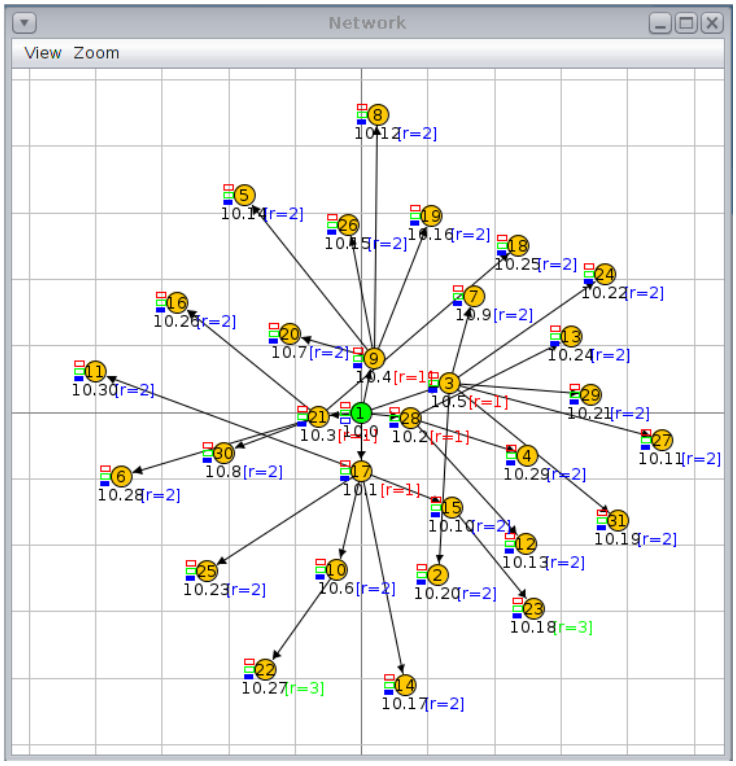


Figure 2.60: Topology of a 30-node network simulated in Cooja using a routing protocol that only takes into consideration the received RSSI of the surrounding nodes

```
00:41.001       ID:1    ++++++++++++++  ROUTING TABLE  ++++++++++++++
00:41.009       ID:1    [10. 1]           | MAC:00:12:74:11:00:11:11:11  |  Next Hop: 10.0   |  Ring: 1  |  #Children: 5
00:41.017       ID:1    [10. 2]           | MAC:00:12:74:1c:00:1c:1c:1c  |  Next Hop: 10.0   |  Ring: 1  |  #Children: 3
00:41.025       ID:1    [10. 3]           | MAC:00:12:74:15:00:15:15:15  |  Next Hop: 10.0   |  Ring: 1  |  #Children: 4
00:41.033       ID:1    [10. 4]           | MAC:00:12:74:09:00:09:09:09  |  Next Hop: 10.0   |  Ring: 1  |  #Children: 5
00:41.041       ID:1    [10. 5]           | MAC:00:12:74:03:00:03:03:03  |  Next Hop: 10.0   |  Ring: 1  |  #Children: 6
00:41.049       ID:1    [10. 6]           | MAC:00:12:74:0a:00:0a:0a:0a  |  Next Hop: 10.1   |  Ring: 2  |  #Children: 1
00:41.057       ID:1    [10. 7]           | MAC:00:12:74:14:00:14:14:14  |  Next Hop: 10.4   |  Ring: 2  |  #Children: 0
00:41.065       ID:1    [10. 8]           | MAC:00:12:74:1e:00:1e:1e:1e  |  Next Hop: 10.3   |  Ring: 2  |  #Children: 0
00:41.073       ID:1    [10. 9]           | MAC:00:12:74:07:00:07:07:07  |  Next Hop: 10.5   |  Ring: 2  |  #Children: 0
00:41.081       ID:1    [10.10]           | MAC:00:12:74:0f:00:0f:0f:0f  |  Next Hop: 10.1   |  Ring: 2  |  #Children: 1
00:41.089       ID:1    [10.11]           | MAC:00:12:74:1b:00:1b:1b:1b  |  Next Hop: 10.5   |  Ring: 2  |  #Children: 0
00:41.097       ID:1    [10.12]           | MAC:00:12:74:08:00:08:08:08  |  Next Hop: 10.4   |  Ring: 2  |  #Children: 0
00:41.105       ID:1    [10.13]           | MAC:00:12:74:0c:00:0c:0c:0c  |  Next Hop: 10.2   |  Ring: 2  |  #Children: 0
00:41.113       ID:1    [10.14]           | MAC:00:12:74:05:00:05:05:05  |  Next Hop: 10.4   |  Ring: 2  |  #Children: 0
00:41.121       ID:1    [10.15]           | MAC:00:12:74:1a:00:1a:1a:1a  |  Next Hop: 10.4   |  Ring: 2  |  #Children: 0
00:41.129       ID:1    [10.16]           | MAC:00:12:74:13:00:13:13:13  |  Next Hop: 10.4   |  Ring: 2  |  #Children: 0
00:41.137       ID:1    [10.17]           | MAC:00:12:74:0e:00:0e:0e:0e  |  Next Hop: 10.1   |  Ring: 2  |  #Children: 0
00:41.145       ID:1    [10.18]           | MAC:00:12:74:17:00:17:17:17  |  Next Hop: 10.10  |  Ring: 3  |  #Children: 0
00:41.154       ID:1    [10.19]           | MAC:00:12:74:1f:00:1f:1f:1f  |  Next Hop: 10.5   |  Ring: 2  |  #Children: 0
00:41.162       ID:1    [10.20]           | MAC:00:12:74:02:00:02:02:02  |  Next Hop: 10.5   |  Ring: 2  |  #Children: 0
00:41.170       ID:1    [10.21]           | MAC:00:12:74:1d:00:1d:1d:1d  |  Next Hop: 10.5   |  Ring: 2  |  #Children: 0
00:41.178       ID:1    [10.22]           | MAC:00:12:74:18:00:18:18:18  |  Next Hop: 10.5   |  Ring: 2  |  #Children: 0
00:41.187       ID:1    [10.23]           | MAC:00:12:74:19:00:19:19:19  |  Next Hop: 10.1   |  Ring: 2  |  #Children: 0
00:41.195       ID:1    [10.24]           | MAC:00:12:74:0d:00:0d:0d:0d  |  Next Hop: 10.2   |  Ring: 2  |  #Children: 0
00:41.203       ID:1    [10.25]           | MAC:00:12:74:12:00:12:12:12  |  Next Hop: 10.3   |  Ring: 2  |  #Children: 0
00:41.211       ID:1    [10.26]           | MAC:00:12:74:10:00:10:10:10  |  Next Hop: 10.3   |  Ring: 2  |  #Children: 0
00:41.219       ID:1    [10.27]           | MAC:00:12:74:16:00:16:16:16  |  Next Hop: 10.6   |  Ring: 3  |  #Children: 0
00:41.227       ID:1    [10.28]           | MAC:00:12:74:06:00:06:06:06  |  Next Hop: 10.3   |  Ring: 2  |  #Children: 0
00:41.235       ID:1    [10.29]           | MAC:00:12:74:04:00:04:04:04  |  Next Hop: 10.2   |  Ring: 2  |  #Children: 0
00:41.243       ID:1    [10.30]           | MAC:00:12:74:0b:00:0b:0b:0b  |  Next Hop: 10.1   |  Ring: 2  |  #Children: 0
00:41.246       ID:1    ++++++++++++++++++++++++++++++++++++++++++++++++
```

Figure 2.61: Routing table of the network simulated in Figure 2.60

2.6.7.2 Routing protocol for already associated stations

After listening to the **last secondary beacon**, all associated STAs enter in the **AUX-Discovery** state and remain awake in order to answer possible discovery requests sent by nodes willing to associate to the WSN. At this point, it is worth noting that associated STAs which have previously reached the maximum number of children will not answer to subsequent petitions.

If a request is received by an already associated STA, it will provide the applicant node with a route through an unicast reply at the current rime address of the applicant STA (one rime address is provided by default), indicating the received RSSI in the node discovery message, its own ring and its own current number of children.

When the **AUX-Discovery** period is over, all STAs enter in the **AUX-Association** state and remain awake in order to listen to possible association requests (generated if they are considered to be the best potential parent of an applicant STA). Putting in sleep mode the STAs that have not listened to any discovery request in the **AUX-Discovery** state has been discarded, as it is possible that a parent STA does not listen to this request but one of his descent does. Then, all STAs should remain awake in order to provide a full path to applicant nodes.

Finally, the notification of all STAs incorporated to the network in the last association turn is done by the gateway, which transmits a broadcast message summarizing all this information. The message must be listened to by all STAs, so that applicant STAs can know if they have been finally admitted in the network, and parents know who exactly their children are from now on.

2.7 TRANSPORT LAYER

As we recall from general network layers concept, the major tasks of a Transport Layer are:

1. To guarantee the reliable transmission of network packets through end-to-end retransmissions or other strategies.
2. To reduce or avoid the network congestion due to too much traffic flowing in the routers or other relay points.

TCP is the transport layer used in Internet. However, traditional TCP/IP implementations require far too many resources both in terms of code size and memory usage to be useful in small systems. Code size of a few hundred kilobytes and RAM requirements of several hundreds of kilobytes have made it impossible to fit the full TCP/IP stack into systems with a few tens of kilobytes of RAM and room for less than 100 kilobytes of code [45].

The transport protocol runs over the network layer. It enables end-to-end message transmission, where messages may be fragmented into several segments at the transmitter and reassembled at the receiver. This protocol provides the following functions: orderly transmission, flow and congestion control, loss recovery, and possibly QoS guarantees such as timing and fairness [46].

In WSNs several new factors, such as the convergent nature of upstream traffic and limited wireless bandwidth, can result in congestion. Congestion impacts normal data exchange and may lead to packet loss. In addition, wireless channel introduces packet loss due to bit-error rate, which not only affects reliability, but also wastes energy. As a result, the two major problems that WSN transport protocols need to cope with are congestion and packet loss.

Congestion

Congestion in WSNs has a direct impact on energy efficiency and application QoS. There are mainly two causes for congestion in WSNs. The first is due to the packet-arrival rate exceeding the packet-service rate. This is more likely to occur at sensor nodes close to the sink, as they usually carry more combined upstream traffic. The second cause is link-level performance aspects such as contention, interference, and bit-error rate. This type of congestion occurs on the link.

Typically, there are three mechanisms that can deal with this problem:

- Congestion detection: A common mechanism would be to use queue length, packet service time, or the ratio of packet service time over packet interarrival time at the intermediate nodes. For WSNs using CSMA-like Medium Access Control (MAC) protocols, channel loading can be measured and used as an indication of congestion.
- Congestion notification: After detecting congestion, transport protocols need to propagate congestion information from the congested node to the upstream sensor nodes or the source nodes that contribute to congestion.
- Rate adjustment: Upon receiving a congestion indication, a sensor node can adjust its transmission rate.

Packet loss

In wireless environments, both congestion and bit error can cause packet loss, which deteriorates end-to-end reliability and QoS, and furthermore lowers energy efficiency. Other factors that result in packet loss include node failure, wrong or outdated routing information, and energy depletion.

Loss recovery systems in WSN are based on two main mechanisms:

- Loss detection and notification: A common mechanism is to include a sequence number in each packet header. The continuity of sequence numbers can be used to detect packet loss. Loss detection and notification can be either end-to-end or hop-by-hop. In the end-to-end approach, such as in TCP protocol, the end-points (destination or source) are responsible for loss detection and notification. In the hop-by-hop method, intermediate nodes detect and notify packet loss.
- Retransmission-based loss recovery: Retransmission of lost or damaged packets can be also either end-to-end or hop-by-hop. In the end-to-end approach, the source performs retransmission. In hop-by-hop retransmission, an intermediate node that intercepts loss notification searches its local buffer. If it finds a copy of the lost packet, it retransmits the packet. Otherwise it relays loss information upstream to other intermediate nodes.

2.7.1 TRANSPORT LAYERS FOR WIRELESS SENSOR NETWORKS

Several transport protocols have been designed for WSNs, some of which have addressed congestion or reliability only, while others have examined both. Literature categorizes them into three types: congestion control protocols, protocols for reliability, and protocols considering both congestion control and reliability.

- **Protocols for congestion control**

Several congestion control protocols have been proposed for upstream convergent traffic in WSNs. They differ in congestion detection, congestion notification, or rate-adjustment mechanisms.
Examples: STCP, Fusion, CODA, CCF, PCCP, ARC, Siphon, Trickle.

- **Protocols for reliability**
  Some transport protocols examine upstream reliability; others only investigate downstream reliability.
  Examples upstream: STCP, ESRT, RMST, RBC, DTC.
  Examples downstream: GARUDA, PSFQ, DTC.

- **Protocols for congestion control and reliability**
  STCP is a generic end-to-end upstream transport protocol. It provides both congestion control and reliability, allocating most responsibility at the sink. Intermediate nodes detect congestion based on queue length and notify the sink by setting a bit in the packet headers.

2.7.2 OVERVIEW OF THE ENTOMATIC TRANSPORT LAYER

Transport layer is defined in the current system as the procedural means of transferring variable-length uplink data sequences from an STA to the gateway and acknowledging their correct reception. The congestion problem has not been considered in the ENTOMATIC network due to the low number of stations planned to be allocated in the coverage area of the network. However, if some congestion could appear in a node dealing with too many children, it would be easily controlled by tuning a suitable number of children that can maintain each node of the network.

In turn, reliable end-to-end communications from the STAs to the GW, where retransmissions are only executed when needed and by the minimum number of involved STAs, are detected by the gateway after the end of each transmission window and notified to all stations by means of a broadcast-type end-to-end ack message. The retransmission-based loss recovery is performed hop-by-hop, using the same path which was used to send the original data packet.

2.7.2.1 End-to-end ACK

Uplink data is received by the GW in the last round of the *staggered wakeup pattern* from nodes in ring 1. Once compared the data sources with the expected uplink traffic, the GW emits a broadcast message called *end-to-end ACK* (e2e ACK) with the list of acknowledged STAs. Figure 2.62 shows the e2e ACK operation at the end of the transmission windows. Apart from being simple, quick and simultaneously listened by all network elements, end-to-end ACKs allow STAs to evaluate the state of their path to the GW and act consequently in subsequent transmission windows.

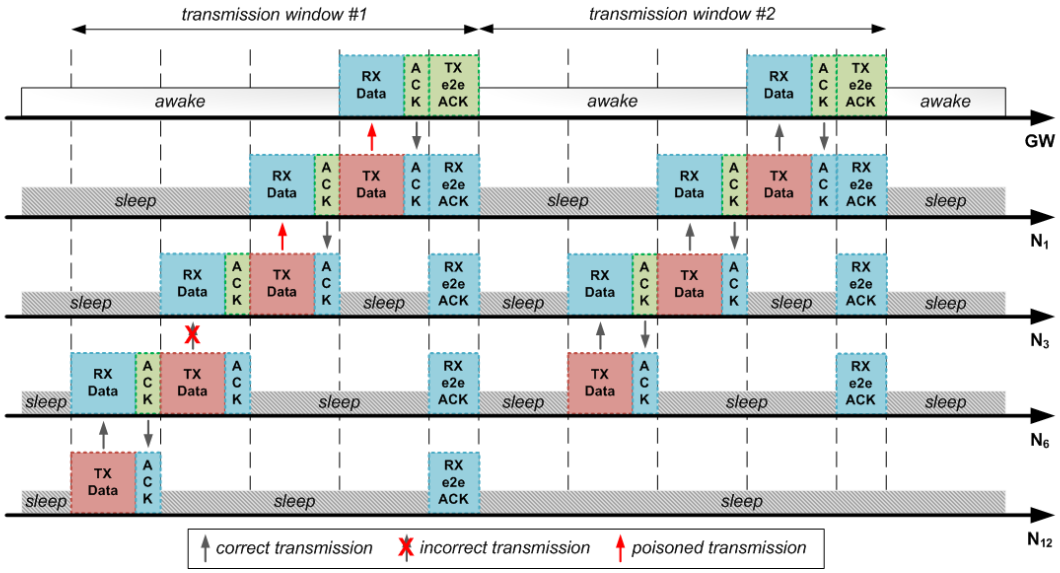


Figure 2.62: Data transmission phase in a multi-hop network running the ENTOMATIC system. Note the communication problems in the first transmission window between nodes N_6 and N_3.

2.7.2.2 Poisoning mechanism

The poisoning mechanism is intended to identify which specific nodes experience communication problems in their path to the GW, so that they can take measures to ensure the success of subsequent data transmissions. Nodes having problems with their children transmit packets to lower rings with the poison flag activated. An STA is considered *poisoned* if one of the following conditions occurs:

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

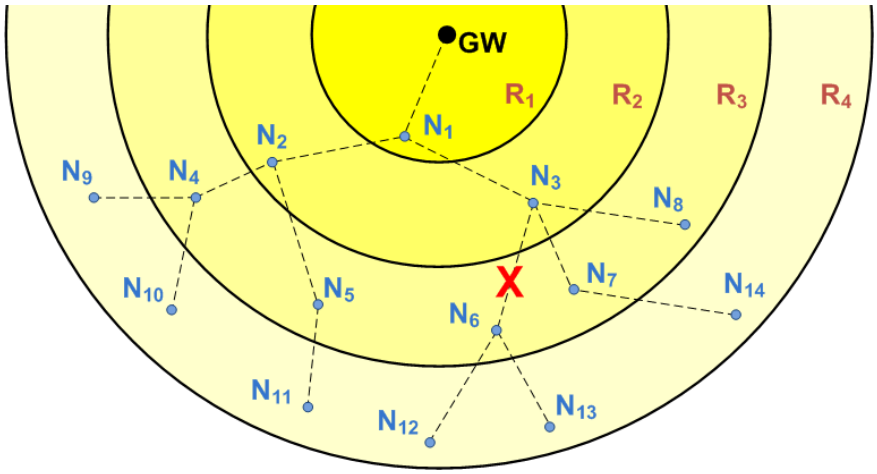


Figure 2.63: Network topology of the multi-hop wireless network from Figure 2.62.
It consists of a gateway (GW) and 30 stations (N_1 - N_{30}) deployed in 4 rings (R_1 - R_4)

In Figure 2.63, node N_3 activates its poison flag after not receiving data from its child N_6. In its way to the GW, a data packet from N_3 poisons its next hop: N_1. Therefore, nodes N_6, N_3, and N_1 form a *poisoned path*, as shown in Figure 2.65.

2.7.2.3 Transmission windows

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

This decision takes into account if the STA has been previously poisoned by one of its children as well as several other conditions according to the decision flowchart from Figure 2.64 .Whenever an STA decides to go to sleep, it will remain in this state until the next primary beacon.

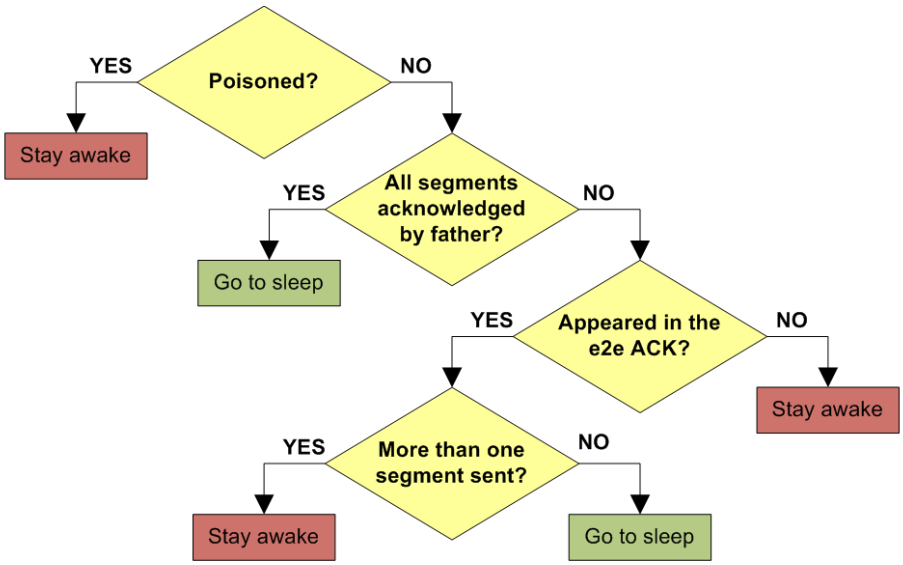


Figure 2.64: STA's decision flowchart to stay awake or to go sleep before the start of a new transmission window

2.7.2.4 Distributed caching

Due to the nature of multi-hop networks, lost packets cause expensive retransmissions along every hop of the path between the sender and the receiver [47]. To alleviate this problem, a distributed caching system is used, so that parents acknowledge the correct reception of packets from children and cache their data until it is properly received in the GW.

In combination with the above presented mechanisms of the transport layer (e2e ACKs, poisoning mechanism, and additional transmission windows), the distributed caching reduces the number of segment transmissions and end-to-end retransmissions in networks with communication problems.

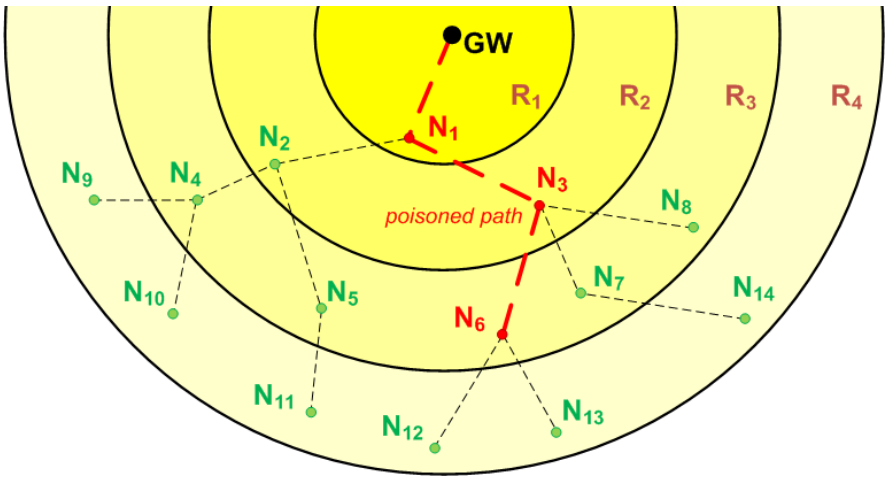


Figure 2.65: State of the network from Figure 2.63 after the corresponding e2e ACK.

Note the *poisoned path* passing through nodes N_6, N_3, and N_1. Together with the GW, these nodes (colored in red) stay awake during the first transmission window. The rest of nodes (colored in green) go to sleep as they are not involved in the new transmission process.

As it can be seen in Figure 2.65, nodes N_{12} and N_{13} can go to sleep after the first transmission window, because their data packets have been acknowledged by node N_6, which will cache them in memory together with its own data to be sent in the next transmission window.

2.8 APPLICATION LAYER

2.8.1 MAIN OPERATION

The ENTOMATIC protocol stack conceives end devices as elements controlled by the GW by means of beacons. This centralized approach allows STAs to remain asleep the majority of the time and their single concern is to be awake enough in advance for listening to the next beacon. Network synchronization is thus easily achieved, and the GW can ask for specific requests and distribute/deliver configuration changes in just one hop.

When a beacon signals the beginning of an uplink data transmission phase, channel access is split into as many slots as network rings. Nodes are only active during their own slot (for transmitting data) and the one belonging to their children[14] (for receiving data).

The first slot is allocated to the furthest ring from the GW and the rest are scheduled consecutively. To reduce the total number of transmissions and avoid using high power transmission levels, a multi-hop approach is used, where data received by STAs is aggregated to that generated by them, and sent to the corresponding parent at the minimum power level which ensures reliable communication. This process is repeated as many times as rings have the network.

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

The network association is also started by a beacon and remains stable until a change in the topology is detected or the mechanism is reset by the GW. However, the agreed transmission power between adjacent nodes in the association stage is constantly monitored and adjusted in order to reduce the energy consumption.

2.8.2 FRAME STRUCTURE

2.8.2.1 Frame size

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

Type of communication	IEEE 802.15.4 MTU	Size of RDC layer header	Size of network layer header	Available space for the application layer
Broadcast	127 bytes	2 bytes	16 bytes	**109 bytes**
Unicast	127 bytes	2 bytes	32 bytes	**93 bytes**

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

As it will be described in 2.8.3 with the description of the different application packets, data and statistics packets consist of 10 and 20 bytes of information, respectively. Having into account the available space for the application layer, in case an intermediate STA aggregates data from other stations, it could include up to 8 different data payloads and up to 4 different statistics payloads.

2.8.2.2 Packet headers

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

Table 2.40: Packet headers encoding

Identifier	Packet	Header bit index															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	Void beacon	Packet id															
		0	0	0	0												
1	Data pkt	Packet id				RSSI control		Agg?	Total segments			Current segment					
		0	0	0	1	x	x	x	x	x	x	x	x	x			
2	Data poissoned	Packet id				RSSI control		Agg?	Total segments			Current segment					
		0	0	1	0	x	x	x	x	x	x	x	x	x			
3	ACK	Packet id				RSSI control											
		0	0	1	1	x	x										
4	Data beacon	Packet id				Kill Flag											
		0	1	0	0	x											
5	e2e ACK	Packet id				Routing Aux											
		0	1	0	1	x	x	x									
6	Node discovery	Packet id				Mode											
		0	1	1	0	Req =0 Resp = 1											
7	Association	Packet id				Mode											
		0	1	1	1	Req =0 Resp = 1 Ret = 2											
8	Re-Association beacon	Packet id				Mode											
		1	0	0	0	Req =0 Resp = 1 Ret = 2											
9	Statistics packet	Packet id				RSSI control		Agg?	Total segments			Current segment					
		1	0	0	1	x	x	x	x	0	0	x	0	0			
10	Statistics poissoned	Packet id				RSSI control		Agg?	Total segments			Current segment					
		1	0	1	0	x	x	x	x	0	0	x	0	0			
11	Statistics beacon	Packet id				Kill Flag											
		1	0	1	1	x											
12	Radar	Packet id				Mode											
		1	1	0	0	Req =0 Resp = 1											
13	Connection test	Packet id				Mode											
		1	1	0	1	Req =0 Resp = 1											

The fields encoded in the headers are detailed below:

- **Packet id:** packet identifier.
- **RSSI control:** flag for acting according the power mechanism.
- **Aggregation:** flag indicating if more than one segment will be sent in a complete transmission.
- **Total segment:** total number of segments in a complete transmission.
- **Current segment:** number of the segment that has been sent.
- **Kill flag:** flag for identifying if any of the STAs in the network has been killed (or de-associated).
- **Discovery, Association, Re-association, Radar and Connection Test mode:**
  - **Req:** node discovery / association request.
  - **Resp:** node discovery / association response.
  - **Ret:** (only for Assoc. and Re-Assoc) node request retransmitted to a lower ring.

2.8.2.3 Management frames

There are four main types of management packets: discovery, association, radar, and connection test. While the first one is related to the node discovery phase, the second is used for requesting and confirming association. The radar packet is useful to gather information of the network and its stations from a single point (mainly the GW). The connection test is the first message transmitted by an STA once it has been switched on, and its main purpose is to check the availability of GWs in its own range coverage.

Table 2.41: Management frames

Type	Sender	Transmission	Management frames byte index																	
			0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Discovery request	STA	Broadcast																				
Discovery response	GW / STA	Unicast	*rssi_rx*			*ring*		*ch*														
Association request	STA	Unicast	*mac*									*node_id*										
Association response	GW	Broadcast	*timestamp*				*length*		*new*		*mac_children*						*rime_parent*		*next_hop*		*ring*	*ch*
Association retransmission	STA	Unicast	*mac_child*									*node_id_ch*		*rime*								
Radar request	GW	Broadcast	*seq*																			
Radar response	STA	Unicast	*seq*				*mac*									*rssi_recv*		*bat*				
Connection test request	STA	Broadcast																				
Connection test response	GW	Unicast	*t_response*																			

- **Discovery:**
  - **Request:** the node discovery request is just composed with the 2 byte header. No additional fields are included in the packet.
  - **Response:** the node discovery response includes the RSSI received at the potential parent (*rssi_rx*), and its ring (*ring*) and number of children (*ch*).
- **Association:**
  - **Request:** the association request packet includes the MAC address (*mac*) and node ID (*node_id*) of the STA willing to be associated.
  - **Retransmission:** association request retransmissions include the information contained in the request forwarded and an additional field with the RIME address of the intermediate hop (*rime*).
  - **Response:** the association response is sent via broadcast by the GW and summarizes the information regarding the new STAs associated to the network. That is, the fields included are the timestamp for synchronization (*timestamp*), the packet length (*length*) for identifying the number of association response packets to be sent, the number of new associated STAs (*new*), and other fields containing the information of the new routing connections (shown darker in Table 2.41). Specifically, such fields are the MAC address of the new STAs associated (*mac_ch*), the RIME address of the assigned parent (*rime_parent*), the allocated next hop (*next_hop*), the ring of the associated STA (*ring*), and the number of children (*ch*).
- **Radar:**
  - **Request:** the radar request only includes a sequence number (*seq*).
  - **Response:** the radar response copies the received sequence number (*seq*) and also includes the own MAC of the STA which answers to the request (*mac*), the RSSI received in the request message (*rssi_recv*) and its current battery level (*bat*).
- **Connection test:**
  - **Request:** the connection test request is just composed with the 2 byte header. No additional fields are included in the packet.
  - **Response:** the connection test response includes the time until the next beacon is transmitted (*t_response*),

2.8.2.4 Beacons

There are four different beacon types that can be emitted by the GW in a broadcast transmission: re-association beacon, void beacon, data beacon, and statistics beacon. As seen in blue color in ***2.8.2.2.***

Packet headers, each one of them has its own packet header. As for the content of the beacon itself, they can be grouped into re-association beacons and regular beacons (this group contains the aforementioned void beacon, data beacon, and statistics beacon).

Table 2.42: Beacon frames

Type	Beacons byte index																													
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
Re-association beacon	timestamp				t_euthanasia				t_reassoc_wake				association_info																	
													max_rssi		num ranges	gap_r1	t_range	w_1	w_2	w_3	w_4	max ch	num slots	t_slot						
Regular beacon	timestamp				t_duty				t_action				t_unit	t_tx	t_assoc	R	W	-	association_info											
																			max_rssi		num ranges	gap_r1	t_range	w_1	w_2	w_3	w_4	max ch	num slots	t_slot

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

- **Regular beacon (void, data, or statistic beacon):**
  - **timestamp:** Field for synchronization purposes
  - **t_duty:** Time to next duty cycle
  - **t_action:** Time to the next primary beacon
  - **t_unit:** Time unit in which the mentioned variables are represented (usually given in seconds)
  - **t_tx:** Time an STA can be in TX state
  - **t_assoc:** Time slot for association
  - **R:** Number of rings in the network
  - **W:** Number of transmission windows
  - **-:** Reserved space (currently not used)
  - **association_info:** Block of fields containing information corresponding to the association phase of the system. It consists of the following parameters:
    - **max_rssi:** Maximum RSSI allowed in the re-association process
    - **num ranges:** Number of complete association ranges
    - **gap_r1:** RSSI gap of the first association range
    - **t_range:** Time length of an association range
    - **w_i:** Weights to compute the best response to the association proces
    - **w_1:** Weight of the received RSSI

- **w_2:** Weight of the RSSI received at the candidate node
- **w_3:** Weight of the ring of the candidate
- **w_4:** Weight of the number of children of the candidate
- **max ch:** Maximum number of children per STA set in the network
- **num slots:** Number of node discovery slots
- **t_slot:** Time of each node discovery slot

2.8.3 APPLICATION PACKETS

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

The data and statistics gathering process is performed by means of 4 types of packets: data and statistics, containing information about sensor measurements and performance metrics, and link and end-to-end acknowledgements (i.e., ACK and e2eACK, respectively). Table 2.43 shows the frame structure of the mentioned application packets.

Table 2.43: Data, statistics, and ACKs frame structure

Type	Application packets byte index																			
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Data packet	*rime*		*seq*	*events*	*flies*	*temp*		*hum*	*light*	*bat*										
Statistics packet	*rime*		*seq*	*-*	*pkts_sent*		*pkts_acked*		*acks_sent*		*rtt_link*		*rtt_e2e*		*time_states*				*power_levels*	
ACK packet	*segment*	*seq*																		
e2eACK beacon	*ack_encoded*																			

2.8.3.1 Data packets

Data packets are intended to compile the information collected by traps on the olive field and contain the following fields:

- **Rime**: Network address of the STA (although the network address of the sender is always included in the IEEE 802.15.4 header, it is indispensable to attach this information in every data packet to avoid misunderstandings about the data source when performing data aggregation).
- **Seq:** Data packet sequence number.
- **Events:** Number of events detected (in our case, the total number of insects detected by the fly sensor).
- **Flies:** Similarly to *events,* in this case this field informs about the specific number of *Batrocera Oleae* flies detected.
- **Temp:** Temperature measured by the sensor in °C with two decimals accuracy (for instance, 25.45 °C).
- **Hum:** Humidity measured by the sensor in % without decimals (for instance, 58 %).
- **Light:** Luminance measured by the sensor in % without decimals (for instance, 72 %).
- **Battery:** Remaining battery capacity of the Zolertia Re-Mote in % without decimals (for instance, 94%).

2.8.3.2 Statistics packets

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

- **Rime:** Network address of the STA (although the network address of the sender is always included in the IEEE 802.15.4 header, it is indispensable to attach this information in every data packet to avoid misunderstandings about the data source when performing data aggregation).
- **Seq:** Statistics packet sequence number.
- **-:** Reserved space (currently not used).
- **Pkts_sent:** Number of data packets sent by the STA.
- **Pkts_acked:** Number of data packets properly acknowledged by the STA's parent.
- **Acks_sent:** Number of link ACK's packets sent to the STA's children.
- **Rtt_link:** Average Round Trip Time (RTT) at link level expressed in ms.
- **Rtt_e2e:** Average Round Trip Time (RTT) at end-to-end level expressed in s.
- **Time_states:** Percentage of time spent by the STA in each of the possible transceiver states.
- **Power_lvls:** Maximum and minimum power level used by the STA during the period from the last statistics packets sent.

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

Table 2.44: Network performance metrics generated by the ENTOMATIC gateway

#	Variable	Metric	Type [15]	Update period
1	t_O	Observation time	Accumulated	Cycle
2	$n_{A,c}$	Number of associated STAs cycle c	Current	Cycle
3	D	Number of duty cycles	Accumulated	Cycle
4	PDR	Packet delivery ratio	Accumulated	Cycle
5	CSR	Cycle stability ratio	Accumulated	Cycle
6	N_A	Total number of associations	Accumulated	Cycle
7	N_K	Number of kills	Accumulated	Cycle
8	$\bar{N}_R$	Mean number of rings	Accumulated	Cycle
9	$\bar{d}_A$	Mean association delay	Accumulated	Cycle
10	$N_{A,c}$	Number of associations in cycle c	Current	Cycle
11	N_c	Number of rings in cycle c	Current	Cycle
12	$\bar{N}_{ACK,c}$	Mean number of ACKs sent per STA in cycle c	Current	Statistics
13	$\bar{t}_{k,d}$	Mean share of time in state k in cycle c	Current	Statistics
14	$\bar{\rho}_d$	Mean number of transmissions per packet	Accumulated	Statistics
15	RTT_{link}	RTT at link level	Current	Statistics
16	RTT_{e2e}	RTT at end-to-end level	Current	Statistics
17	$\bar{t}_{k,d}$	Percentage of time in each state	Accumulated	Statistics
18	P_{max} /P_{min}	Maximum and minimum employed power level	Current	Statistics

- **Observation time and number of duty cycles**
  Time and number of duty cycles that had taken place during the experiment. We consider that time starts when the GW button is pressed (first beacon sent). It is important to note that

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

before pressing the GW button, all the STAs in the network should be already turned on (in RX/listening state).

- **Associations and disassociations**
  - $n_{A,c}$: Number of associated STAs in the last cycle c. That is, the number of stations included in the routing table in the last cycle.
  - N_A: Total number of associations performed during the observation time.
  - $n_{K,c}$: Number of disassociated STAs in the last cycle c. That is, the number of stations removed from the routing table in the last cycle.
  - N_K: Total number of disassociations performed during the observation time.

- **PDR and CSR**
  - The Packet Delivery Ratio (PDR) metric measures the network efficiency in terms of data reception. It is the ratio of the number of expected payloads received and expected.

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

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

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

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

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

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

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

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

- **Mean number of transmissions per data packet**
  Mean number of transmissions per packet[16] for the stations associated in cycle d.

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

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

- **Number of ACKs sent per STA**

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

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

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

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

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

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

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

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

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

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

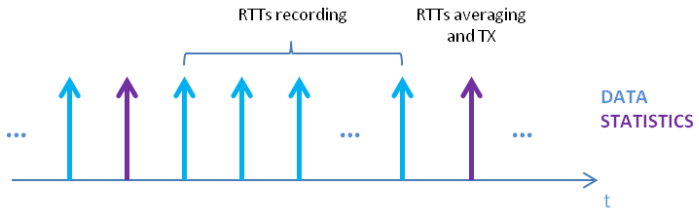


Figure 2.66: Mechanism of RTT averaging

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

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

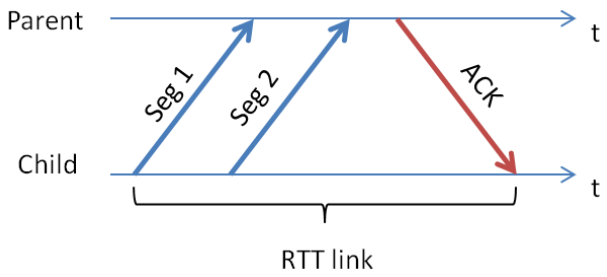


Figure 2.67: Computation of RTT link between a parent and a child when sending more than one data segment

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

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

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

2.8.3.3 Link ACK (ACK) packets

The link ACK packet informs about the proper reception of a data packet by a parent. It includes the following fields:

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

2.8.3.4 End-to-end ACK (e2e ACK) packets

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

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

2.9 INPUT AND OUTPUT

2.9.1 DATA ACQUISITION

Once an STA receives a beacon from the gateway asking for a new report of environmental information, the installed routines in the Zolertia Re-Mote ask the corresponding sensor for this data. Three different sensors working under their own communication protocol are available in every node of the ENTOMATIC network: a fly sensor, a temperature and humidity sensor, and a luminance sensor.

2.9.1.1 Fly sensor

The communication between the Zolertia Re-Mote and the fly sensor is performed through a serial connection defined as ***9600 8N1,*** with its main characteristics summarized in Table 2.45.

Table 2.45: Serial connection parameters between the Zolertia RE-Mote and the fly sensor

Serial connection parameters	Value
Baud rate	9600 bps
Data bits	8
Parity bits	No parity
Synchronization bits	1

The physical connection between the fly sensor and the Zolertia RE-mote (by using the UART #1) consists of 4 wires: one for sending data (TX), one for receiving data (RX), the power supply (3.3V), and the ground connector (GND). A diagram of this connection can be found in Figure 2.68.

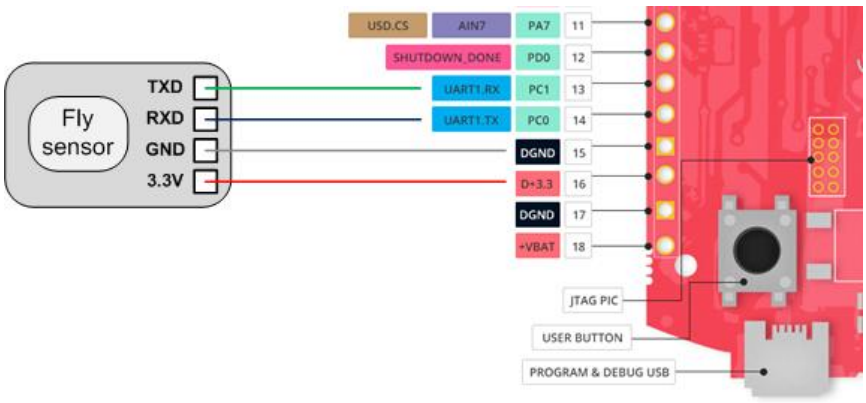


Figure 2.68: Physical connection between the fly sensor and the Zolertia RE-mote

As for the communication protocol between both devices, it will be based on the request-response system of the following lines:

A. Zolertia Re-Mote request to fly sensor

- Re-Mote request:

  `"Status?\r\n"`

- Fly sensor response:

  `"ID=123456,Counts=12,Batt=76,Lat=35.24356,Lon=24.01452,Power=ON\r\n"`

  Where:
  - ID: trap ID consisting of 6 digits
  - Counts: Detected insects
  - Batt: Battery Percentage Value (0-100)
  - Lat, Lon: GPS Coordinates
  - Power: Trap activity (ON or OFF)

B. Fly sensor to Zolertia Re-Mote

- Fly sensor request:

  `"Status?\r\n"`

- Zolertia Re-Mote response:

  `"Time=2016/08/01-12:34:57,alThr=7,Power=ON\r\n"`

  Where:
  - Time: Current Local Time [48]
  - alThr: Insects Alarm Threshold (If the number of detected insects is greater than alThr the trap sends immediately the Alarm to the Zolertia Re-Mote)
  - Power: ON or OFF. The Zolertia Re-Mote can enable or disable the insect detector.

- In addition, in case of alarm the fly sensor will send the following message:

  `"ID=123456,Counts=7,Batt=75\r\n"`

2.9.1.2 Sensirion SHT15 temperature and humidity sensor

The Sensirion SHT15 digital temperature and humidity sensor (Figure 2.69) is fully calibrated and offers high accuracy and excellent long term stability at low price. CMOSens digital technology integrates two

sensors as well as the necessary circuitry on a single chip. Note that the utilization format chosen for this sensor is based on a prefabricated board (*breakout*) that includes two resistors, a capacitor and 4 connection pins (VCC, DATA, SCK and GND), as shown in Figure 2.70.

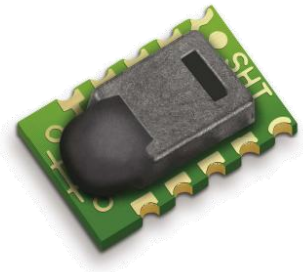


Figure 2.69: Sensirion SHT15 temperature and humidity sensor

Figure 2.70: Sensirion SHT15 breakout

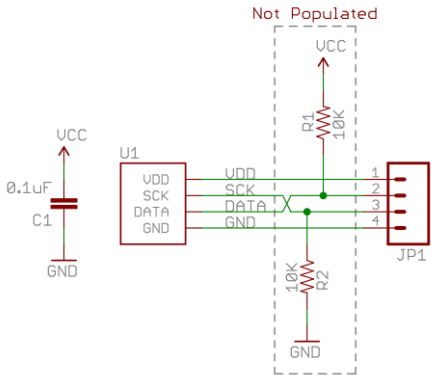


Figure 2.71: Electrical circuit diagram of the Sensirion SHT15 breakout

From the datasheet of the Sensirion SHT15 temperature and humidity sensor [48], we can extract its main features, which are summarized in the following lines (for more detailed information, see Figure 2.72 and Figure 2.73):

- 2 factory-calibrated sensors to measure relative humidity and temperature
- 2-wire digital interface
- Accurate calculation of dew point
- Measurement range: 0-100% RH
- Accuracy of humidity measurement: +/- 2% RH (10 ... 90% RH)
- Repeatability in humidity: +/- 0.1% RH
- Accuracy in temperature measurement: +/- 0.3 ° C @ 25 ° C
- Fast response time: <4 s.
- Low energy consumption (typ 30 µW)
- Low cost
- Low-cost high precision sensor
- Leading CMOSens technology for superior long-term stability

Relative Humidity

Parameter	Condition	min	typ	max	Units
Resolution [1]		0.4	0.05	0.05	%RH
		8	12	12	bit
Accuracy [2] SHT10	typical		±4.5		%RH
	maximal	see Figure 2			
Accuracy [2] SHT11	typical		±3.0		%RH
	maximal	see Figure 2			
Accuracy [2] SHT15	typical		±2.0		%RH
	maximal	see Figure 2			
Repeatability			±0.1		%RH
Hysteresis			±1		%RH
Non-linearity	linearized		<<1		%RH
Response time [3]	τ (63%)		8		s
Operating Range		0		100	%RH
Long term drift [4]	normal		< 0.5		%RH/yr

Temperature

Parameter	Condition	min	typ	max	Units
Resolution [1]		0.04	0.01	0.01	°C
		12	14	14	bit
Accuracy [2] SHT10	typical		±0.5		°C
	maximal	see Figure 3			
Accuracy [2] SHT11	typical		±0.4		°C
	maximal	see Figure 3			
Accuracy [2] SHT15	typical		±0.3		°C
	maximal	see Figure 3			
Repeatability			±0.1		°C
Operating Range		-40		123.8	°C
		-40		254.9	°F
Response Time [6]	τ (63%)	5		30	s
Long term drift			< 0.04		°C/yr

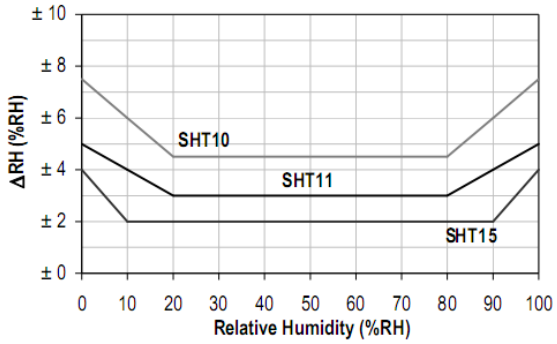


Figure 2: Maximal RH-tolerance at 25°C per sensor type.

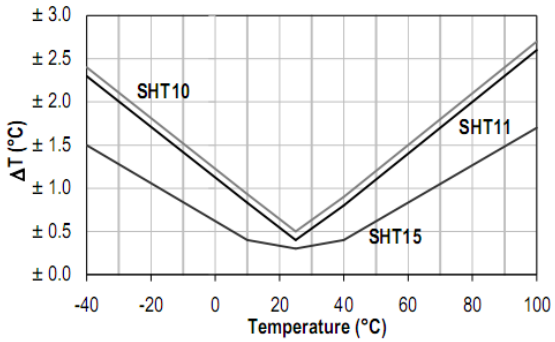


Figure 3: Maximal T-tolerance per sensor type.

Figure 2.72: Main operational features of the Sensirion SHT15

Electrical and General Items

Parameter	Condition	min	typ	max	Units
Source Voltage		2.4	3.3	5.5	V
Power Consumption [5]	sleep		2	5	µW
	measuring		3		mW
	average		90		µW
Communication	digital 2-wire interface, see Communication				
Storage	10 – 50°C (0 – 125°C peak), 20 – 60%RH				

Figure 2.73: Electrical parameters of the Sensirion SHT15

The Zolertia RE-Mote uses the Molex 5-pin WM4903-ND male header connector to connect digital sensors based on I2C and SPI protocols. The pins are 2.54 mm spaced and the connector has the following pin-out:

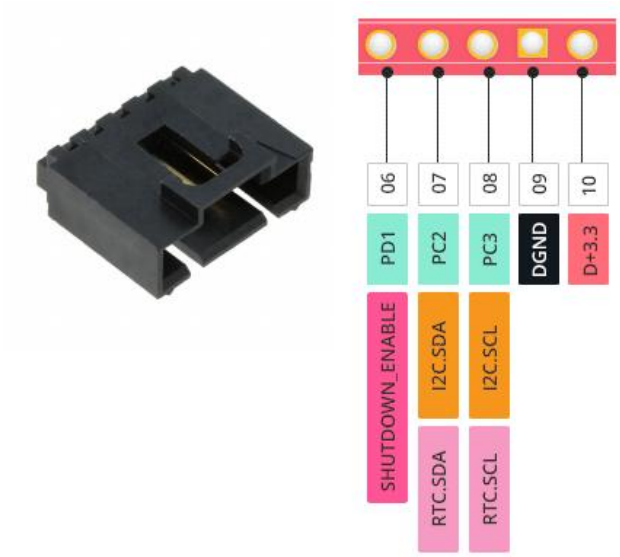


Figure 2.74: Zolertia Re-mote's digital connector pin-out

The Zolertia Re-Mote uses its digital I2C/SPI connector to communicate with the Sensirion SHT15 sensor. Although the sensor has a 2-wire interface similar to I2C, it is actually not compatible with I2C, so that new communication routines had to be programmed by means of the serial data (SDA) and serial clock (SCK) lines.

The purpose and configuration of the 4 different pins of the Sensirion SHT15 breakout is detailed in the following lines:

- **Power supply pins VCC and GND**
  The supply voltage of the SHT15 sensor must be within the range of 2.4 to 5 V, with the recommended voltage being 3.3 V. In addition, the VCC and GND pins must be decoupled using a 100 nF capacitor.

- **Serial clock input (SCK)**
  The SCK pin is used to synchronize the communication between the Zolertia Re-Mote's microcontroller and the SHT15 sensor. As its interface consists of a completely static logic, there is no minimum frequency.

- **Serial data (SDA)**
  The tri-state DATA pin is used to transfer data to and from the sensor. To send a command to the sensor, DATA is valid on the rising edge of the serial clock (SCK) and must remain stable for as long as the value of the SCK signal is high. After the lowering edge of the SCK signal, the DATA value must be changed.

2.9.1.3 DHT22 temperature and humidity sensor

As an alternative to the SHT15 temperature and humidity sensor, the DHT22 sensor (also named as AM2302) has also been integrated and can be used in the ENTOMATIC wireless modules. Unlike the SHT15, the DHT22 uses an analog connector of the Zolertia RE-Mote.

The DHT22 is a basic, low-cost temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and spits out a signal on the data pin. ts fairly simple to use, but requires careful timing to grab data. The only real downside of this sensor is that new data can only be got from it once every 2 seconds; however, this is not a drawback for the ENTOMATIC project, as time between two consecutives measures will be always much higher.

Figure 2.75: DHT22 temperature and humidity sensor

From the datasheet of the DHT22 temperature and humidity sensor [49], we can extract its main features, which are summarized in the following lines (for more detailed information, see Figure 2.76 and Figure 2.77):

- Supply Voltage: 3.3-5.5V
- Temperature Range:-40-80℃/ resolution0.1℃/ error <±0.5℃
- Humidity Range: 0-100%RH / resolution0.1%RH / error±2%RH
- Wiring map: VCC,GND,S
- Size: 38 x 20mm(1.50x0.79")

5.1 Relative humidity

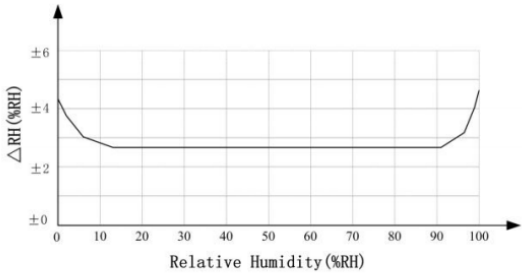
Table 2: AM2302 Relative humidity performance table

Parameter	Condition	min	typ	max	Unit
Resolution			0.1		%RH
Range		0		99.9	%RH
Accuracy [1]	25℃		±2		%RH
Repeatability			±0.3		%RH
Exchange		Completely interchangeable			
Response [2]	1/e(63%)		<5		S
Sluggish			<0.3		%RH
Drift [3]	Typical		<0.5		%RH/yr

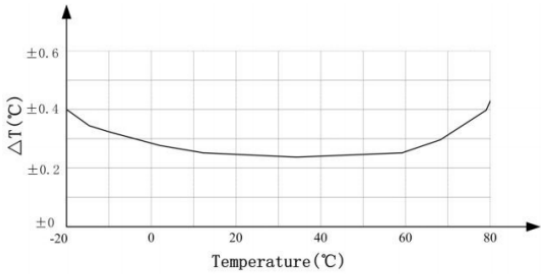
5.2 Temperature

Table 3: AM2302 Relative temperature performance

Parameter	Condition	min	typ	max	Unit
Resolution			0.1		℃
			16		bit
Accuracy			±0.5	±1	℃
Range		−40		80	℃
Repeat			±0.2		℃
Exchange		Completely interchangeable			
Response	1/e(63%)		<10		S
Drift			±0.3		℃/yr



Pic2: At25℃ The error of relative humidity



Pic3: The maximum temperature error

Figure 2.76: Main operational features of the DHT22 temperature and humidity sensor

Table 4: AM2302 DC Characteristics

Parameter	Condition	min	typ	max	Unit
Voltage		3.3	5	5.5	V
Power consumption [4]	Dormancy	10	15		μA
	Measuring		500		μA
	Average		300		μA
Low level output voltage	I_{OL}[5]	0		300	mV
High output voltage	Rp<25 kΩ	90%		100%	VDD
Low input voltage	Decline	0		30%	VDD
Input High Voltage	Rise	70%		100%	VDD
Rpu[6]	VDD = 5V VIN = VSS	30	45	60	kΩ
Output current	turn on		8		mA
	turn off	10	20		μA
Sampling period		2			S

Figure 2.77: Electrical parameters of the DHT22 temperature and humidity sensor

The RE-Mote has 2 x ADC available ports that can be used with the Molex 3-pin WM4901-ND male header connector, providing the normally used GND and VCC pins to connect analogue sensors. Depending on the sensor power operation requirement you can use the ADC1 (3.3V) or the ADC3 (5V). The pins are 2.54 mm spaced and the connector has the following pin-out:

Figure 2.78: Zolertia Re-mote's analog connector pin-out

In this case, the Zolertia Re-Mote uses one of the two ADC connectors to communicate with the Grove luminance sensor breakout. ADC communication libraries included in Contiki OS porting of the Zolertia Re-Mote have been used for this purpose.

2.9.1.4 Luminance sensor

The GL5528 light dependent resistor is a photo-resistor made of semi-conductor material, so that the conductance changes with luminance variation [50]. The resistance of this device varies according to the change of the strength of incident visible rays. Thus, in dark conditions its resistance raises up to 1MΩ, whereas with light conditions, its resistance is reduced to 10-20KΩ. They are generally used to take photosensitive resistor measurement, light control and photoelectric conversion (light changes into electricity).

Figure 2.79: GL5528 light dependent resistor

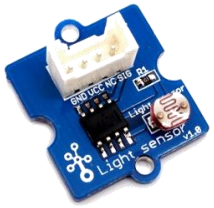


Figure 2.80: Grove luminance sensor *breakout*

From the datasheet of the GL5528 photoresistor [50], we can extract its main features, which are summarized in the following lines (for more detailed information, Table 2.46):

- Grove-light breakout
- Voltage: 3-5V
- Supply Current: 0.5-3mA
- Light resistance: 20KΩ
- Dark resistance: 1MΩ
- Response time: 20-30 secs
- Peak Wavelength: 540 nm
- Ambient temperature: -30 ~ 70 °C

Table 2.46: Main features of the light dependent resistor GL5528 series

CdS Photoresistors

Series (Size) (mm)	Style	Voltage Max	Power Consumption Max	Ambient Temp	Spectrum Value Max	Photo Resistance	Dark Resistance	γ	Response Time (ms)	
		(VDC)	(mW)	(℃)	(nm)	(10 Lux) (KΩ)	(MΩ)		Rise	Decay
Ø5	GL5516	150	90	-30 ~ +70	540	5 - 10	0.5	0.5	30	30
	GL5528	150	100	-30 ~ +70	540	10 - 20	1	0.6	20	30
	GL5537-1	150	100	-30 ~ +70	540	20 - 30	2	0.6	20	30
	GL5537-2	150	100	-30 ~ +70	540	30 - 50	3	0.7	20	30
	GL5539	150	100	-30 ~ +70	540	50 - 100	5	0.8	20	30
	GL5549	150	100	-30 ~ +70	540	100 - 200	10	0.9	20	30

The Grove luminance sensor breakout is a photo-resistor (light dependent resistor) to detect the intensity of light in the environment. The resistance of photo-resistor decreases when the intensity of light increases. A dual OpAmp chip LM358 on board produces voltage corresponding to intensity of light (i.e. based on resistance value). The output signal is analog value, the brighter the light, the larger the value. Two different breakouts have been ported to the ENTOMATIC platform: the Grove light sensor v1.1 and the v1.2.

The RE-Mote has 2 x ADC available ports that can be used with the Molex 3-pin WM4901-ND male header connector, providing the normally used GND and VCC pins to connect analogue sensors. Depending on the sensor power operation requirement you can use the ADC1 (3.3V) or the ADC3 (5V). The pins are 2.54 mm spaced and the connector has the following pin-out:

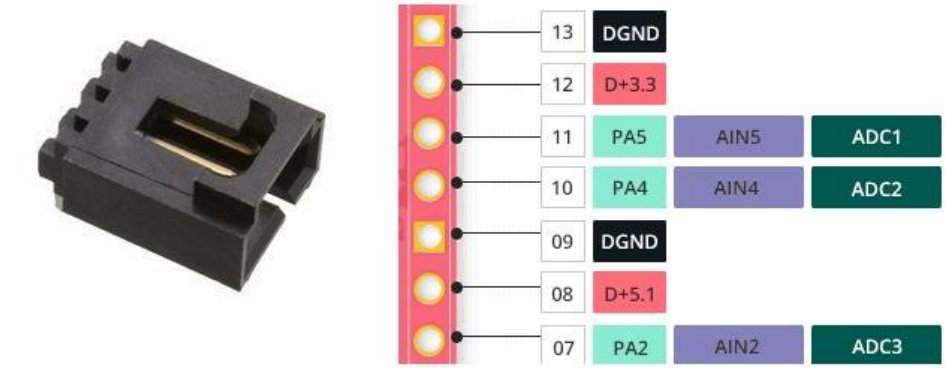


Figure 2.81: Zolertia Re-mote's analog connector pin-out

In this case, the Zolertia Re-Mote uses one of the two ADC connectors to communicate with the Grove luminance sensor breakout. ADC communication libraries included in Contiki OS porting of the Zolertia Re-Mote have been used for this purpose.

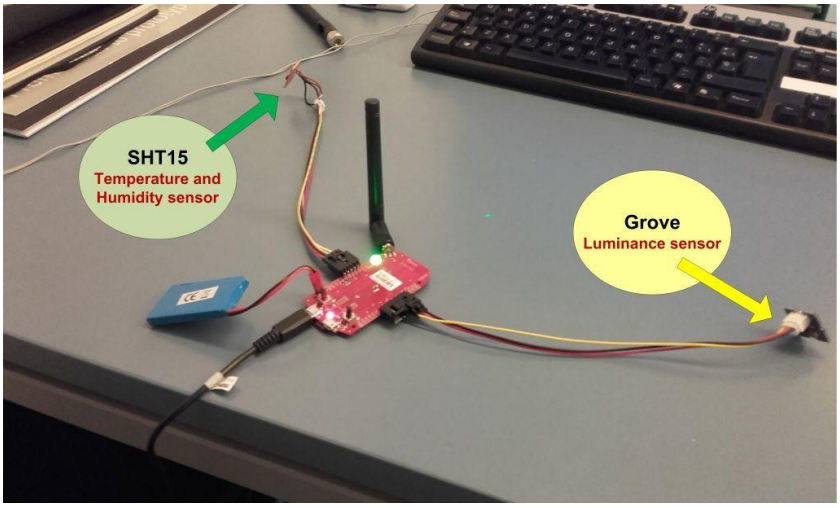


Figure 2.82 Zolertia Re-Mote with the SHT15 temperature/humidity [digital] and the Grove luminance sensor [analog] connected

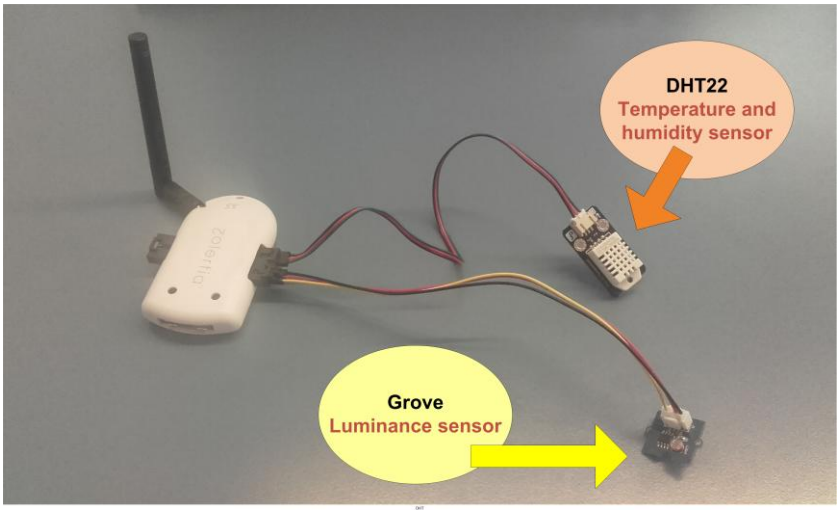


Figure 2.83: Zolertia Re-Mote with the DHT22 temperature/humidity [analog] and the Grove luminance sensor [analog] connected

2.9.2 LEDS

LEDs are a simple but important tool to communicate with users or to debug programs. Each one of the hardware platforms over which the ENTOMATIC system has been tested contains its own set of leds. In the following sections the color code used for each system state is detailed.

2.9.2.1 Crossbow TelosB

Apart from the leds used to indicate data transmission or reception, the Crossbow TelosB contains three different leds: Red, Green, and Blue.

Table 2.47: Meaning of leds' color code in Crossbow TelosB when running ENTOMATIC application

Color code	Light type (blinking or permanent)	Description
Blue	Permanent	Associated STA
Blue + Red	Permanent	Associated STA in 'poison' state
Red	Permanent	The STA has applied the self-disassociation mechanism
Green	After pressing the 'user' button, permanent	The STA is 'alive' (i.e., it contains enough battery to perform typical network operations)

Figure 2.84: A Crossbow TelosB node with all its leds (red, green, and blue) switched on

2.9.2.2 Zolertia Re-Mote

On the contrary, the Zolertia Re-Mote node contains one single RGB LED to allow more than 7 colour combinations.

Table 2.48: Leds meaning in Zolertia Re-Mote when running ENTOMATIC application

Colour code	Light type (blinking or permanent)	Description
Blue	Permanent	Associated STA
Pink	Permanent	Associated STA in 'poison' state
Red	Permanent	The STA has applied the self-disassociation mechanism
Green	After pressing the 'user' button, permanent	The STA is 'alive' (i.e., it contains enough battery to perform typical network operations)

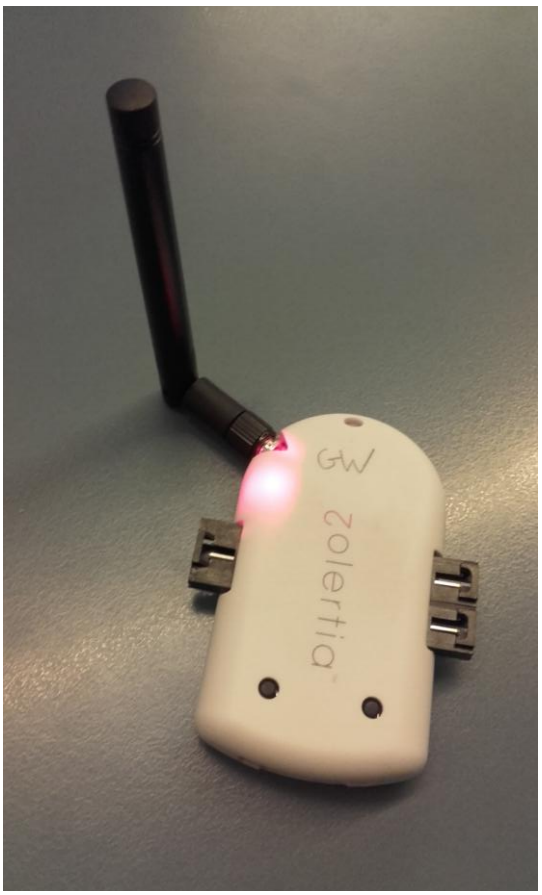


Figure 2.85: A Zolertia Re-Mote node with its red led switched on

2.9.3 MONITORING AND DEBUGGING TOOL

Apart from the color code implemented both in Crossbow TelosB and in Zolertia RE-Mote, it was highly necessary to design and develop a tool responsible of gathering and showing all the possible information from any station running the ENTOMATIC communication system. Prior to the development of this monitoring tool, some requirements were established:

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

The result was a Java-based monitoring tool able to gather, show and store all the log messages emitted by an STA directly connected to a PC via an USB cable. As can be seen in Figure 2.86, the tool informs about the COM port in which the STA is connected (for instance, `/dev/com5`) and show the ID of the STA once it has been properly recognized (in the example, ID is 6).

Different filters based on character strings can be applied on the gathered information in order to show only some relevant data or those debugging flags placed in the code run by STAs. Information can also be exported to a .txt file for post-processing purposes.

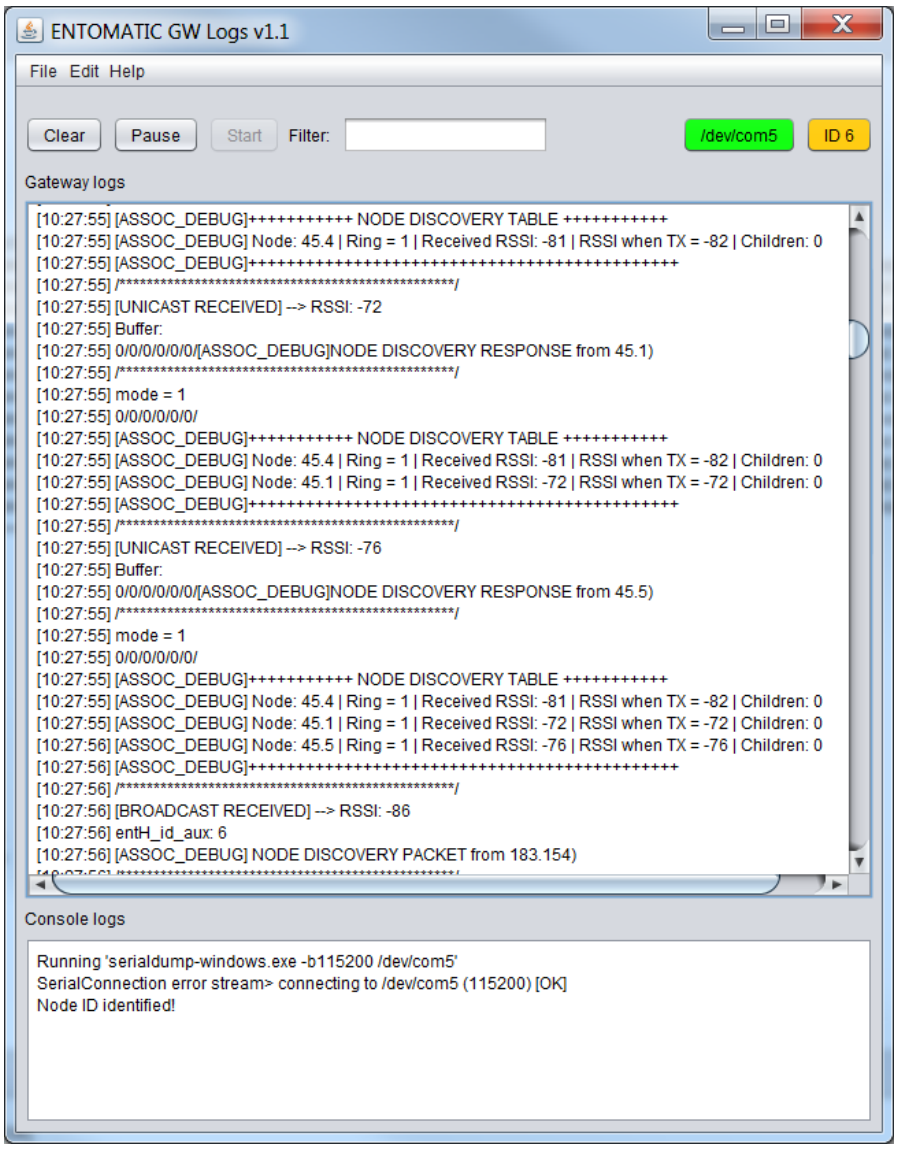


Figure 2.86: ENTOMATIC monitoring tool based on Java

2.10 BATTERY MANAGEMENT

2.10.1 BATTERY CHARACTERIZATION

The battery level is always valuable information in nodes forming a WSN. That is the reason why this value is transmitted along with every data packet.

2.10.1.1 Crossbow TelosB

In the case of the Crossbow TelosB, its energy source are two Energizer® rechargeable AA NiMH (Nickel Metal Hybride) batteries of 1.2 V each one, which can be charged up to 1000 cycles.

Figure 2.87: Energizer rechargeable AA NiMH battery

A typical discharge profile for a battery discharged at the 5-hour rate (the 0.2C rate) is shown in Figure 2.88. The initial drop from an open-circuit voltage of approximately 1.4 volts to the 1.2 volt plateau occurs rapidly [51]. Similar to lithium AA primary batteries, the nickel-metal hydride battery exhibits a sharp "knee" at the end of the discharge where the voltage drops quickly. As can be seen by the flatness of the plateau and the symmetry of the curve, the mid-point voltage (MPV - the voltage when 50 percent of the available capacity is discharged) provides a useful approximation to average voltage throughout the discharge.

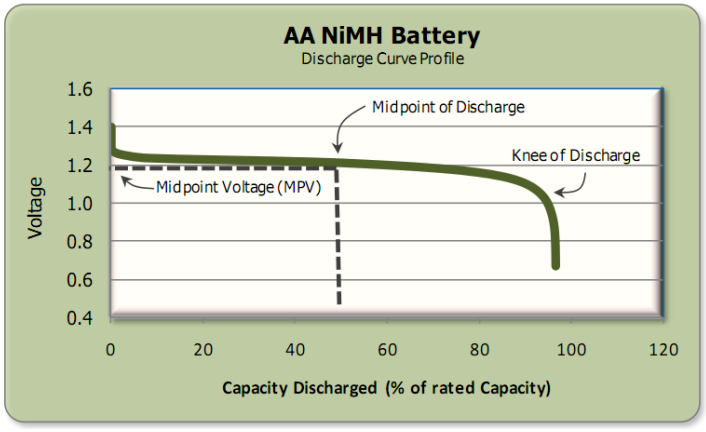


Figure 2.88: Typical Discharge Profile / NiMH Battery

The principal environmental influences on the location and shape of the voltage profile are the discharge temperature and discharge rate. As indicated in Figure 2.89, small variations from room temperature (± 10 ºC) do not appreciably affect the nickel-metal hydride battery voltage profile. However major excursions, especially lower temperatures, will reduce the mid-point voltage while maintaining the general shape of the voltage profile.

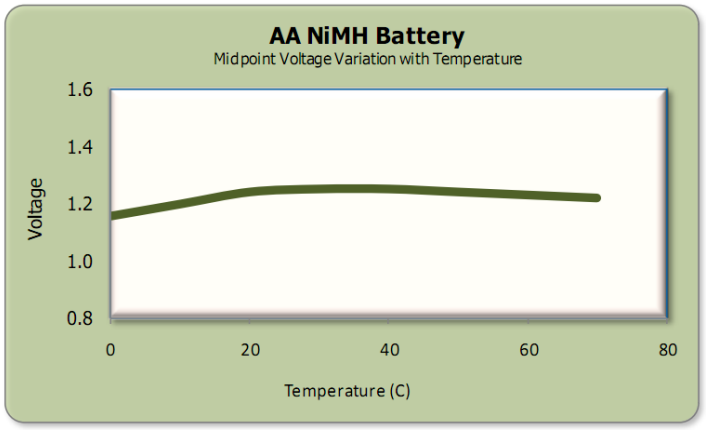


Figure 2.89: Midpoint Voltage Variation with Temperature

Contiki OS libraries include a mechanism to obtain a voltage reference of these batteries. The percentage of remaining battery has been deduced after several tests for characterizing its behaviour. When using 2 AA Energizer NiMH batteries with 2.300 mAh each one, the following results have been obtained after asking the microprocessor for its battery voltage. Due to the variable behaviour of discharging batteries, several thresholds have been defined to delimit the possible states of a battery.

Table 2.49: Defined battery states for AA batteries when powering Crossbow TelosB mote

Battery state	Voltage value (mV)	Voltage reference (mV)
Totally charged	2359	x > 2350
Operation mode	from 2071 to 2359	2050 < x < 2350
Running out of battery	2071	x < 2050

Thanks to these voltage references, each station is able to send in every data message the percentage of its remaining battery by considering the following reference table:

Table 2.50: Equivalence table between voltage value and battery level

Voltage value (mV)	Battery level (%)
2350	100 %
2320	90 %
2290	80 %
2260	70 %
2230	60 %
2200	50 %
2170	40 %
2140	30 %
2110	20 %
2080	10 %
2050	0 %

2.10.1.2 Zolertia Re-Mote

Zolertia Re-Mote's energy source is a 3.7 V LiPo (lithium polymer) rechargeable battery of 800 mAh capacity (P/N: 053048). It is directly connected to the device and can be recharged through an USB connector or a solar panel.

Figure 2.90: LiPo rechargeable battery

The durability of the LiPo rechargeable battery has been tested in a test performed in the UPF laboratory. Two Zolertia Re-Motes have established bidirectional communication, with one of them directly connected to a PC, and the other one located at 1 meter of distance and only powered through a LiPo rechargeable battery (this battery has been totally charged prior to the test).

The mote connected to the PC sends every 5 seconds at maximum power (14 dBm) a 'beacon' message consisted of 45 bytes (the minimum payload accepted by Contiki OS). Once this message is received by the other mote, it answers by also sending at maximum power (14 dBm) a 45-byte unicast message, which includes a packet sequence number and the current battery level in mV. (Both devices use the 50 kbps data rate mode). All this information is printed by a Java-based application run in the PC, where the other mote is connected.

As Figure 2.91 shows, after 16775 transmissions (that is, 23.2986 hours) the battery is exhausted. It is worth noting that most of time (more than 22 hours, or up to 95% of time) the battery level remains stable (around 3280 mV.) until it reaches a certain point when it dramatically falls down until 2494 mV. (detail of this drop is shown in Figure 2.92).

This analysis reveals that continuously reporting the exact value of the battery level has no sense, because it is always pretty much the same. However, it becomes crucial to detect the beginning of the battery level's drop in order to activate an alarm which would be used to activate the process of replacing the battery before it becomes completely exhausted. Hence, a threshold located in **3200 mV** is proposed to trigger this alarm.

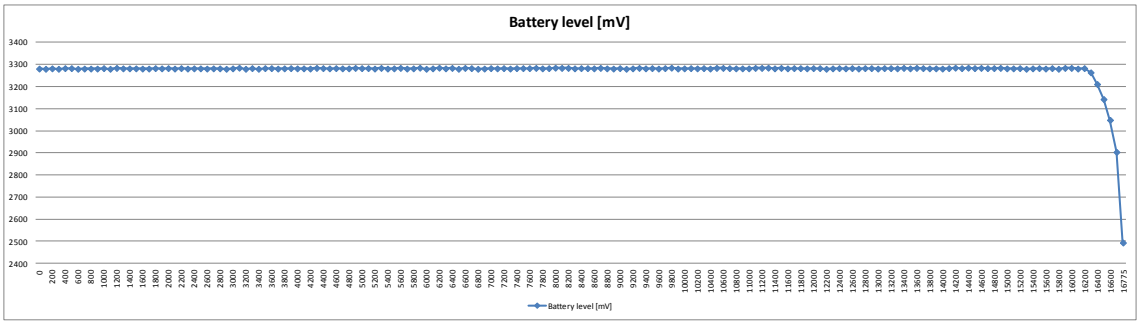


Figure 2.91: Battery level evolution of the Zolertia Re-Mote

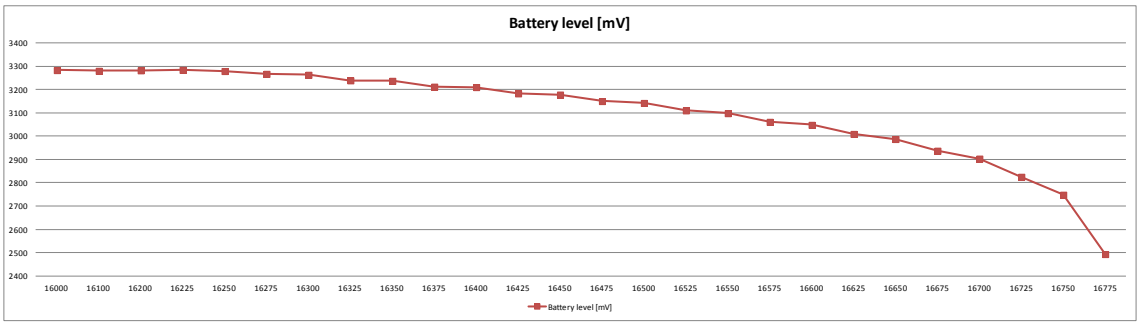


Figure 2.92: Detail of the battery level when the device is running out of energy

No further analysis on battery lifetime were performed at this stage, due to the low resolution of battery level in (mV) provided by the hardware. Only with reliable values of load current in the studied device, there had been possible to estimate the battery lifetime by computing the following equation[17]:

[17] http://www.digikey.es/en/resources/conversion-calculators/conversion-calculator-battery-life

$$Lifetime\ (h) = \frac{Q(mAh)}{\bar{C}(mA)} \cdot 0.7$$

where Q is the battery capacity in mAh and $\bar{C}$ is the average load current.

Nevertheless, subsection 2.11.2.2. Tests with Zolertia Re-Mote provides even more accurate values by using the own software energy module of the employed devices.

2.11 PERFORMANCE TESTS

2.11.1 SYSTEM VALIDATION IN SIMULATOR

The presented communication protocol was firstly validated in the Cooja simulator. Under ideal channel conditions (i.e., without noise and intereference from other transceivers not belonging to the network), several tests were carried out in order to evaluate the performance of the protocol. Basically, we were able to draw the following conclusions:

- **Packet Delivery Ratio**: the protocol ensures almost **100%** of packet delivery ratio in all the tested network topologies.
- **Energy consumption**: the STAs remain in the transceiver's sleep mode most of the time (> 99%) due to the implemented wake up patterns. Such mechanism allows to notably reduce the energy consumption.

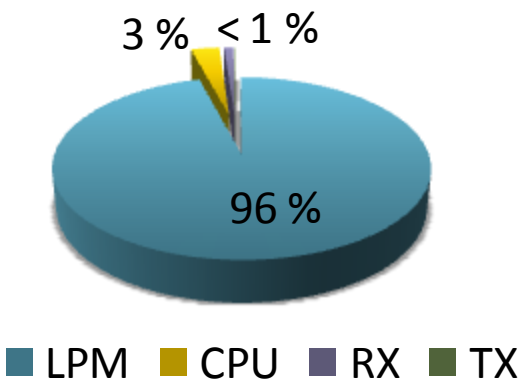


Figure 2.93: Energy consumption distribution among the four operational states: low power mode (LPM), processing (CPU), receiving (RX), and transmitting (TX)

- **Reliability and robustness**: redundancy beacons, multiple transmission windows (including the end-to-end ACK), and the backoff MAC mechanism allow to overcome unexpected events like packet collisions and parent STAs overloading. That is, if a packet is lost, the protocol is designed to recover that packet with the minimum additional energy consumption.

  In Figure 2.95, we see the results of intentionally overloading the GW, making it to receive too many packets in a very short period of time and, consequently, being unable to acknowledge all the packets from STAs. Nevertheless, the end-to-end ACK mechanism allows STAs to identify if their packets have been properly received at the GW.

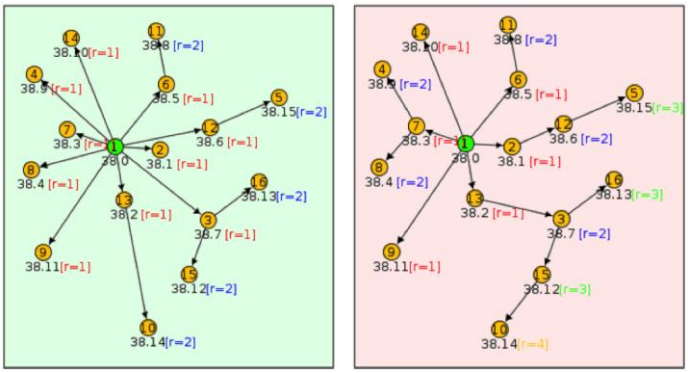


Figure 2.94: Two network topologies tested successfully in Cooja.

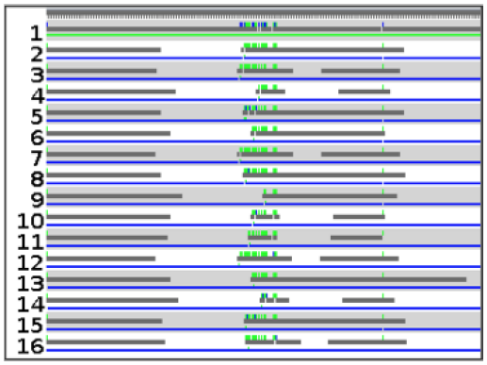


Figure 2.95: Unexpected events: GW overloading.

2.11.2 LABORATORY TESTBED

The design and development of the ENTOMATIC network has been carried out in the facilities of the Universitat Pompeu Fabra in Barcelona. The right wing of the 2nd floor of the Tanger building (see Figure 2.96), located in the Communication Campus [18], has been used as location for the first operational testbed of the network prior to tests in real environments.

For this purpose, this scenario has been reproduced using Google SketchUp v.16 [19], a 3D modeling tool. The resulting model maintains the highest fidelity with the real environment because it is generated from the building plans. Thus, all the items shown on the map (walls, light walls and even furniture) are contained in the reproduction made by SketchUp.

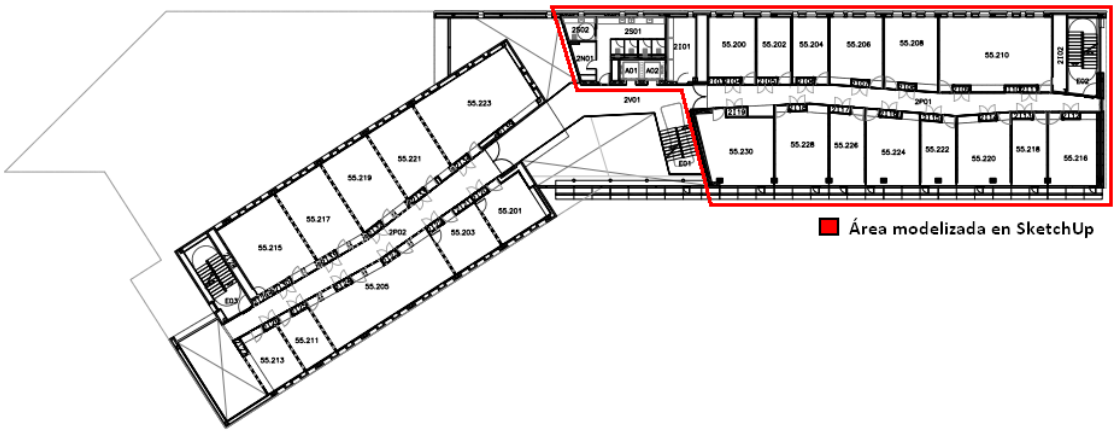


Figure 2.96: Area from the 2nd floor of the Tanger building

[18] UPF Communication Campus (DTIC) - https://www.upf.edu/campus/en/comunicacio/tanger.html
[19] Google SketchUp 16 - http://www.sketchup.com/

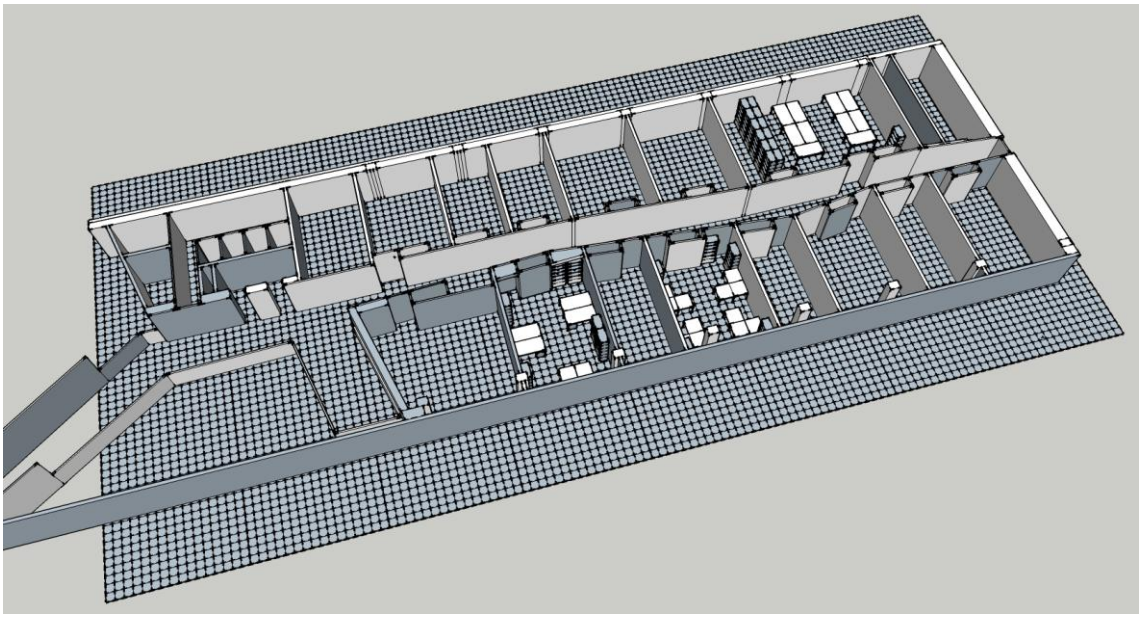


Figure 2.97: Perspective view of the right wing of the 2nd floor of the Tanger building modelled with Google SketchUp

The modelling of the environment where the network will operate is very useful to evaluate the maximum coverage allowed by antennas as well as to provide an overview of the network topology together with the possible links between different nodes. In this case, it also helps to characterize the propagation conditions of an indoor scenario, which by definition are always worse than those of the outdoors.

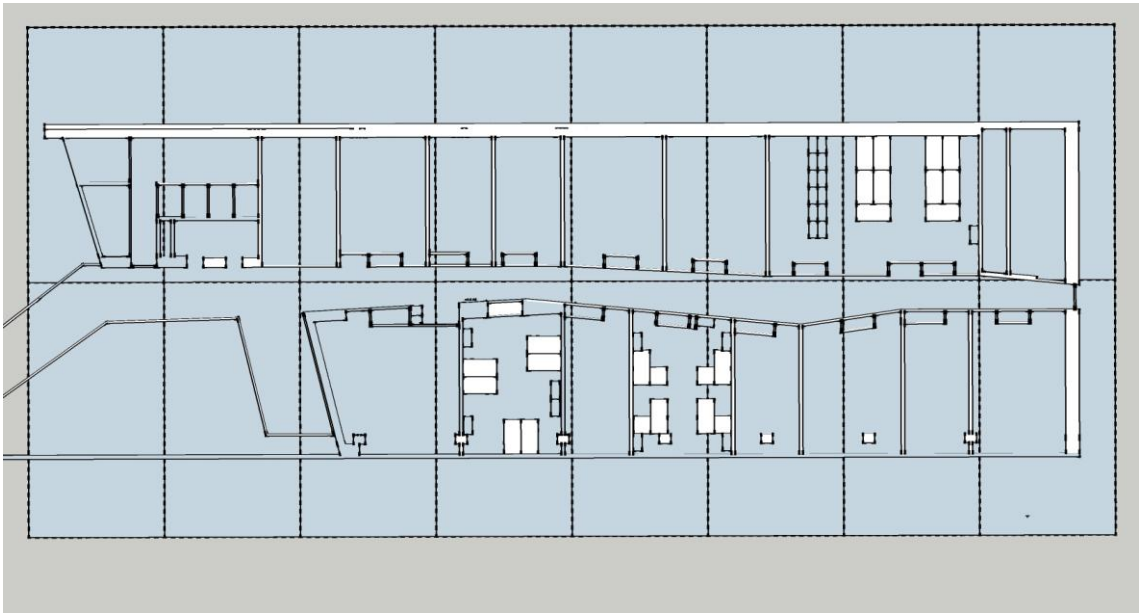


Figure 2.98: Top view of the right wing of the 2nd floor of the Tanger building modeled with Google SketchUp

2.11.2.1 Tests with Crossbow TelosB

First real tests of the proposed communication protocol were performed in the Crossbow TelosB platform, while the ordered Zolertia Re-mote platform was still being manufactured. Below we show the results gathered during the first experiment using the presented testbed scenario [20].

*Experiment **metadata**:*

- **Communication protocol version:** v1.0. (several new improvements are included in the current version of the protocol).
- **Location:** 2nd floor of the DTIC department at the UPF communication campus, Barcelona, Spain.
- **Date:** May Thursday 21st – Tuesday 30th
- **Number of STAs:** 30
- **Range:** ~ 30 meters
- **Observation time:** 225 hours (~ 9 days) ≡ 224 duty cycles
- **Platform:** Crossbow TelosB
  - Operational frequency = 2.4 GHz
  - Data rate = 250 kbps
  - Packet size ≅ 50 bytes
- **Operation:** Test (LEDs ON, UART and logs ON, etc.)
- **Notes:** The location experiment hinders the communication among nodes due to the high frequency of Wi-Fi (2.4 GHz) connected devices (smartphones, laptops, etc.). In addition, the channel frequency of the TelosB devices allows shorter ranges than the Re-Mote's ones (operating at 900 MHz).

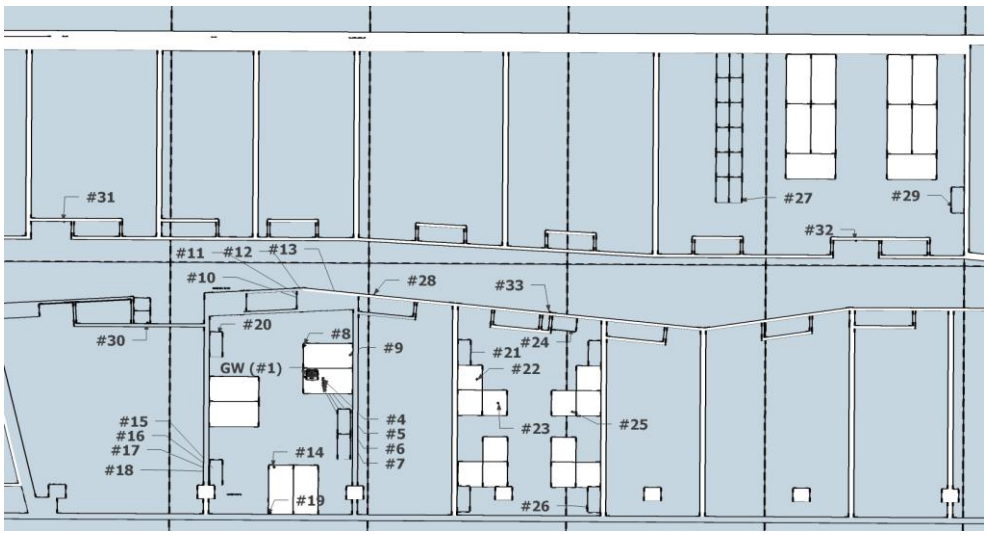


Figure 2.99: Detail of node distribution on the 2nd floor of the Tanger building

[20] **NOTE:** *it is important to note that these results were gathered through a **preliminar** version of the communication protocol. For the evaluation of the current protocol and **platform** see Section 2.10.1.2.*

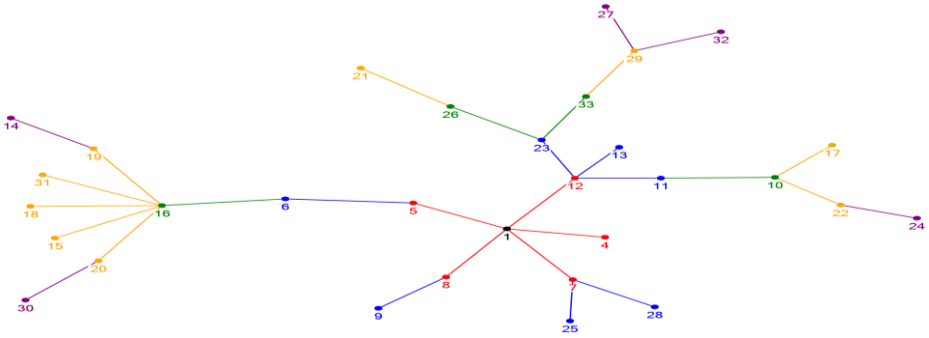


Figure 2.100: Network logical topology at duty cycle #7.

Associations and de-associations. It is noticeable that 29 out of 30 STAs are properly associated during the period compised between the re-association and the first data beacon, being the STA #29 (the furthest one from the gateway) the only one not yet associated. Thus, we can derive that the routing algorithm is working as expected even in real scenarios.

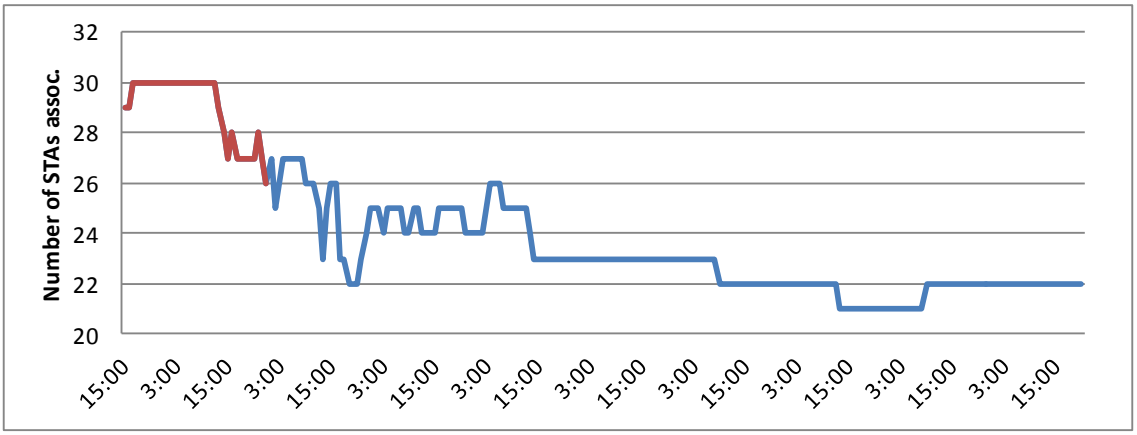


Figure 2.101: Temporal evolution of the number of STAs associated (observation time: 9 days).

We noticed that STAs #27, #29 and #33 were disassociated during more than 4 days. We analysed their register logs and discovered that STA #29 had depleted its battery (its green LED cannot be turned on). However, #27 and #33 could turn the light on, which let us to think that they may have not had enough energy to transmit.

In addition, some other de-associations occurred during the experiment. Most of them were caused by the three furthest STAs (#27, #29 and #32), as shown in the histogram from Figure 2.102, which in the majority of cases have some relation with the routing connection of STAs from #29 to #33 (see Figure 2.100). Hence, when any of these 3 considered STAs is disassociated, it produces the disassociation of corresponding children (i.e., if connection #29 - #33 fails, it causes the disassociation of children #27 and #32).

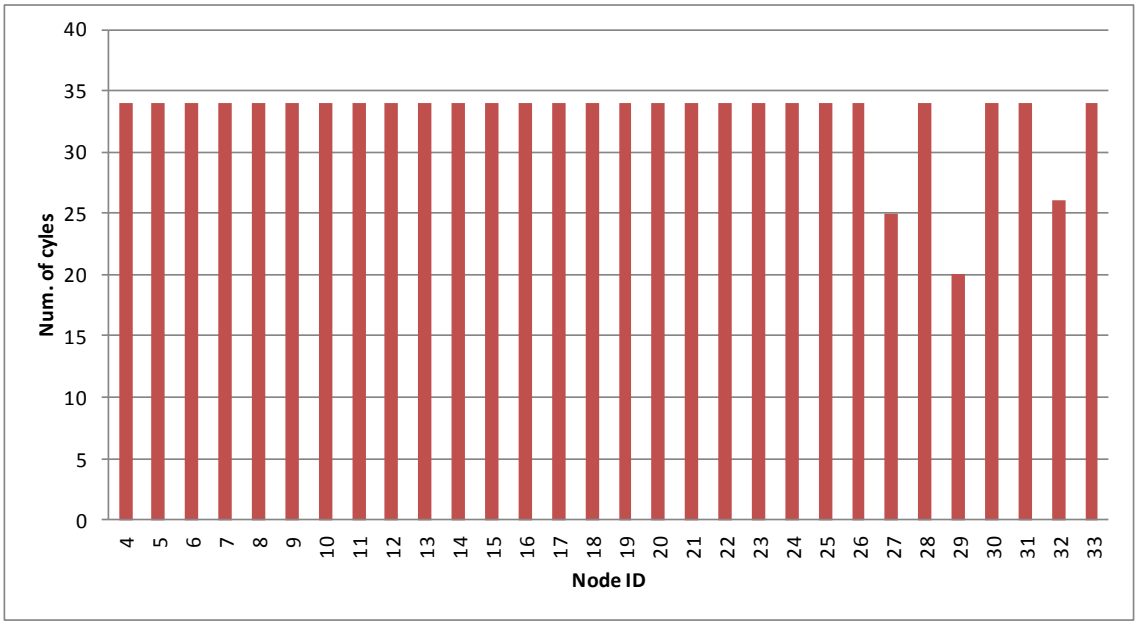


Figure 2.102: Histogram of number of cycles associated per STA.

PDR. As shown in Figure 2.103, the Packet Delivery Ratio (PDR) tends to be higher than 97%, which is a very good value and we can conclude that the network stabilizes itself. It is important to note that in scenarios with better channel conditions this value would be much closer to 100%. As the conditions of the experiment were not fully propitious due to the indoor issues with respect to obstacles and the 2.4 GHz ISM channel interferences, we think that the obtained PDR is reasonably appropriate and would be notably improved in outdoor scenarios.

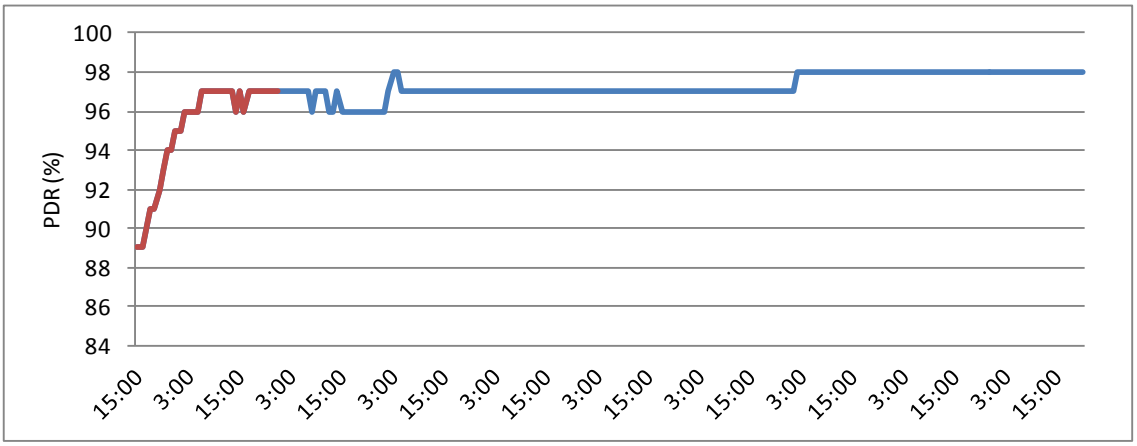


Figure 2.103: Temporal evolution of the PDR (observation time: 9 days).

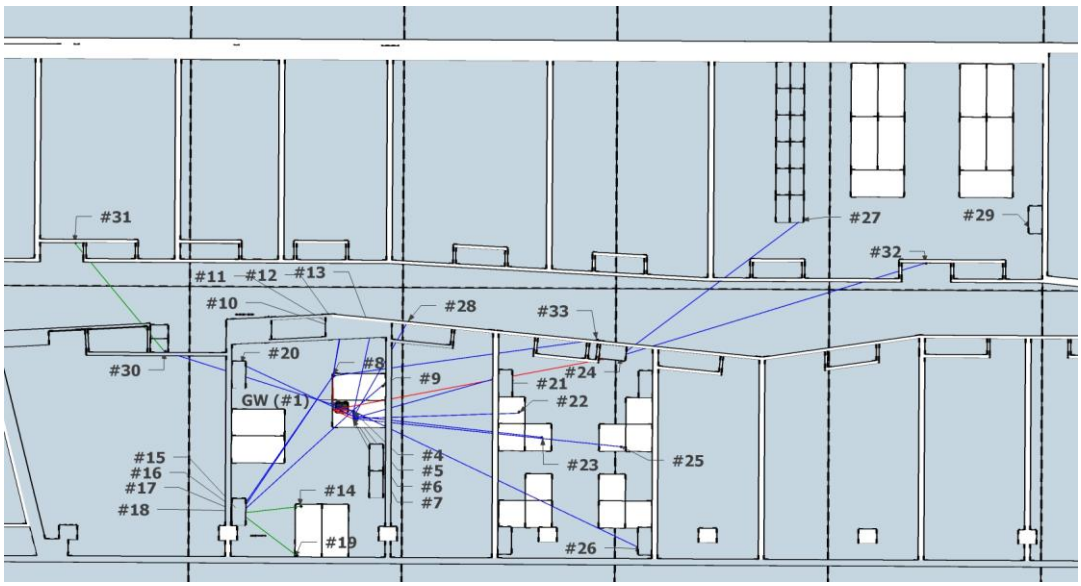


Figure 2.104: Network topology and links of the testbed considered in the current subsection

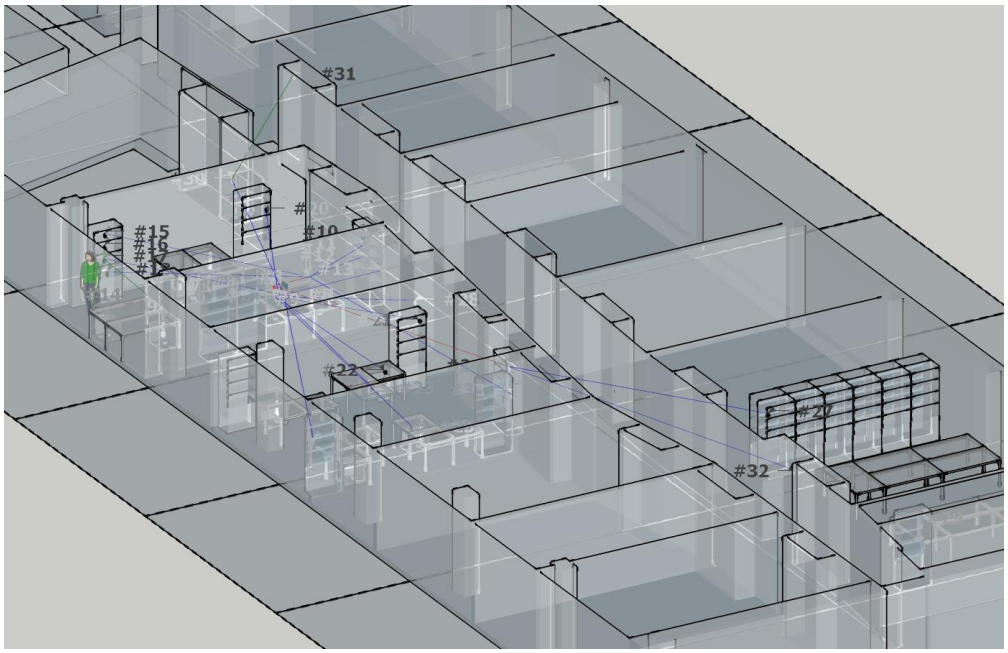


Figure 2.105: 3D view of the performed test (1)

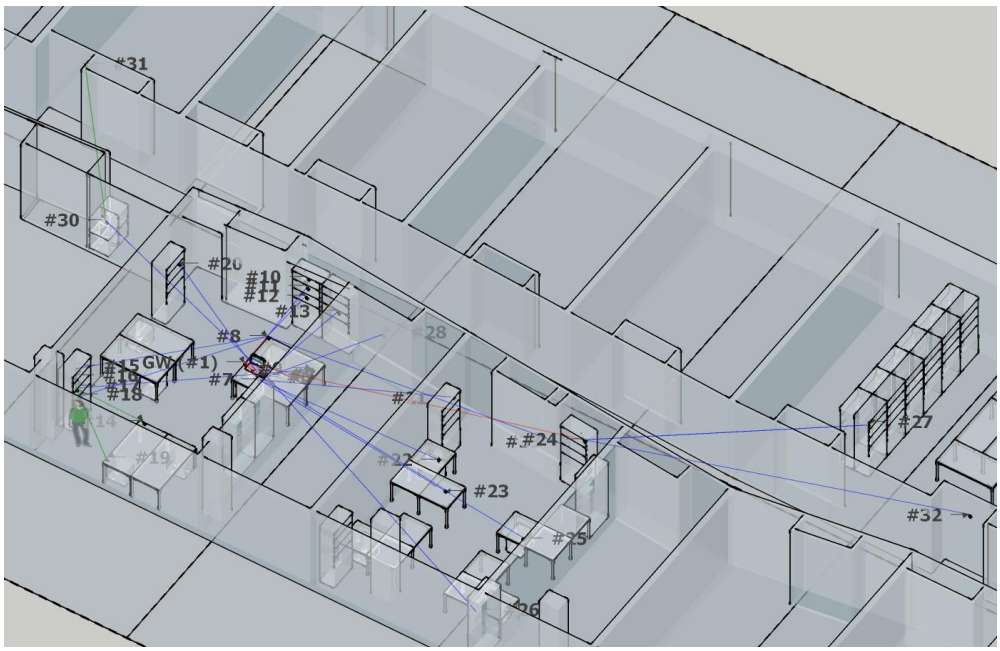


Figure 2.106: 3D view of the performed test (2)

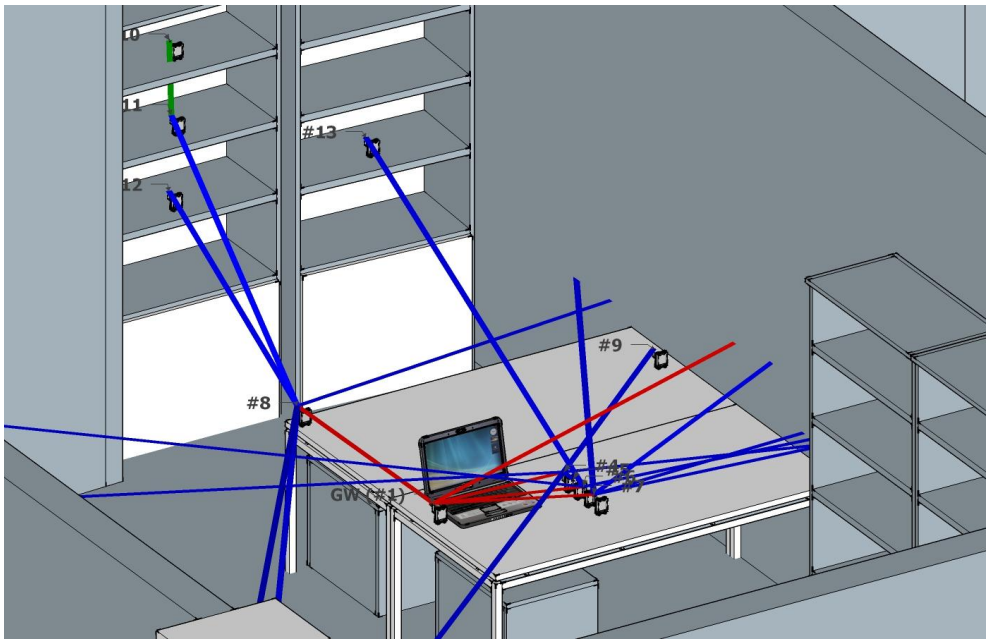


Figure 2.107: Detail of the network connections near the gateway

2.11.2.2 Tests with Zolertia Re-Mote

Performance evaluation of the ENTOMATIC protocol was performed in a testbed located in the 2nd floor, right wing of the Tanger building at UPF facilities[21]. The testbed consisted of 13 Zolertia RE-Mote nodes (one of them acting as a gateway and connected to a PC) running the ENTOMATIC protocol stack over IEEE 802.15.4.

All tests were executed considering no mobility and with the same STAs' placement (see Figure 2.108). All STAs were powered by an 800 mAh battery except the gateway, which was continuously powered by

[21] https://www.upf.edu/campus/en/comunicacio/tanger.html

the PC. Results and metrics of tests were directly obtained from the gateway or thanks to the *statistics messages* periodically sent by STAs. These messages contain information about different metrics such as the number of packets sent and acknowledged, RTT delays, as well as power profiles of microprocessor and radio module.

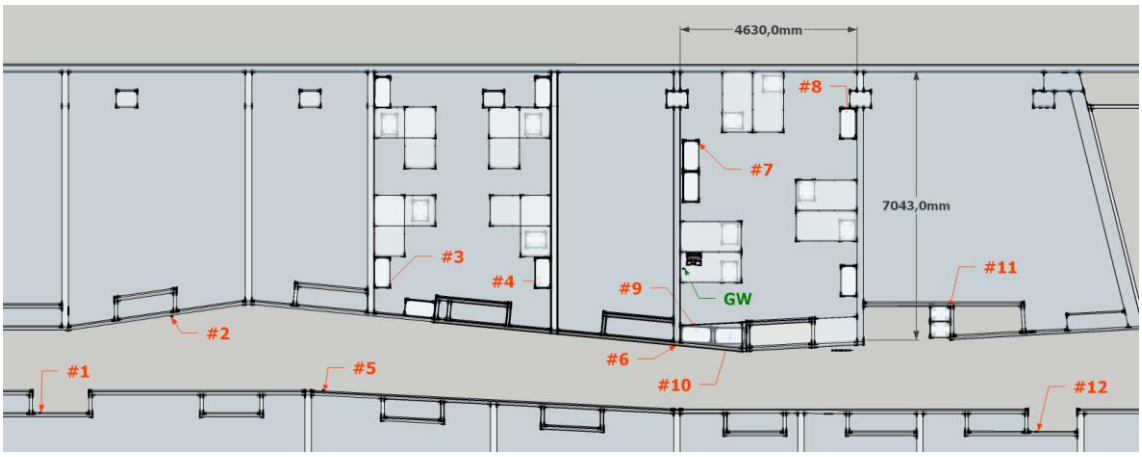


Figure 2.108: STA's placement at UPF facilities

The calculation of total energy consumption (E_T) is based on these two power profiles ($E_{\mu P}$ and E_{RADIO}, for the microprocessor and the radio module, respectively), as shown in the attached equations.

$$E_T = E_{\mu P} + E_{RADIO}$$
$$E_{\mu P} = V_{DD} \cdot (t_{CPU} \cdot I_{CPU} + t_{LPM} \cdot I_{LPM})$$
$$E_{RADIO} = V_{DD} \cdot (t_{RX} \cdot I_{RX} + t_{TX} \cdot I_{TX} + t_{SL} \cdot I_{SL})$$

V_{DD} is the supply voltage, and t and I are the time and the current corresponding to the following states: processing (CPU) and low power mode (LPM) for the microprocessor; receiving/listening to the channel (RX), transmitting (TX)[22], and sleeping (SL) for the radio module. All these values have been extracted from the datasheet of the employed microprocessor, the Texas Instruments CC2538 [16] and summarized in Table 2.51.

Table 2.51: Current values for the different operational states

	Operational State	Current
Microprocessor (µP)	Processing (CPU)	$I_{CPU} = 13\ mA$
	Low power mode (LPM)	$I_{LPM} = 0.4\ \mu A$
Radio module	Receiving (RX)	$I_{RX} = 19\ mA$
	Transmitting (TX)	$I_{TX} = 39 - 61\ mA$
	Sleeping (SL)	$I_{SL} = 0.12\ \mu A$

All tests begin with a *reassociation primary beacon* in which all nodes try to associate to the network. From them on, the GW emits a new (*reassociation* or *data*) primary beacon every $T_p = 3\ min$. *Data primary beacons* can ask STAs for a new *application* or *statistics packet*. In all our tests, *application* and *statistics packets* generated by STAs contain, respectively, 10 and 20 bytes of net information[23].

ASSOCIATION MECHANISM

[22] Current consumption in transmission depends on the power transmission level. See [16] for detailed information.
[23] Implementation of IEEE 802.15.4 in Contiki OS increases the minimum length of any transmitted packet up to 43 bytes after including headers and, if necessary, applying padding.

To prove the performance and the coherence of the proposed association mechanism (and its underlying routing) in a changing environment, the GW was configured to consecutively broadcast 200 sequences consisted of a *reassociation primary beacon* and a *data primary beacon*.

Nodes were thus forced every two beacons to renew their association to the network and compute their best parent limiting to 5 the number of children per STA and according to:

$$S(RSSI, r, ch) = w_1 \cdot RSSI_t + w_2 \cdot RSSI_r + w_3 \cdot r + w_4 \cdot c$$

with the following parameters: $w_1 = w_2 = 10$, $w_3 = 1$, and $w_4 = 5$.

Interspersed *data primary beacons* were used to check the reliability of routing paths and allow not yet associated STAs to have another opportunity to join the network.

Under these premises, and after 200 repetitions, an average number of 11.97 STAs were associated to the network after the *data primary beacon* of the given sequence (i.e., 99.75% of success). As for the packet delivery ratio (PDR), it achieved 100% for the associated STAs.

Routing tables compiled by the GW were processed and adapted to graphical representation in Figure 2.109, where line's thickness is proportional to link's frequency appearance. Preference of STAs for establishing paths with closer neighbours in their way to the GW becomes evident, just like the importance of *clear paths* (i.e., without obstacles) such as the formed by the floor corridor.

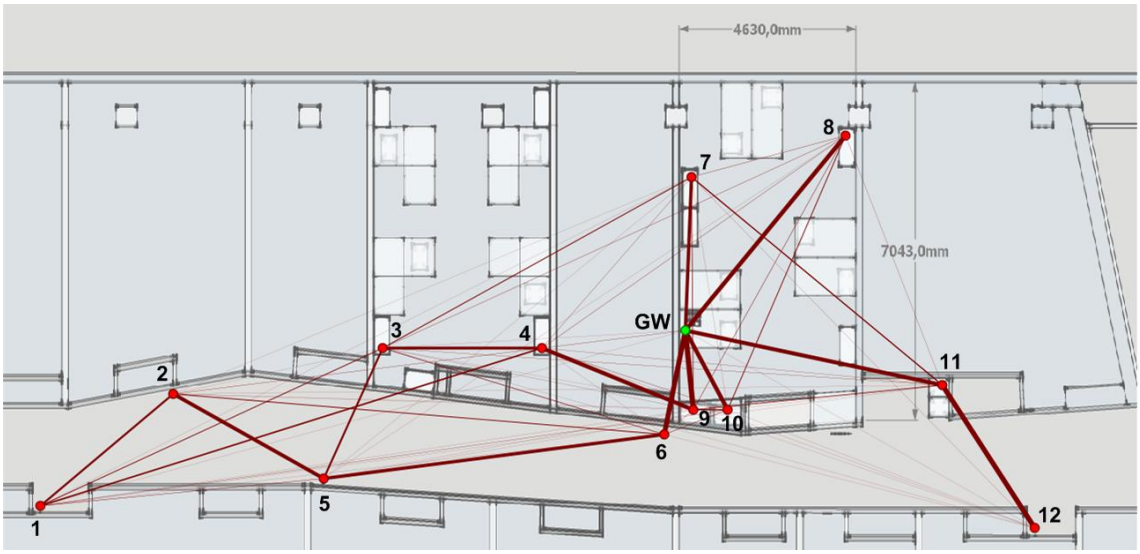


Figure 2.109: Association diagram when each STA admits up to 5 children

The limitation of 5 children can clearly be appreciated in STAs #6, #8, #9 and #10 being almost always directly connected to the GW in ring 1, while other STAs (principally #7 and #11) alternate the last remaining position of that ring.

RELIABILITY

Once all STAs are associated to the network and their paths to the GW properly established, the next goal is to analyze the reliability and the cost (in terms of energy consumption) of sending data. To do that, the GW was programmed to send 20 beacons with the following sequence: beacon #1 is a *reassociation beacon*, beacons #10 and #20 are *data primary beacons* asking for *statistics packets*; the rest of beacons are *data primary beacons* asking for *application packets*. To send their packets, STAs are provided with 5 transmission windows ($w = 5$).

In addition, different network configurations were applied. Firstly, two different MAC layers inherent to Contiki OS were tested: NULLMAC and X-MAC [28]. While NULLMAC maintains STAs awake during *active* periods, X-MAC combines the introduction of sleeping periods for receivers with the use of strobed preambles for senders.

Secondly, and always over the same node deployment, two different network topologies were tested: single hop and multi-hop. In the first case, all nodes are directly connected to the GW, while in the second case, STAs are free to establish their own routes to the GW with the single limitation of having 5 children per STA.

Table 2.52: Definition of error configurations for the proposed testbed

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

And thirdly, the whole system has been altered with the introduction of a certain error probability when sending application packets as well as to their corresponding ACKs (It is worth noting here that messages implied in the association process and statistics packets are not affected by these errors). A random function internally generated by STAs is responsible for allowing new transmissions, being always the error probability for ACK packets half of the value than that for application packets. Four different error configurations have been defined for this purpose (see Table 2.52).

The results with the obtained PDR in all these configurations are compiled in Figure 2.110 and Figure 2.111. After 5 transmission windows, PDR is in any configuration above 95%, and it even achieves values above 90% after 3 and 4 transmission windows when using X-MAC and NULLMAC, respectively. In this case, NULLMAC specially suffers from the effect of collisions, due to the backoff implementation[24] and the higher number of concurrently active STAs.

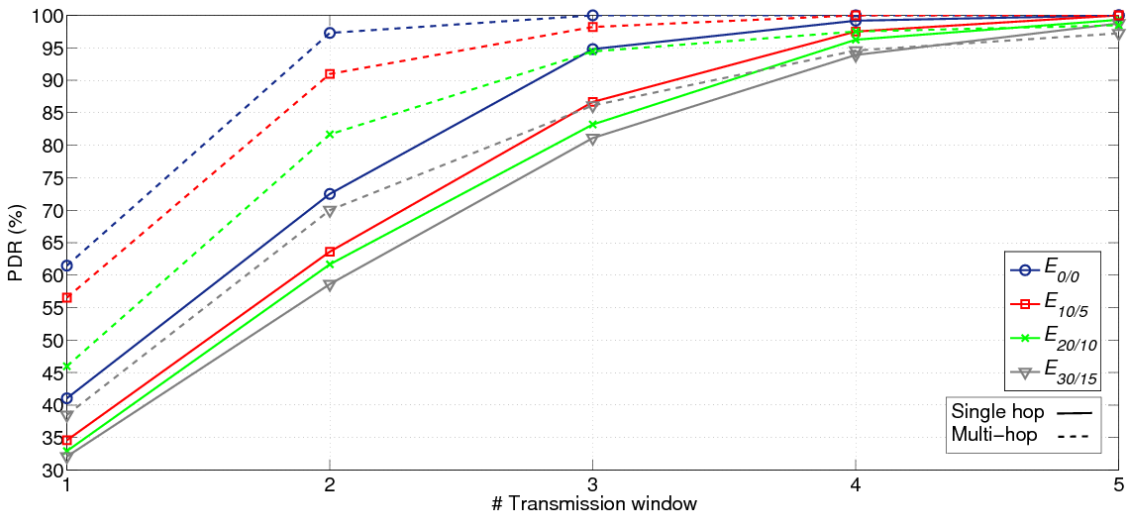


Figure 2.110: Packet delivery ratio (PDR) in the proposed testbed for NULLMAC layer

[24] Main values of the IEEE 802.15.4 CSMA default backoff implementation in Contiki OS: `macMinBE=0`, `macMaxBE=4`, and `macMaxCSMABackoffs=5`

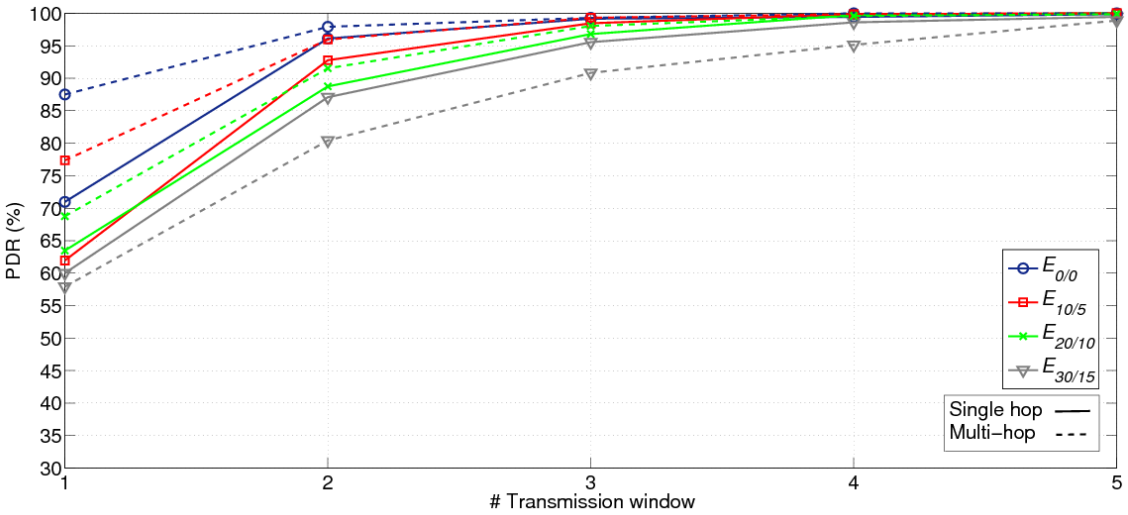


Figure 2.111: Packet delivery ratio (PDR) in the proposed testbed for X-MAC layer

Another insight from obtained results is how multi-hop topology outperforms single hop in all possible configurations except when using X-MAC with $E_{30/15}$. Again, the inherent reduction of concurrently active STAs competing for the channel during the same time period (in this case, due to the allocation of STAs to different slots according to their ring) proves beneficial for system's reliability.

The network's ability to properly deliver data packets to its destination was also analyzed by computing the quotient between the total number of packets sent by STAs and those properly received by the GW. As shown in Figure 2.112, multi-hop schemes still have better performance than single hop in low-error configurations. On the contrary, in highly unfavorable channels, multi-hop schemes with high dependence between parents and children (i.e., parents sending in each transmission window whatever they have gathered) could even require more retransmissions, as in X-MAC.

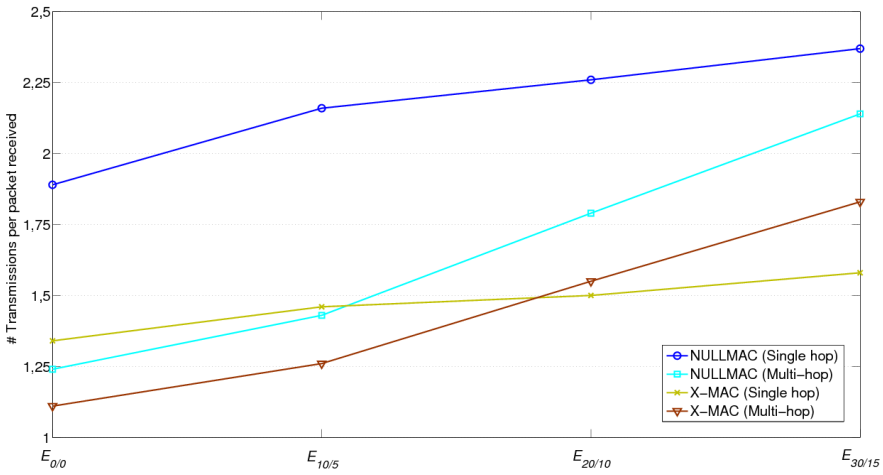


Figure 2.112: Number of transmissions per packet received in the proposed testbed

ENERGY CONSUMPTION

The effect of this interdependence can also be observed in total energy consumption (Figure 2.113) computed after the 20 transmitted beacons (i.e., after performing a 1-hour test). Important savings (up to 15%) can be achieved when using multi-hop schemes in low-error configurations $E_{0/0}$ - $E_{20/10}$ and similar or worse values in $E_{30/15}$.

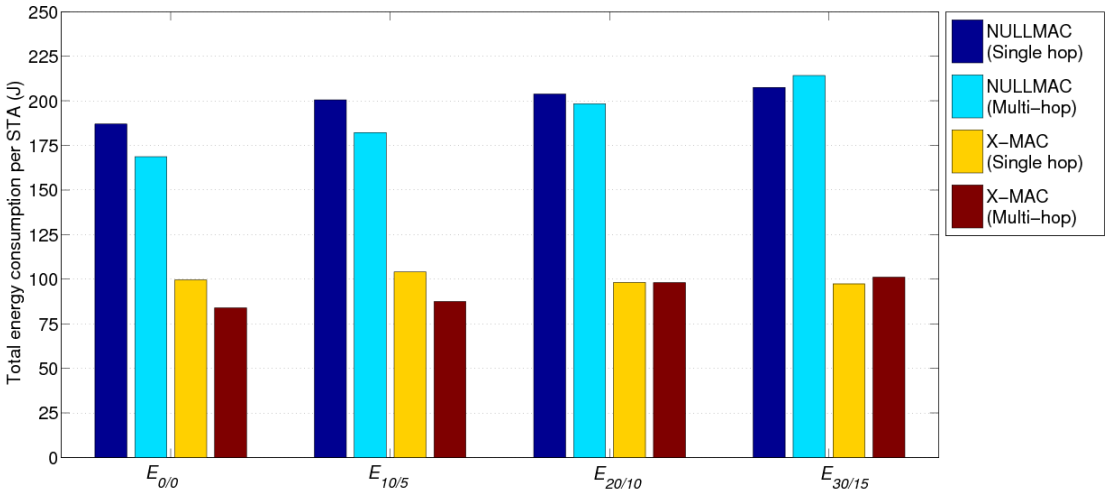


Figure 2.113: Average total energy consumption per STA after 20 beacons when $T_p = 3\ min.$

Sleeping time percentage of STAs' microprocessor has been in all studied cases above 97% for X-MAC and 99% for NULLMAC, due to the higher number of operations involved in the first case. However, the impact of radio module sleeping periods introduced by X-MAC layer reduces total energy consumption in up to 50% with respect to NULLMAC, whose STAs are always listening to the channel when being active. In this case, values of energy consumed per bit of payload delivered are confined between 50 - 65 mJ/bit for X-MAC, and 105 - 140 mJ/bit for NULLMAC.

As for the battery lifetime, Table 2.53 compiles the duration in days of the 800mAh battery included in the Zolertia RE-Mote for the current testbed with $T_p = 3\ min.$ as well as two estimations/extrapolations with $T_p = 1\ h.$ and $T_p = 4\ h.$ The temporal flexibility of the TDMA-based system employed allows this kind of extrapolations, by assuming that in the non-active time periods both the microprocessor and the radio module remain asleep.

Table 2.53: Lifetime of an 800 mAh battery running the described testbed

			Battery lifetime (days)		
			$T_p = 3\ min.$	$T_p = 1\ h.$	$T_p = 4\ h.$
NULLMAC	Single hop	$E_{0/0}$	2.37	47.35	187.87
		$E_{10/5}$	2.21	44.19	175.41
		$E_{20/10}$	2.18	43.46	172.53
		$E_{30/15}$	2.14	42.70	169.53
	Multi-hop	$E_{0/0}$	2.63	52.51	208.17
		$E_{10/5}$	2.44	48.60	192.79
		$E_{20/10}$	2.24	44.66	177.27
		$E_{30/15}$	2.07	41.35	164.23
X-MAC	Single hop	$E_{0/0}$	4.46	88.75	349.64
		$E_{10/5}$	4.27	85.00	335.07
		$E_{20/10}$	4.52	89.99	354.46
		$E_{30/15}$	4.56	90.79	357.55
	Multi-hop	$E_{0/0}$	5.29	105.17	413.16
		$E_{10/5}$	5.08	101.04	397.21
		$E_{20/10}$	4.53	90.10	354.85
		$E_{30/15}$	4.39	87.38	344.31

RESILIENCE AGAINST FAILURES

To prove the adaptability and resilience of the implemented routing protocol, the network was subjected to the deliberate shutdown of two of its STAs. In this way, the GW was programmed to send 50 beacons with the following sequence: beacon #1 is a *reassociation beacon*, beacons multiple of 10 are *data primary beacons* asking for *statistics packets*; the rest of beacons are *data primary beacons* asking for *application packets*. In addition, the disassociation mechanism was programmed in the GW to remove an STA from the network if not receiving any data packet during one *primary beacon*.

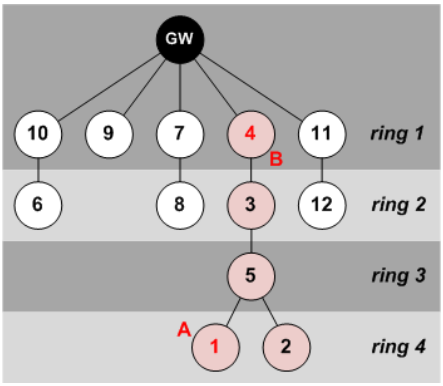


Figure 2.114: Logical network topology after the *reassociation beacon*

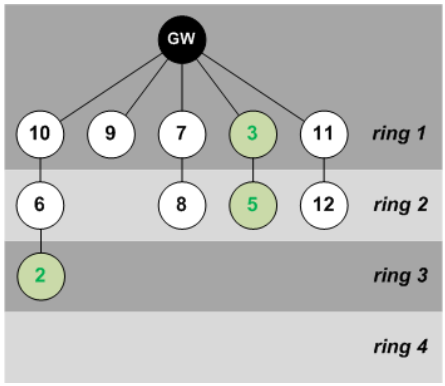


Figure 2.115: Logical network topology from beacon #15 until beacon #50

Once finished the initial *reassociation* stage, the network is organized in four rings, as shown in Figure 2.114. After beacon #4 **(A)**, STA #1 was switched off but it did not imply further problems to the network, as this STA did not have any children. However, after beacon #12 **(B)**, STA #4 was also switched off and it forced the network to reconfigure itself. The path to the GW of STAs #2, #3 and #5 is broken and they have to look for a new route in the association phase of successive *primary beacons*. After beacon #15, all active STAs (i.e., all of them except #1 and #4) have a path to the GW and the network is again stabilized (see Figure 2.115).

This test is also useful to analyze the performance of the proposed power regulation mechanism. Thus, by setting $RSSI_{min} = -110\,dBm$ and $RSSI_{max} = -100\,dBm$, the temporal evolution of the transmission power in function of the primary beacon number from Figure 2.116 is obtained. It is worth noting here that the Zolertia RE-Mote devices used have 31 different power levels (from -16dBm to 14dBm with steps of 1 dB [18]) and STAs are programmed to use by default the maximum power level.

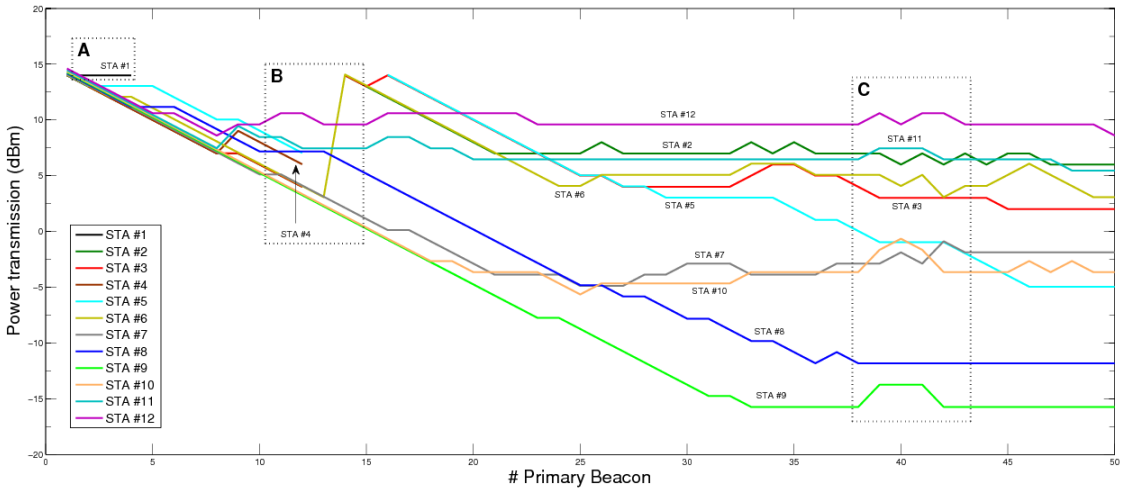


Figure 2.116: Temporal evolution of STAs' power transmission level

The reduction in the transmission power level is clear in most of the analyzed STAs, being the most significant examples STAs #7, #8, #9 and #10; the nearest ones to the GW. This fact results in a lower energy consumption, as $I_{TX} = 61\ mA$ when transmitting at 14 dBm, but almost half ($I_{TX} = 39\ mA$) when doing it at -16 dBm.

The effects of switching off nodes are also visible in the transmission power, as shown in **(A)** and **(B)** from Figure 2.116. While STA #1 in **(A)** simply stops working, nodes involved in the shutdown of STA #4 in **(B)** experience notable changes. Thus, STAs #2, #3 and #5 *disappear* along with the shutdown of STA #4. However, they become associated again during beacons #13 and #15 with maximum transmission power. STA #6, in turn, when becoming parent of #2 sets the maximum power level to establish connection with its new child.

Lastly, the power regulation mechanism proves its good performance against channel alterations as shown in area **(C)**. In this case, and due to the test execution on a real scenario, the presence of people in the floor corridor may have disturbed the channel conditions. To overcome this issue, some STAs (#9, #10, #11 and #12) select temporarily greater power transmission levels that are reestablished once finished the detected problem.

2.11.3 RANGE COVERAGE TESTS

One of the tasks performed during this period is the performance of different range coverage tests at different orchards. The Re-Mote has been deployed at different outdoor scenarios in order to collect information on the range coverage and propagation loses in real orchard fields. This different tests has been performed to guarantee the necessarily range to cover orchards of different sizes.

Basically, the requirements of the project state that a trap should be place every hectare. The maximum distance between two nodes at the middle of two adjacent hectares becomes, approximately, 280 meters. Hence, the radio equipment of the traps should easily cover this distance.

Moreover, we also take the chance to obtain a propagation model. The idea is to obtain a general idea of the signal propagation in outdoor scenarios. We use the simplified Path Loss Model [20] where the RSSI is modeled as:

$$P_r(dBm) = P_t(dBm) + K(dB) - 10\gamma \log_{10}\left(\frac{d}{d_0}\right)$$

Where K is a unitless constant which depends on the antenna characteristics and the average channel attenuation, d0 is a reference distance for the antenna far-field, and γ is the path loss exponent.

In that sense, we performed different measurements in order to model the propagation characteristics of the RSSI collect from two Zolertia RE-Mote nodes. At the following images we show the different experiments that we performed at Falset.

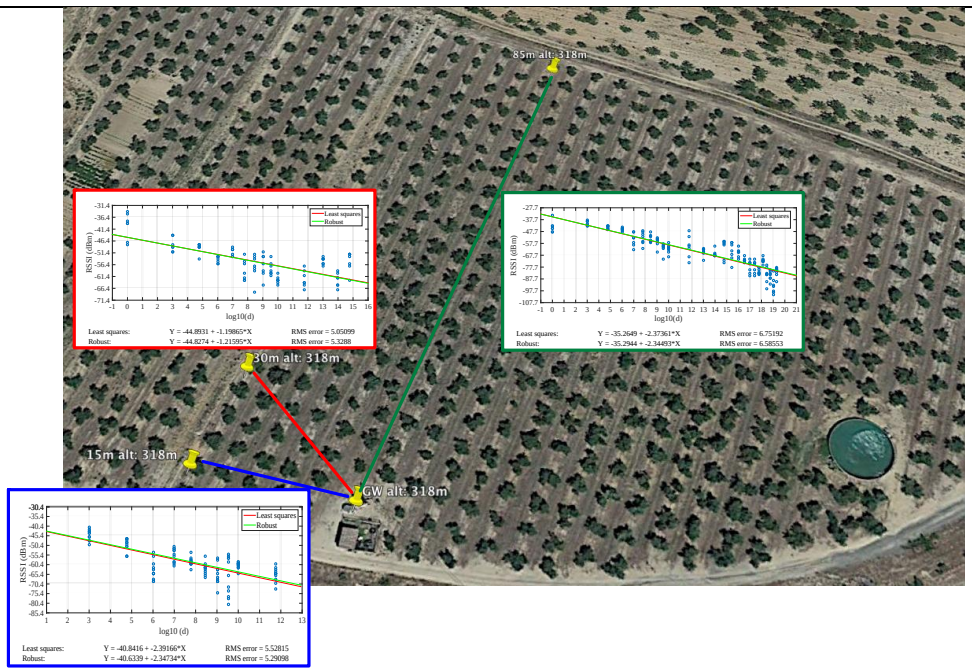


Figure 2.117: Test performed in an orchard in Falset (Picture 1)

These first tests show us the variability of the RSSI measurements. The different parameters obtained by the model differ from one scenario to the other.

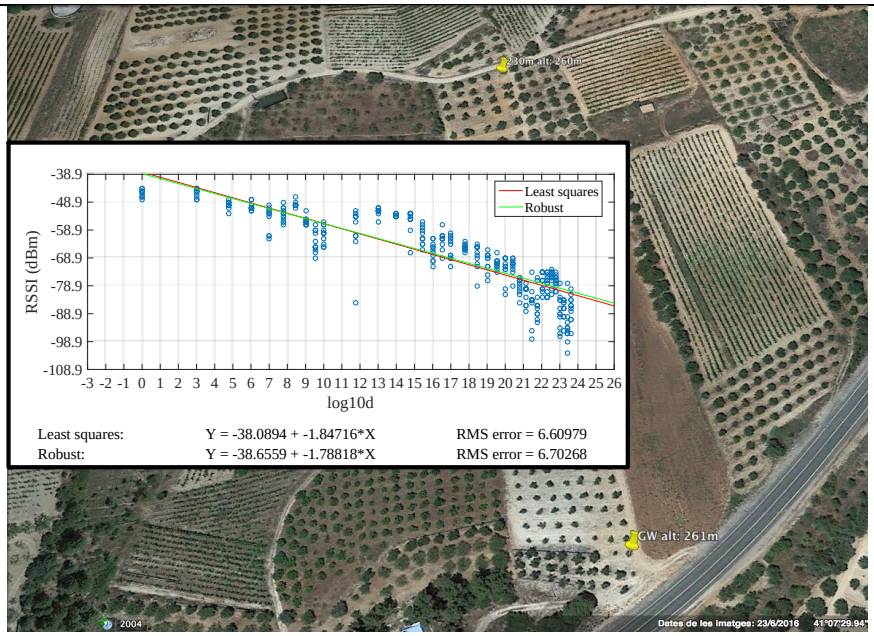


Figure 2.118: Test performed in an orchard in Falset (Picture 2)

At this second round of measurements, although the model parameters are not the same as in previous measurements, values are almost the same. Moreover, as we expect the characterization of the propagation signal is not a trivial issue. In order to obtain a precise model it is necessary to performed long measurements campaigns and it can be also be used more complex models that can parameterizes better the different interferences that signals can suffer. Moreover, in this project with this general analyze of the propagation is enough.

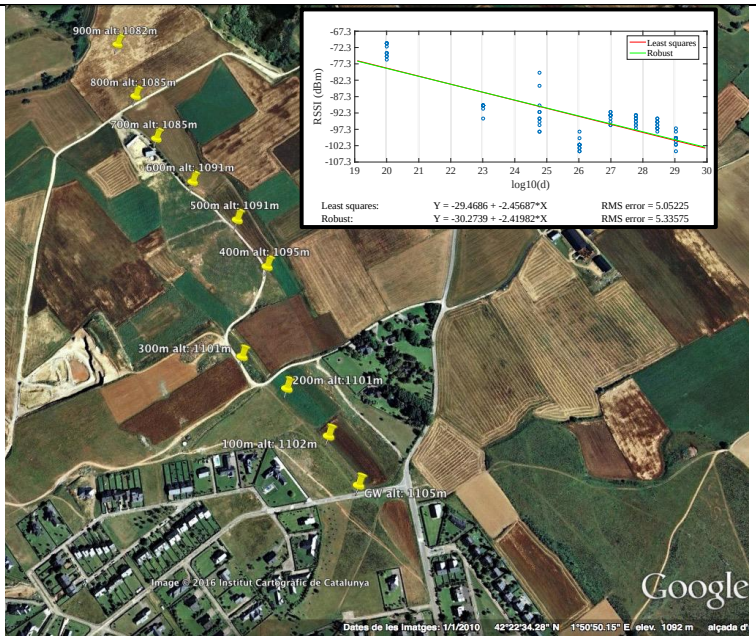


Figure 2.119: Test performed in La Cerdanya

Lastly, we have performed a long-range test in order to be sure that the radio equipment could cover a minimum distance of 280 meters. The characteristics of the orchards of Falset does not allow to us to cover large distances. Falset's orchards are mainly small fields due to the orography characteristics of area of Priorat, where Falset is located.

For that reason we have performed this last test at a vast area in the region of *La Cerdanya*, at the Catalan Pyrenees. In this occasion the distance achieved is around 900 meters with NLOS condition. At this distance the number of packets received diminishes, hence, we can assure that for distances lower than 900 meters. This distance outperforms the minimum requirement of 280 meters that guarantees that two traps placed in adjacent hectares can communicate; even further nodes can establish communication between them. Following the philosophy of the communication protocol designed and developed in this work package, nodes adapt their links automatically depending on the characteristics of the scenario. Hence, the larger the coverage, the larger the possible links a node could select.

2.11.4 FALSET OLIVE FIELD

On 10-11 March 2016, UPF performed performance tests of the ENTOMATIC wireless sensor network in several selected olive fields of the *Falset-Marçà Cooperative* [25] from the town of Falset (Catalonia).

Falset is the capital of the Priorat region in Catalonia, Spain. The central part of the comarca, "Priorat històric," produces the famous and prestigious wine of the Denominació d'Origen Calificada Priorat. Wines from elsewhere in the comarca are denominated as *Montsant*. The area is also known for the production of hazelnuts.

[25] Falset-Marçà cooperative - http://www.etim.cat/

Figure 2.120: Priorat region location within Catalonia (Spain)

Priorat had a steady loss of population during the 20th century, but has recently as of 2004 experienced a more prosperous economy, resulting in an end to this trend. In 2001, the population was 9,196, with only the capital (Falset) exceeding a population of 1,000.

Priorat has an area of 496 km² and is bordered by the River Ebro, and by the regions of Ribera d'Ebre, Baix Camp, les Garrigues, and Conca de Barberà. The region is mostly hilly, and in its extreme north is the Montsant mountain range, rising to over 1000 m; the south is bordered by the Mola de Colldejou and the mountain ranges of Serra de Llaberia and Serra de Santa Marina. The climate is continental: dry and hot in summer, cold in the winter.

The *Falset-Marçà Cooperative (DO Montsant)* is the result of the merger between the cooperatives of Marçà and Falset, two towns separated by only three kilometers by road in the southern part of the Priorat region. The production of high quality wine and oil is its main activity.

Tests performed in the olive fields from the *Falset-Marçà Cooperative* were useful to validate the proper operation of most mechanisms implemented in the ENTOMATIC protocol stack. However, some errors regarding the association process were detected, which had strong impact on the subsequent packet transmissions. No further conclusive results were obtained, as this first version of the system only included some minor performance metrics.

Figure 2.121: Test Oliveyard #1

Figure 2.122: Test Oliveyard #2

Two different olive fields were selected to perform the validation of the ENTOMATIC system. Up to 30 nodes Crossbow TelosB were deployed on them, with one of them acting as a gateway and directly connected to a PC (as shown in Figure 2.123), which gathered and computed performance metrics generated by the network.

Figure 2.123: PC directly connected via USB cable to a Crossbow TelosB acting as a gateway

Figure 2.124: Detail view of one Crossbow TelosB placed on the branch of an olive tree

Figure 2.125: View of one of the olive fields from the *Falset-Marçà Cooperative*

Experience obtained from the tests performed in Falset in March 2016 greatly helped us to develop new mechanisms and improve the already implemented in order to increase the transmission reliability of the ENTOMATIC communication protocol, as can be seen in further tests with advanced versions of the system on Zolertia RE-Mote devices.

As part of the consisting tasks of the ***WP7: Integration of the prototype ENTOMATIC system,*** it is planned to go again to Falset and test the whole ENTOMATIC communication platform (i.e., the WSN and the cellular link to the data receiver server) on the Falset-Marçà Cooperative fields during the first trimester of 2017.

3 GATEWAY TO CLOUD SERVER COMMUNICATION

3.1 GENERAL CONSIDERATIONS

The remote, scattered and isolated positions of traps (and gateways) on olive orchards complicate the use of WiFi technologies to communicate themselves. Another issue to have into account when using WiFi networks is energy consumption, which can be much higher than in Wireless Sensor Networks. Finally, the lack of availability of WiFi access points and backbones in rural areas encourages the deployment of cellular networks (3G, GPRS) to perform communication from the gateway to the cloud server.

3.1.1 NETWORK TOPOLOGY

The network topology chosen for the gateway to cloud communication is a *point to point topology*. This is the simplest way to establish a permanent link between two endpoints. Switched point-to-point topologies are the basic model of conventional telephony. The value of a permanent point-to-point network is unimpeded communications between these two endpoints.

Easiest to understand, of the variations of point-to-point topology, is a point-to-point communications channel that appears, to the user, to be permanently associated with the two endpoints. A children's tin can telephone is one example of a physical dedicated channel.

The mobile cellular networks are the best solution for this part of the ENTOMATIC network, so that each one of the gateways used around the world (in turn gathering data from its corresponding Wireless Sensor Network) will be connected to the central server by means of a cellular connection, as shown in Figure 3.1.

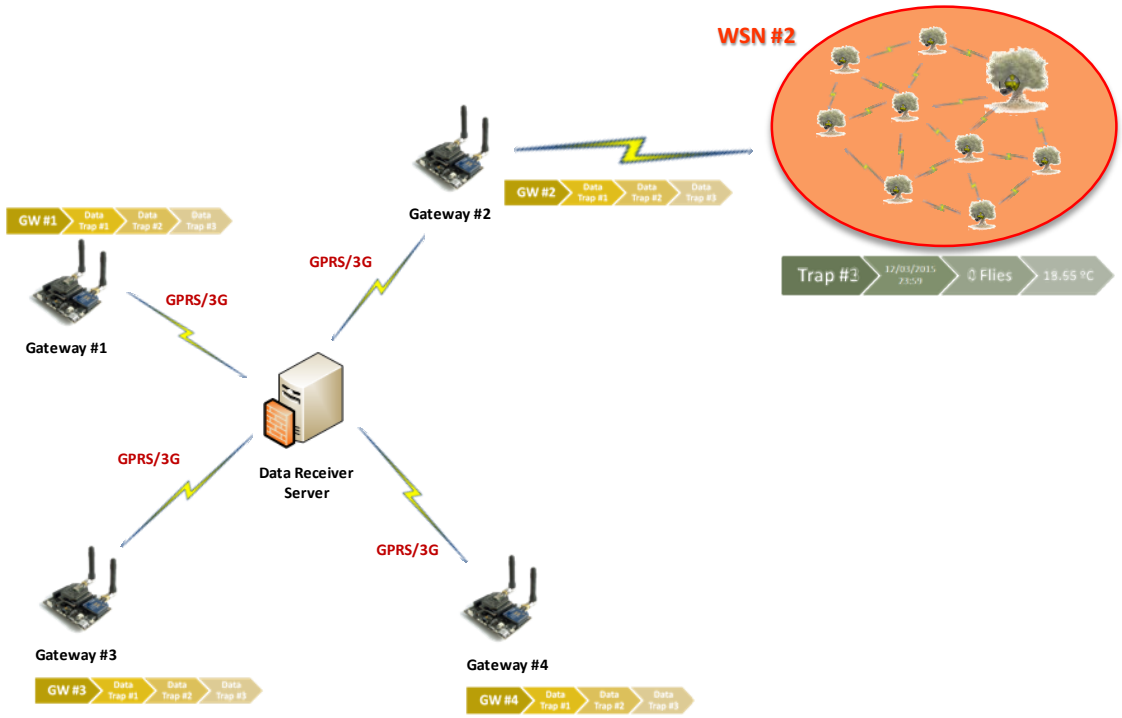


Figure 3.1: Proposed connection between WSN and Data Receiver Server

3.2 HARDWARE PLATFORM

The gateway will be the only element of the ENTOMATIC network with two different network interfaces:

1. An 868 MHz interface based on a Zolertia RE-Mote board which will let the gateway communicate with the rest of nodes of the Wireless Sensor Network
2. A cellular interface to directly communicate with the cloud server via a point to point connection

For the second purpose, three different cellular boards have been analyzed as possible candidates for sending ENTOMATIC network data from the gateway to a central server through a cellular network:

A. SIM5218 (3G technology)

The SIM5218 series is a Tri-Band/Single-Band HSPA/WCDMA and Quad-Band GSM/GPRS/EDGE module solution which supports up to 7.2Mbps downlink speed and 5.76Mbps uplink speed services. It has strong extension capability with rich interfaces including UART, USB2.0 high speed, Embedded SIM Card, SD Card, Camera sensor, GPS, etc.

With abundant application capability like embedded LUA script, TCP/UDP/FTP/HTTP/SMTP/POP3 and MMS, it will add much value to customers' application. It is ideal for a wide range of products including USB Modems, gateways, router, PDA, Video phone, and much more.

B. SIM900 (GPRS technology)

SIMCom presents an ultra compact and reliable wireless module-SIM900. This is a complete Quad-band GSM/GPRS module in a SMT type and designed with a very powerful single-chip processor integrating AMR926EJ-S core, allowing you to benefit from small dimensions and cost-effective solutions.

Featuring an industry-standard interface, the SIM900 delivers GSM/GPRS 850/900/1800/1900MHz performance for voice, SMS, Data, and Fax in a small form factor and with low power consumption. With a tiny configuration of 24mm x 24mm x 3 mm, SIM900 can fit almost all the space requirements in your M2M applications, especially for slim and compact demands of design.

C. SIM908 (GPRS + GPS technology)

SIM908 module is a complete Quad-Band GSM/GPRS module which combines GPS technology for satellite navigation. The compact design which integrated GPRS and GPS in a SMT package will significantly save both time and costs for customers to develop GPS enabled applications. Featuring an industry-standard interface and GPS function, it allows variable assets to be tracked seamlessly at any location and anytime with signal coverage.

Table 3.1: Comparison table of the different cellular platforms analyzed

Model	3G (SIM5218)	GPRS (SIM900)	GPRS+GPS (SIM908)
Download speed	7.2Mbps	0.02Mbps	0.02Mbps
Upload speed	5.5Mbps	0.01Mbps	0.01Mbps
FTP	Yes	Yes	Yes
FTPS (Secure)	Yes	No	No
HTTP	Yes	Yes	Yes
HTTPS (Secure)	Yes	No	No
TCP/UDP Sockets	Yes	Yes	Yes
Mails	Yes	No	No
GPS	Yes (A-GPS + S-GPS + NMEA)	No	Yes, but only NMEA stand-alone mode
Video Camera	Yes (Photo + Video)	No	No
Video Calls	Yes	No	No
Protocols	3G, WCDMA, HSPA, UMTS, GPRS, GSM	GPRS/GSM	GPRS/GSM
Frequency Bands	850, 900, 1800, 1900, 2100MHz	850, 900, 1800, 1900MHz	850, 900, 1800,1900MHz
SD Card	Yes (up to 32GB)	No	No

After having considered the requirements of the ENTOMATIC system in terms of data capacity and latency as well as detecting the limitations of energy consumption and price for the 3 aforementioned boards, we propose the use of the SIM900 module in the current project.

3.2.1 SIM900 GSM/GPRS MODULE

The SIM900 is a complete Quad-band GSM/GPRS solution in a SMT module which can be embedded in the customer applications.

Featuring an industry-standard interface, the SIM900 delivers GSM/GPRS 850/900/1800/1900MHz performance for voice, SMS, Data, and Fax in a small form factor and with low power consumption. With a tiny configuration of 24mm x 24mm x 3 mm, SIM900 can fit almost all the space requirements in your M2M application, especially for slim and compact demand of design.

- SIM900 is designed with a very powerful single-chip processor integrating AMR926EJ-S core
- Quad - band GSM/GPRS module with a size of 24mmx24mmx3mm
- SMT type suit for customer application
- An embedded Powerful TCP/IP protocol stack
- Based upon mature and field-proven platform, backed up by our support service, from definition to design and production

Table 3.2: Main specifications of the SIM900 GSM/GPRS module

General features	
Quad-Band 850/ 900/ 1800/ 1900 MHz	
GPRS multi-slot class 10/8	
GPRS mobile station class B	
Compliant to GSM phase 2/2+	Class 4 (2 W @850/ 900 MHz)
	Class 1 (1 W @ 1800/1900MHz)
Dimensions: 24* 24 * 3 mm	
Weight: 3.4g	
Control via AT commands (GSM 07.07 ,07.05 and SIMCOM enhanced AT Commands)	
SIM application toolkit	
Supply voltage range 3.4 ... 4.5 V	
Low power consumption	
Operation temperature: -30 °C to +80 °C	
Specifications for fax	
Group 3, class 1	
Specifications for data	
GPRS class 10: max. 85.6 kbps (downlink)	
PBCCH support	
Coding schemes CS 1, 2, 3, 4	
CSD up to 14.4 kbps	
USSD	
Non transparent mode	
PPP-stack	

Specifications for SMS via GSM / GPRS	
Point-to-point MO and MT	
SMS cell broadcast	
Text and PDU mode	
Drivers	
MUX Driver	
Specifications for voice	
Tricodec	Half rate (HR)
	Full rate (FR)
	Enhanced Full rate (EFR)
Hands-free operation (Echo suppression)	
AMR **Half Rate (HR)** **Full Rate (FR)**	
Interfaces	
Interface to external SIM 3V/ 1.8V	
Analog audio interface	
RTC backup	
SPI interface	
Serial interface	
Antenna pad	
I2C	
GPIO	
PWM	
ADC	
Compatibility	
AT cellular command interface	
Compatibility	
CE, FCC, ROHS, PTCRB, GCF, AT&T, IC, TA	

3.2.2 INTEGRATED BOARD

As preliminary task of ***WP7: Integration of the prototype ENTOMATIC system***, a first prototype of a gateway (i.e., a device with WSN and GPRS capabilities) has been designed and manufactured. Thus, a single device powered by a battery fulfils the requirements of the central network element. In addition, the board in which all the electronics are embedded contains an additional Li-Ion battery solar charger to harvest solar energy and thus enlarging battery lifetime.

One of the main advantages of the designed board is its flexibility, as it can be used not only as gateway, but also as node station by means of the integrated serial port connector (UART). As it can be seen in Figure 3.2 and Figure 3.3, the same board is used for both purposes by only changing the element

connected to the UART; the flies sensor in the case of the trap (node station), and the GPRS SIM 900 in the case of the gateway.

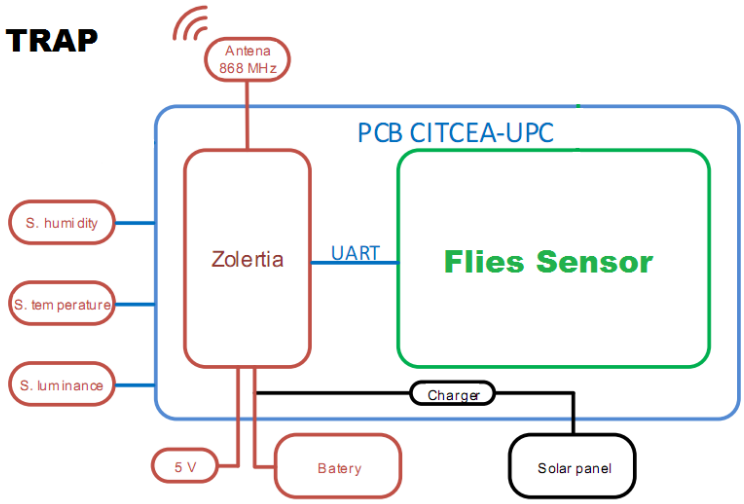


Figure 3.2: Functional diagram of the ENTOMATIC trap electronic elements

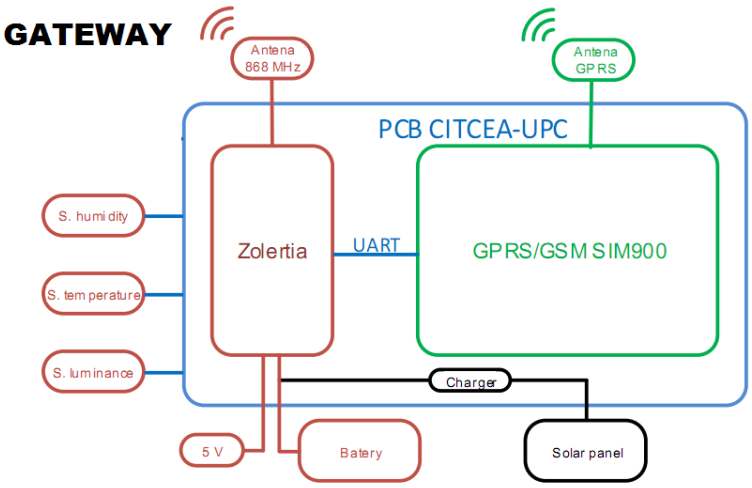


Figure 3.3: Functional diagram of the ENTOMATIC gateway electronic elements

The mechanical design of the PCB has been adapted to the dimensions of the proposed McPhail trap, so that all its elements can be properly and easily connected and disconnected. Figure 3.4 and Figure 3.5 show the PCB mechanical design and a render picture, respectively. The Zolertia RE-Mote device (the common element to node stations and gateway) is easily connected to the designed PCB by means of two parallel pin-outs, as shown in Figure 3.6.

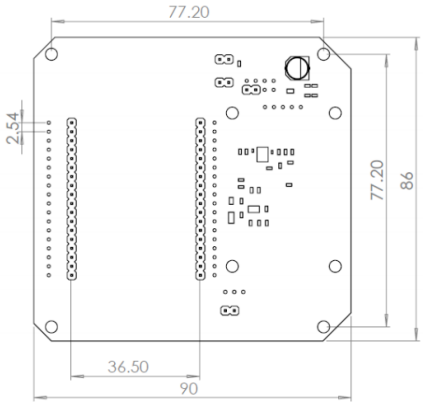


Figure 3.4: PCB mechanical design

Figure 3.5: PCB render picture

Figure 3.6: Designed PCB and Zolertia RE-Mote assembling

Figure 3.7 shows the PCB acting as a gateway (i.e., with the GPRS SIM 900 module connected) and with a solar panel connected. It is worth noting here that the battery has been intentionally located on the Zolertia RE-Mote for visualization purposes, although its final location is the gap between the Zolertia RE-Mote and the PCB.

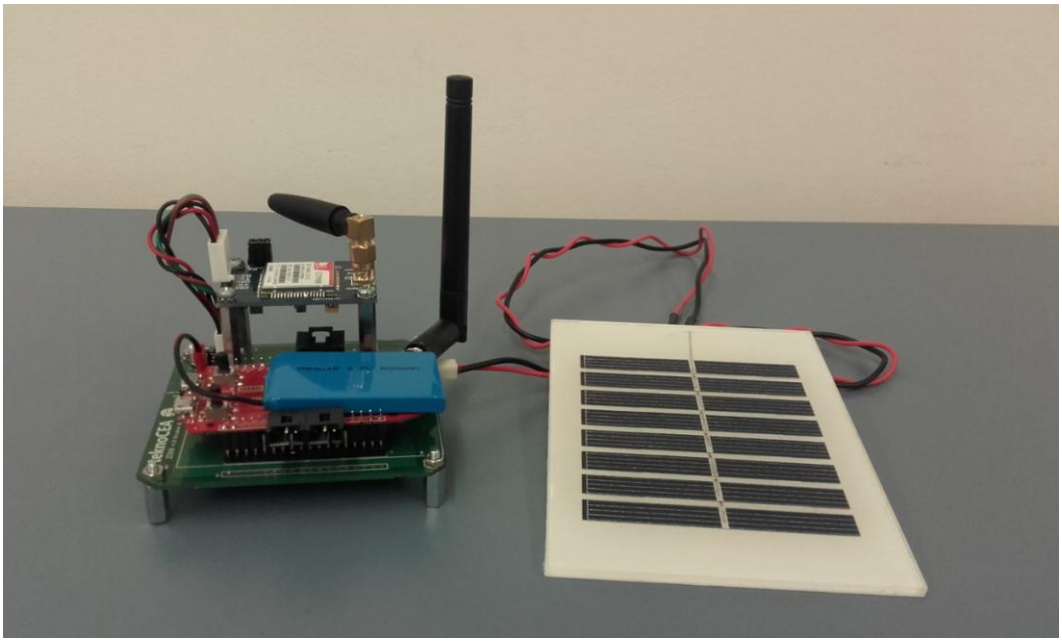


Figure 3.7: Integrated board acting as gateway

3.3 COMMUNICATION PROTOCOL

The communication protocol performed between the gateway and the cloud server should not disturb the normal operation of the gateway as part of the Wireless Sensor Network. Due to this reason, the gateway will wait until the end of the last transmission window of each beacon period to initiate any kind of communication with the server.

3.3.1 CLIENT-SERVER MODEL

The client–server model is used between ENTOMATIC gateways and the data server, where a distributed application structure partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients. In our case, the ENTOMATIC cloud server acts as the ***server,*** and the ENTOMATIC gateways as the ***clients***.

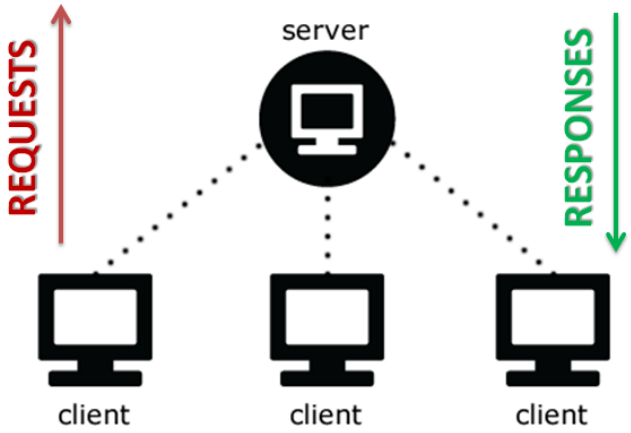


Figure 3.8: Client-server communication paradigm

Clients and servers exchange messages in a request–response messaging pattern: The client sends a request, and the server returns a response. This exchange of messages is an example of inter-process communication. To communicate, the computers must have a common language, and they must follow rules so that both the client and the server know what to expect. The language and rules of communication are defined in a communications protocol.

3.3.1.1 HTTP Protocol

HTTP (Hypertext Transfer Protocol) is the protocol used in every transaction on the World Wide Web. It is a transaction-oriented protocol and follows the request-response scheme between a client and a server. The client which makes the request (a web browser or, in our case, a Zolertia RE-Mote) is known as a "user agent". The transmitted information is called a resource and it is identified by a Uniform Resource Locator (URL). Resources can be files, the result of the execution of a program, a query to a database, the automatic translation of a document, etc.

An HTTP session is a sequence of request-response transactions in a network. Thus, an HTTP client makes a request by establishing a TCP connection to a specific port on an HTTP server (typically port 80). The HTTP server, in turn, will keep listening to that port, waiting for the client's request message.

Once a request has been received, the HTTP server returns a status line, such as "`HTTP/1.1 200 OK`"; i.e., a set of headers that provide information about the current transaction and a response message. The content of this response message is typically the resource claimed by the HTTP client; an error message or other information could also be included.

HTTP defines different methods to indicate the action to be performed on a specific resource. What this resource represents, whether it is preexisting data or data that is dynamically generated, will depend on

the server implementation. The HTTP/1.1 specification maintains the methods already defined by the HTTP/1.0 specification (i.e., GET, POST, HEAD) and defines 5 new methods (OPTIONS, PUT, DELETE, TRACE and CONNECT):

- **GET:** Returns the resource identified in the requested URL.
- **HEAD:** Works like GET, but without the server returning the body of the message. That is, only the header information is returned.
- **POST:** Indicates the server to prepare for receiving information from the client. It is often used to send information from forms.
- **PUT:** Sends the resource identified in the URL from the client to the server.
- **OPTIONS:** Returns the HTTP methods that the server supports.
- **TRACE:** Echoes back to the client whatever string has been sent to the server, and is used mainly for debugging purposes.
- **DELETE**: Allows a client to delete a file on the web server.
- **CONNECT:** This method could allow a client to use the web server as a proxy.

Due to its simplicity for sending and receiving information, the GET method of the HTTP/1.1 specification will be used in the gateway-server communication of the ENTOMATIC project.

3.3.1.2 GET Method

The GET method is used to retrieve data from a web server by specifying parameters in the URL part of the request. It is the main method for retrieving data from a web server.

The structure of a request via GET method is defined as follows,

```
/test/demo_form.asp?name1=value1&name2=value2
```

where the URL is specified before the '?' character and next are attached the set of pairs of variable names and their values.

As for the ENTOMATIC system, whenever the gateway has any messages to transmit to the gateway, it will make a GET request which will include the parameters of the preconfigured actions as well as the corresponding data.

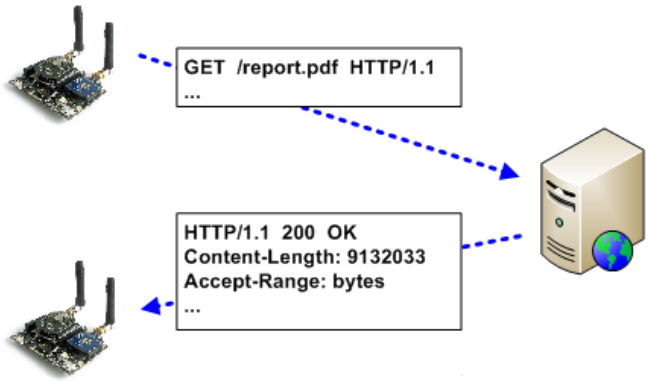


Figure 3.9: Operation diagram of a GET method request and response running over the HTTP protocol in a typical client-server communication

3.3.2 COMMUNICATION METHODS

All client-server protocols operate in the application layer. The application-layer protocol defines the basic patterns of the dialogue. To formalize the data exchange even further, the server may implement an API (such as a web service). The API is an abstraction layer for such resources as databases

and custom software. By restricting communication to a specific content format, it facilitates parsing. By abstracting access, it facilitates cross-platform data exchange.

Table 3.3: List of actions used by the communication protocol

HTTP Method	Name	URL	Action
GET	/newGateway	/Ga	Creates a new gateway
GET	/newSensor	/Se	Creates a new sensor
GET	/newMeasurement	/Me	Creates a new measurement
GET	/newAlarm	/Al	Creates a new alarm

A set of actions has been defined for the ENTOMATIC API, so that gateways and the data receiver server can easily communicate themselves. The definition of these actions can be found in the following lines.

3.3.2.1 New Gateway [Ga]

The gateway is the link where the WSN sends messages to the Internet. When a new gateway is connected to the WSN, a new petition is made to the server.

- **[Ga] HTTP REQUEST**

  The ***newGateway [Ga]*** method registers a new Gateway in the data receiver server.

  http://test.entomatic.upf.edu/Ga?mg=⑯&la=③.③&lo=③.③

  *(*Numbers within circles correspond to the maximum number of bytes contained in the data field.)*

Table 3.4: Proposed *Ga* method request fields

Field	Purpose	Maximum size (bytes)	Example
mg	MAC address of the Gateway	16	00124b0006160f60
la	Latitude coordinate of the Gateway	3+3 = 6	41.40
lo	Longitude coordinate of the Gateway	3+3 = 6	2.202

- **[Ga] SERVER RESPONSE**

  The server responses, in turn, with the following data frame.

  OK 0|timestamp|wg|
  No OK 1|timestamp|

Table 3.5: Proposed *Ga* method response fields

Field	Purpose	Maximum size (bytes)	Example
timestamp	Timestamp with the current server time: *Year/month/day/hour /minute/second*	14	20170126173655
wg	Web address of the Gateway provided by the server	5	103

3.3.2.2 New Sensor [Se]

A wireless sensor node is linked to a gateway, meaning a gateway can have N linked sensors. Nodes are in charge of supplying the measurement data from their sensors. In this case, up to 4 different sensors can be registered with a single request. It is worth noting here that the number of association payloads per HTTP request is variable, depending on the number of associated stations, but this will be always confined between 1 and 4.

- **[Se] HTTP REQUEST**

  The ***newSensor [Se]*** method registers a new Sensor in the data receiver server. It consists of a common part and a variable number of association payloads (from 1 to 4), whose value is defined by the new number (*n)* field.

Common part	Association payload STA #1	Association payload STA #2	Association payload STA #3	Association payload STA #4
46 bytes	45 bytes	45 bytes	45 bytes	45 bytes
226 bytes				

http://test.entomatic.upf.edu/Se?wg=⑤&n=②&ms1=⑯&la1=③.③&lo1=③.③&ms2=⑯&la2=③.③&lo2=③.③&ms3=⑯&la3=③.③&lo3=③.③&ms4=⑯&la4=③.③&lo4=③.③

*(*Numbers within circles correspond to the maximum number of bytes contained in the data field.)*

1. Common part format:
   http://test.entomatic.upf.edu/Se?wg=⑤&n=② ➔ 46 bytes

2. Association payload format:
   &ms1=⑯&la1=③.③&lo1=③.③ ➔ 45 bytes

Table 3.6: Proposed *Se* method request fields

Field	Purpose	Maximum size (bytes)	Example
wg	Web address of the Gateway	5	103
n	Number of sensor requests	2	3
ms1, ms2, ms3, ms4	MAC address of each sensor	16	00124b000615ab18
la1, la2, la3, la4	Latitude coordinate of each sensor	3+3 = 6	41.40
lo1, lo2, lo3, lo4	Longitude coordinate of each sensor	3+3 = 6	2.202

- **[Se] SERVER RESPONSE**

  The server responses, in turn, with the following data frame.

  OK 0|timestamp|n*|ws1|ws2|ws3|ws4|
  None OK 1|timestamp|

Table 3.7: Proposed *Se* method response fields

Field	Purpose	Maximum size (bytes)	Example
timestamp	Current server time	14	20170126173655

	with the format: *Year/month/day/hour /minute/second*		
n*	Number of accepted sensor association requests	2	3
ws1, ws2, ws3, ws4	Web address of each sensor provided by the server	5	103

If all or some stations could be appropriately registered in the platform, the server would response to the request by filling the n* field with the number of newly associated stations, and including in the ws fields attached next their allocated web address. For those that could not be associated, the value of their ws field would be 0.

Example 1:
If all stations were properly associated, the server would response with:

0|timestamp|4|ws1|ws2|ws3|ws4|

being *ws1*, ws2, ws3 and *ws4* the values of their allocated web address.

Example 2:
If the server received a request of 4 associations, but it only could give an address to the first three ones, this would be the response:

0|timestamp|3|ws1|ws2|ws3|0|

being *ws1*, *ws2*, and *ws3* the values of their allocated web address.

Example 3:
If not a single association could be handled by the server, it would response with:

1|timestamp|

3.3.2.3 New Measurement [Me]

Once a gateway and a sensor are signed up in the system, the WSN is ready for sending measurements data. The *measurement_type* table includes all the expected measurements to be found in the URL encoded http request via the *short_name* field. In this case, data from up to 3 different sensors can be accepted. It is worth noting here that the number of data payloads per HTTP request is variable, depending on the number of associated stations, but this will be always confined between 1 and 3.

- **[Me] HTTP REQUEST**

  The ***newMeasurement [Me]*** method introduces new information obtained from the sensor in the data receiver server. It consists of a common part and a variable number of data payloads (from 1 to 3), whose value is defined by the new number (*n)* field.

Common part	Data payload STA #1	Data payload STA #2	Data payload STA #3
46 bytes	65 bytes	65 bytes	65 bytes
241 bytes			

http://test.entomatic.upf.edu/Me?wg=⑤&n=②&ws1=⑤&co1=③&in1=③&fl1=③&te1=②.②&hu1=②&lu1=②&ba1=②&ws2=⑤&co2=③&in2=③&fl2=③&te2=②.②&hu2=②&lu2=②&ba2=②&ws3=⑤&co3=③&in3=③&fl3=③&te3=②.②&hu3=②&lu3=②&ba3=②

*(*Numbers within circles correspond to the maximum number of bytes contained in the data field.)*

1. Common part format:

http://test.entomatic.upf.edu/Me?wg=⑤&n=② ➔ 46 bytes

2. Data payload format:
&ws1=⑤&co1=③&in1=③&fl1=③&te1=②.②&hu1=②&lu1=②&ba1=② ➔ 65 bytes

Table 3.8: Proposed *Me* method request fields

Field	Purpose	Maximum size (bytes)	Example
wg	Web address of the Gateway	5	103
n	Number of data payloads from different sensors	2	3
ws1, ws2, ws3	Web address of the sensor provided by the server	5	213
co1, co2, co3	Message counter to keep control of possible failures	3	3
in1, in2, in3	Number of insects/events detected	3	10
fl1, fl2, fl3	Number of olive flies detected	3	2
te1, te2, te3	Temperature in °C	2+2 = 4	22.24
hu1, hu2, hu3	Humidity (%)	2	41
lu1, lu2, lu3	Luminance (%)	2	20
ba1, ba2, ba3	Battery level (%)	2	98

- **[Me] SERVER RESPONSE**

The server responses, in turn, with the following data frame.

OK 0|timestamp|n*|ws1|ws2|ws3|
None OK 1|timestamp|

Table 3.9: Proposed *Me* method response fields

Field	Purpose	Maximum size (bytes)	Example
timestamp	Current server time with the format: *Year/month/day/hour /minute/second*	14	20170126173655
n*	Number of data payloads properly received	2	3
ws1, ws2, ws3	Web address of each sensor provided by the server	5	103

Similarly to the proposed Se method, if the server receives data payload from only a subset of stations, it will response by filling the n* field with the number of received payloads. Data whose packets were lost are signalled with 0.

Example 1:
In case all packets were properly received, the response of the server would be:

0|timestamp|3|ws1|ws2|ws3|

being *ws1, ws2,* and *ws3* the values of the stations' web address, whose measurements have been properly received.

Example 2:
If the server received a transmission of 3 data payloads, but it only could properly decodify the first and the last one, this would be its response:

0|timestamp|2|ws1|0|ws3|

being *ws1* and *ws3* the values of the stations' web address, whose measurements have been properly received.

Example 3:
If not a single data transmission could be handled by the server, it would response with:

1|timestamp|

3.3.2.4 New Alarm [Al]

Alarms work in the same way as measurements, but with the tables Alarm and Alarm types.

- **[Al] HTTP REQUEST**

  The ***newAlarm [Al]*** method registers a new alarm in the data receiver server.

  http://test.entomatic.upf.edu/Al?ty=①&wg=⑤

  *(*Numbers within circles correspond to the maximum number of bytes contained in the data field.)*

Table 3.10: Proposed *Al* method request fields

Field	Purpose	Maximum size (bytes)	Example
ty	Type of alarm. Used to distinguish between different kind of alarms	1	2
wg	Web address of the Gateway	5	103

The type of alarm determines the value of the type ***(ty)*** field as well as the presence of possible additional fields. The alarms considered in the ENTOMATIC system are described in the following lines.

Table 3.11: Summary of possible alarms depending on the type (*ty*) field value

Value of type (*ty*) field	Purpose
1	Malfunctioning network
2	Malfunctioning device
3	Device running out of battery
4	High population of olive flies

a) Malfunctioning network
This alarm is set off if the gateway detects a high ratio of lost packets or instability in the association mechanism of stations. The web address of the gateway is also attached (*wg*).

http://test.entomatic.upf.edu/Al?ty=1&wg=⑤

*(*Numbers within circles correspond to the maximum number of bytes contained in the data field.)*

b) Malfunctioning device

This alarm is set off if the gateway detects that an associated device is sending values out of normal operation thresholds. After the type (*ty*) field, the web address of the gateway (*wg*) and the one of the malfunctioning device (*ws*) will be attached.

http://test.entomatic.upf.edu/AI?ty=2&wg=⑤&ws=⑤

*(*Numbers within circles correspond to the maximum number of bytes contained in the data field.)*

c) Device running out of battery

This alarm is set off if a device (one of the associated stations or the gateway itself) is running out of battery. Again, after the type (*ty*) field, the web address of the gateway (*wg*), the web address of the malfunctioning device (*wg* when being the gateway or *ws* when being an associated station) will be attached. Lastly, the battery level of the affected device will be also included.

http://test.entomatic.upf.edu/AI?ty=3&wg=⑤&wg/ws=⑤&ba=②

*(*Numbers within circles correspond to the maximum number of bytes contained in the data field.)*

d) High population of olive flies

Lastly, if the population of olive flies detected by a station during the sensing period exceeds a pre-established threshold, an alarm is set off. After the type (*ty*) field, three fields are attached: the web address of the gateway (*wg*), the web address of the corresponding station (*ws*) and the number of olive flies detected (*fl*).

http://test.entomatic.upf.edu/AI?ty=4&wg=⑤&ws=⑤&fl=②

*(*Numbers within circles correspond to the maximum number of bytes contained in the data field.)*

- **[AI] SERVER RESPONSE**

  In all alarm cases, the server responses, in turn, with the following data frame.

  OK 0|timestamp|
  No OK 1|timestamp|

Table 3.12: Proposed *AI* method response fields

Field	Purpose	Maximum size (bytes)	Example
timestamp	Current server time with the format: *Year/month/day/hour /minute/second*	14	20170126173655

3.4 INPUT AND OUTPUT

3.4.1 SERIAL COMMUNICATION WITH GPRS MODULE

The communication between the Zolertia Re-Mote and the GPRS module is performed through a serial connection defined as ***9600 8N1,*** with its main characteristics summarized in Table 3.13.

Table 3.13: Serial connection parameters between the Zolertia RE-Mote and the GPRS module

Serial connection parameters	Value
Baud rate	9600 bps [26]
Data bits	8
Parity bits	No parity
Synchronization bits	1

The physical connection between the GPRS module and the Zolertia RE-mote (by using the UART #1) consists of 4 wires: one for sending data (TX), one for receiving data (RX), the power supply (3.3V), and the ground connector (GND). A diagram of this connection can be found in Figure 3.10.

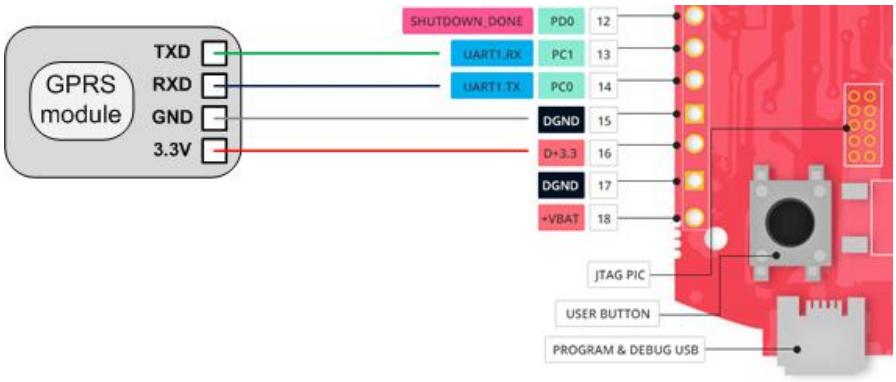


Figure 3.10: Physical connection between the GPRS module and the Zolertia RE-mote

[26] When SIM900 is powered on for the first time, it will be in auto baud mode. This feature makes the GSM module smarter, so that it can detect at which baud rate the microcontroller is sending the data.

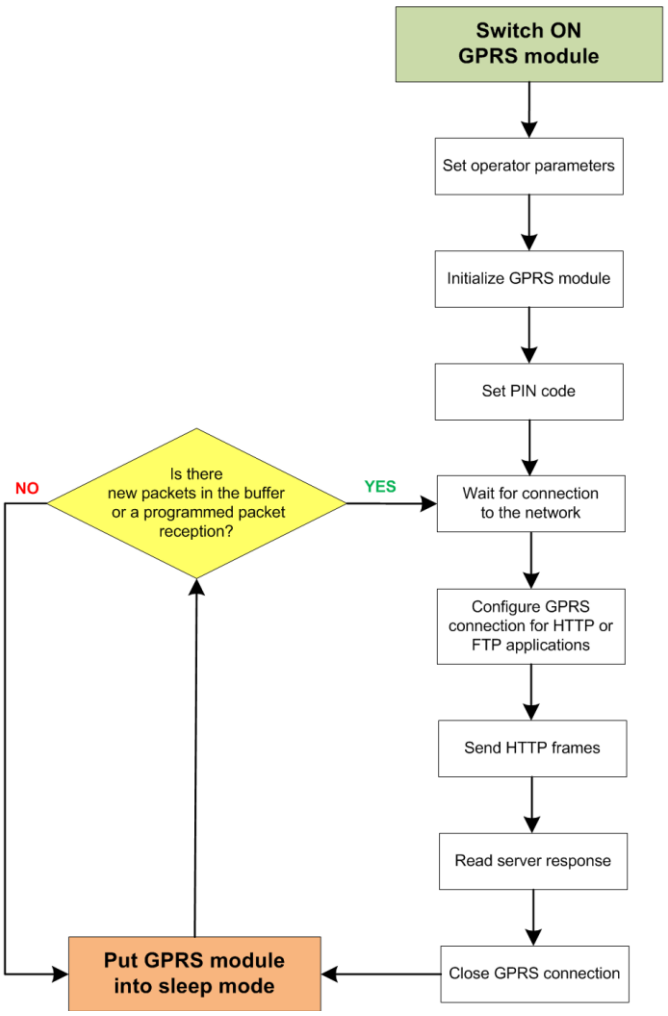


Figure 3.11: Diagram of operations performed by the GPRS module

The communication language between the Zolertia RE-Mote and the SIM900 GPRS module is based on the AT commands. These commands are instructions used to control a modem. AT is the abbreviation of Attention and every command line must start with "AT" or "at". Besides the common AT command set, GSM/GPRS modems and mobile phones support an AT command set that is specific to the GSM technology, which includes SMS-related commands as well as other commands to implement some protocols of the application layer such as HTTP or FTP.

The typical operation of the SIM900 GPRS module in the ENTOMATIC project is described in Figure 3.11 and all the tasks are performed thanks to different AT commands compiled in the SIM900 AT command manual [52] and executed from the Zolertia RE-Mote.

3.5 TRANSMISSION SCHEDULING MODEL

The scheduling of the data acquisition from the opto-electronic and other environmental sensors as well as its transmission to the data receiver server have taken into account two main considerations:

a) The communication system's goal of enlarging as much as possible batteries' lifetime.
b) The well-known routine of the olive fruit flies.

3.5.1 TRANSMISSION SCHEDULING IN THE WSN

As already known thanks to tests performed in Task 4.3 "Construction and testing of functional ENTOMATIC WSN", energy consumption of sensor nodes is directly proportional to the data transmission frequency. In addition, flies are only active during the daytime, so that it has no sense to take and transmit sensor measures at night.

Because of these main determinants, the proposed transmission scheduling model allows to define a predetermined ***active period*** of the system, limited by the boundaries ***MORNING*** and ***NIGHT***, so that data transmissions are only performed within this active period. In the rest of hours, the GW only transmits void beacons, which are used by STAs to keep stable their clock synchronization (see Figure 3.12).

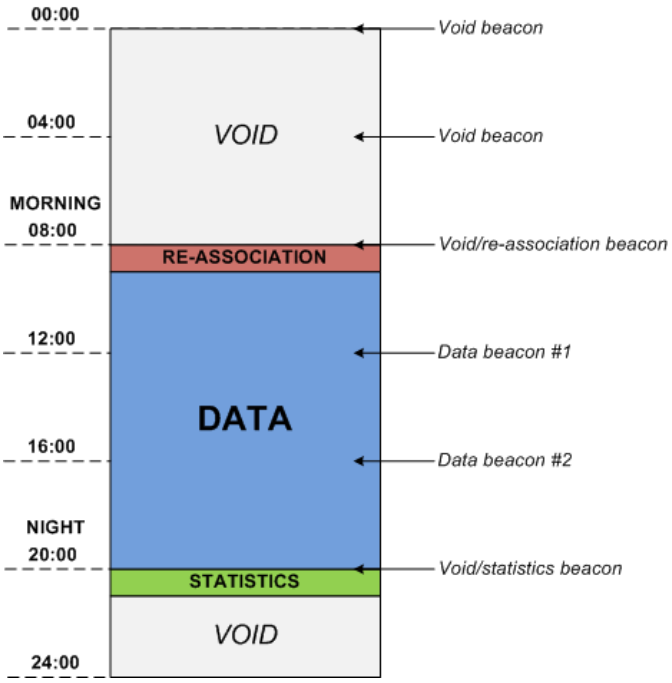


Figure 3.12: Example of transmission scheduling in the WSN based on beacons every 4 hours

As for the active period, it can contain a configurable number of daily measurements depending on the scenario requirements (see Table 3.14). Once selected the number of measurements in the GW, they are fairly distributed among the hours conforming the active period. Lastly, the periodicity of re-association and statistics beacons is also set in the gateway, so that one can choose the period (in days) between two consecutive beacons of each type. In those days when no re-association or statistics beacons are selected, the gap is occupied by a void beacon to keep the network's synchronization.

Table 3.14: Equivalence table between number of sensor measurements and beacon periodicity

Number of daily measurements	Beacon periodicity
2	4 hours
5	2 hours
11	1 hour
23	30 minutes
47	15 minutes
71	10 minutes
143	5 minutes

By way of illustration, let's suppose that we want to configure a network that is only active from 8:00 in the morning to 20:00 at night. Our application requires only 2 measurements per day, and we want to

re-associate the whole network every 4 days and get statistics of the network's performance (i.e., the transmission reliability of packets) every 2 days. Based on this configuration, we obtain the transmission scheduling from Figure 3.13, where the two daily measurements are taken at 12:00 and 16:00. The beacon from 8:00 is used every 4 days to re-associate the whole network, while the one from 20:00 is used every two days to get statistics. As for the rest of beacons, they are used for synchronization purposes.

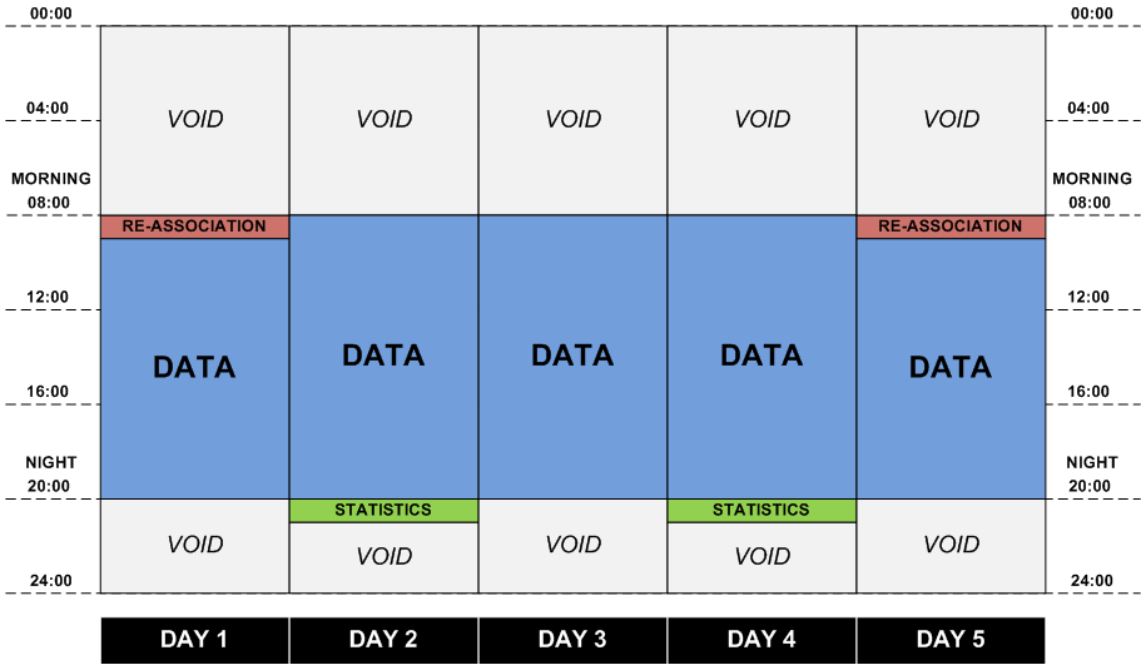


Figure 3.13: Example of transmission scheduling

Lastly, the gateway gets the clock time from the data receiver server every time a communication between these two devices is established. This value is kept in memory, so that the gateway executes every day a routine to fix its own drift with respect to the data receiver server. In turn, the gateway fixes its clock delay with respect to the associated STAs with the first beacon after the beginning of the MORNING boundary.

It is worth noting here that the re-association beacon is always sent in combination with the void beacon corresponding to the time slot determined by MORNING. While the void beacon is sent in the MORNING period, the re-association beacon is sent early in advance (generally 3 minutes earlier).

3.5.2 TRANSMISSION SCHEDULING WITH THE SERVER

The transmission scheduling of the GW's communication with the server is closely related to the transmission scheduling in the WSN. By this way, after performing all the actions indicated by the GW in its beacons, STAs can enter into a sleeping mode while the GW transmits all the gathered information to the data receiver server.

As it can be seen in Figure 3.14, after executing a re-association phase, the GW notifies the server of all new STAs associated to the network through the ***New Sensor [Se]*** method defined in susbsection 3.3.2.2.

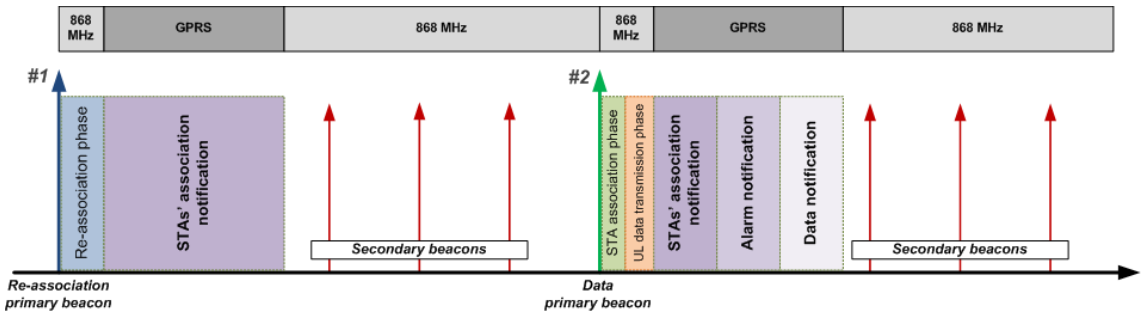


Figure 3.14: Transmission scheduling between the GW and the data receiver server

Similarly, after each data transmission phase, the GW notifies the server of all the possible new stations associated to the network. Then, the GW evaluates the gathered information from the STAs and, if necessary, activates one of the described alarms belonging to the ***New Alarm [Al]*** method from subsection 3.3.2.4. Lastly, the GW retransmits to the data receiver server all the sensor information by means of the ***New Measurement [Me]*** method from subsection 3.3.2.3.

In case the beacon was asking for statistics from the STAs, the GW would only communicate with the data receiver server by sending a ***Malfunctioning network alarm*** (see subsection 3.3.2.4) if a high ratio of lost packets or instability in the association mechanism of stations was detected.

As for the synchronization between the GW and the data receiver server, every downlink message coming from the server includes a timestamp which alleviates the possible clock drift between both devices. In addition, a synchronization routine is activated daily (in the last beacon before the MORNING time) to remove any possible accumulated drift. This routine is based on the daily transmission of the ***New Gateway [Ga]*** method (see subsection 3.3.2.1), whose response includes an updated timestamp.

It is worth noting here that, apart from the synchronization method aforementioned, void beacons do not imply any further communication between the GW and the server.

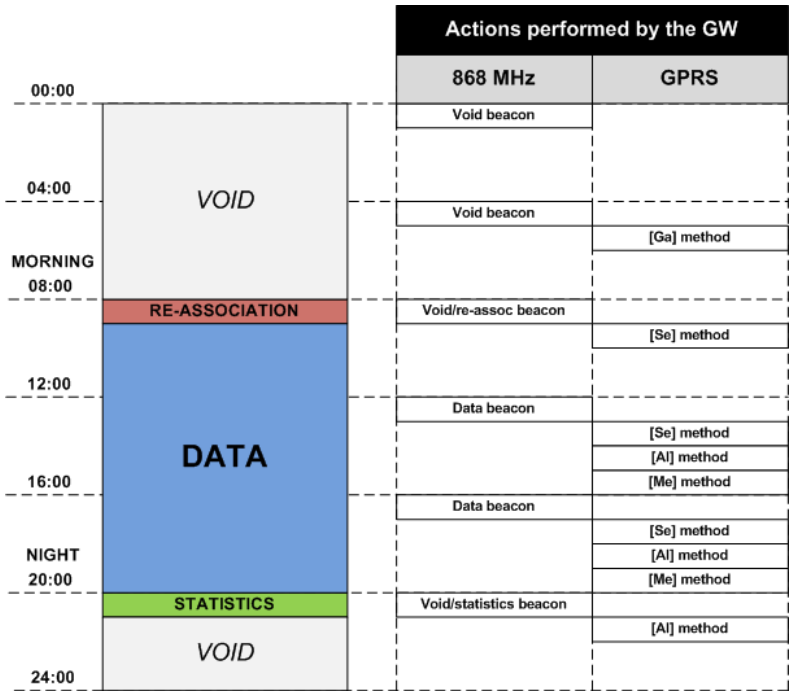


Figure 3.15: Actions performed by the GW in an example of transmission scheduling

By way of illustration, and taking as example the configuration proposed in subsection 3.5.1, Figure 3.15 shows the actions performed by the GW both in the WSN (see 868 MHz columns) and in its link with the data receiver server (see GPRS column).

3.5.3 SYSTEM'S INITIALIZATION

As the communication system consists of two different devices: the GW and the STAs, the following lines describe the steps that should be followed to complete an appropriate system's initialization:

1. Turn on the GW by connecting a battery and pressing the right button.
   a. Wait until the color from its led changes from red to blue. It means that the GW has been correctly registered into the database of the data receiver server. In addition, thanks to the response of the server, the time and date of the GW is set, so that it can start its scheduling system.
   b. Otherwise, if the red led keeps on after 2 minutes, reset the device by pressing the right button.
2. Press the left button to initialize the WSN. The led color will change from blue to green.
3. According to the network's configuration, the GW will wait until the next programmed slot to send the first re-association beacon and, immediately afterwards, the first general (void, data or statistics) beacon.
4. Since the GW has created the WSN (i.e., its led is green), any STA can be turned on by pressing its right button. Then, the *connection test* mechanism will be executed.
   a. If successful, the STA will discover a GW in its coverage range and its led will remain green for 5 seconds. The STA will enter in sleeping mode for a period determined by the response of the GW to the connection test request.
   b. If not successful, its led will remain red for 5 seconds and the device will be automatically switched off. Only by pressing again the right button, the STA will execute again the connection test mechanism.
5. Due to the elevate energy consumption of leds (~3 mA) and the battery constraints of the employed devices, no more state indicators have been programmed in STAs. Any possible failure regarding the network and/or device operation is indicated in the dashboard through the corresponding alarm.

3.6 PERFORMANCE TESTS

3.6.1 SYSTEM VALIDATION

Prior to the design and manufacture of the integrated board (Zolertia RE-Mote + GPRS SIM900) from ***3.2.2 Integrated board***, a first connection between these both devices was established through an intermediate board, as shown in Figure 3.16.

While being powered via an USB cable, the Zolertia RE-Mote makes use of one of its UART to power, in turn, the GPRS SIM900 and exchange data packets thanks to the TX and RX pins (Pin-out connection is shown in Figure 3.10).

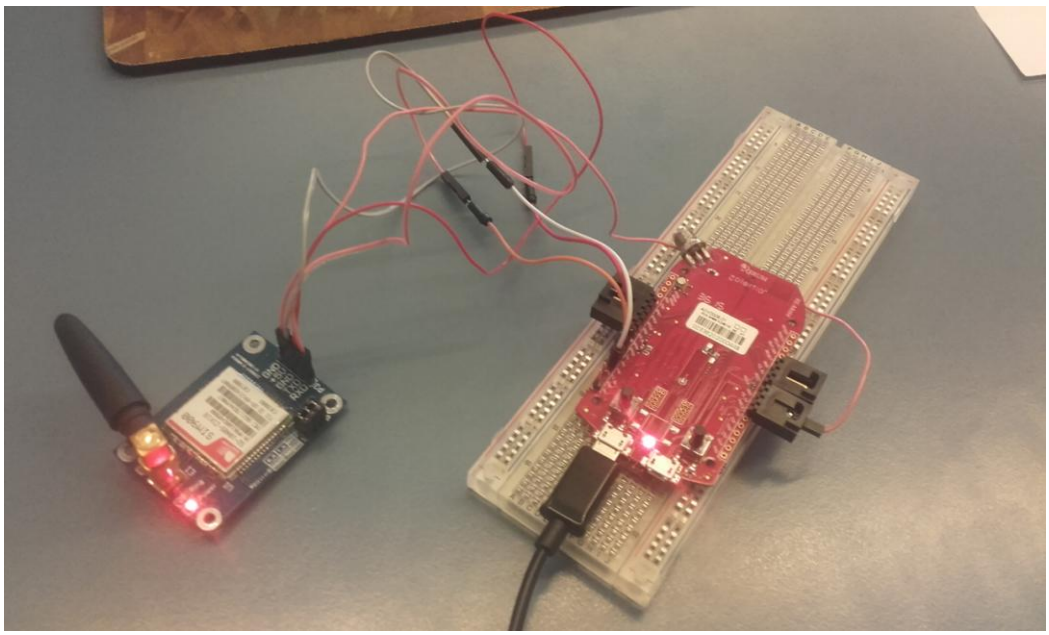


Figure 3.16: Connection between the Zolertia RE-Mote and the GPRS SIM900

To test the GPRS connection with the ENTOMATIC data receiver server under this configuration, the GPRS SIM900 was loaded with an SIM card, two fictitious nodes were configured in the ENTOMATIC dashboard (see Figure 3.17), and simulated data was programmed in the Zolertia RE-Mote, which acted as a gateway. The two fictitious nodes (with ID #207 and #208) were associated to a new network controlled by the gateway (consisted of a Zolertia RE-Mote device and a GPRS SIM900 connected through an intermediate board).

Different data frames were generated in the gateway with simulated data and sent to the data receiver server with successful results, as shown in Figure 3.18.

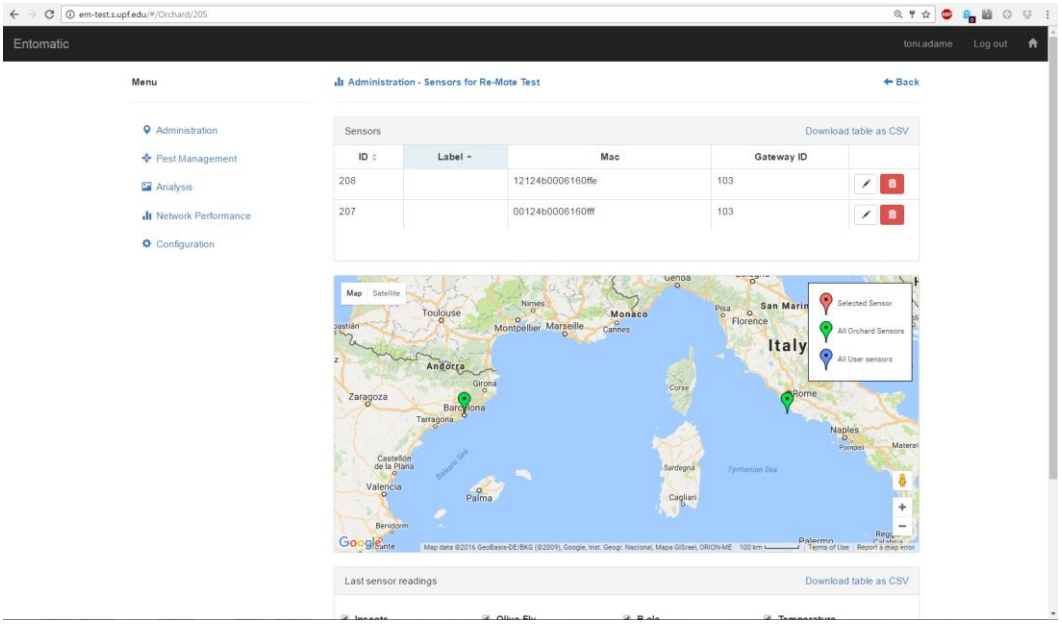


Figure 3.17: ENTOMATIC dashboard with the two fictitious nodes

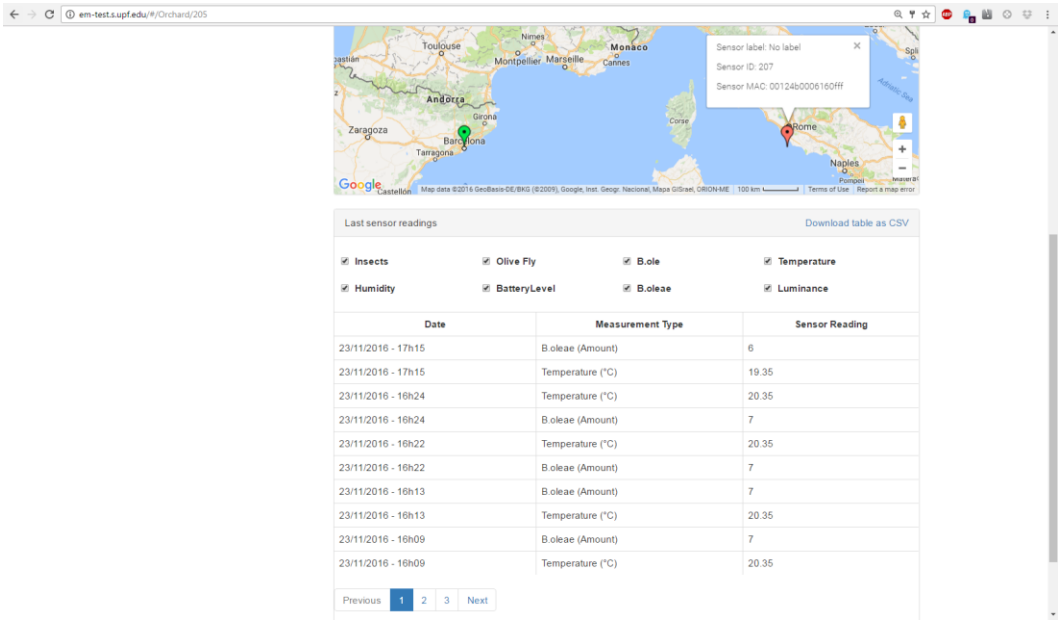


Figure 3.18: Data received by the ENTOMATIC server with information from sensor with ID #207

3.6.2 LABORATORY TESTBED

Performance evaluation of the whole communication system was performed in a testbed located in the 2nd floor, right wing of the Tanger building at UPF facilities[27]. The testbed consisted of 6 Zolertia RE-Mote nodes (one of them acting as a gateway and connected to a PC) running the ENTOMATIC protocol stack over IEEE 802.15.4.

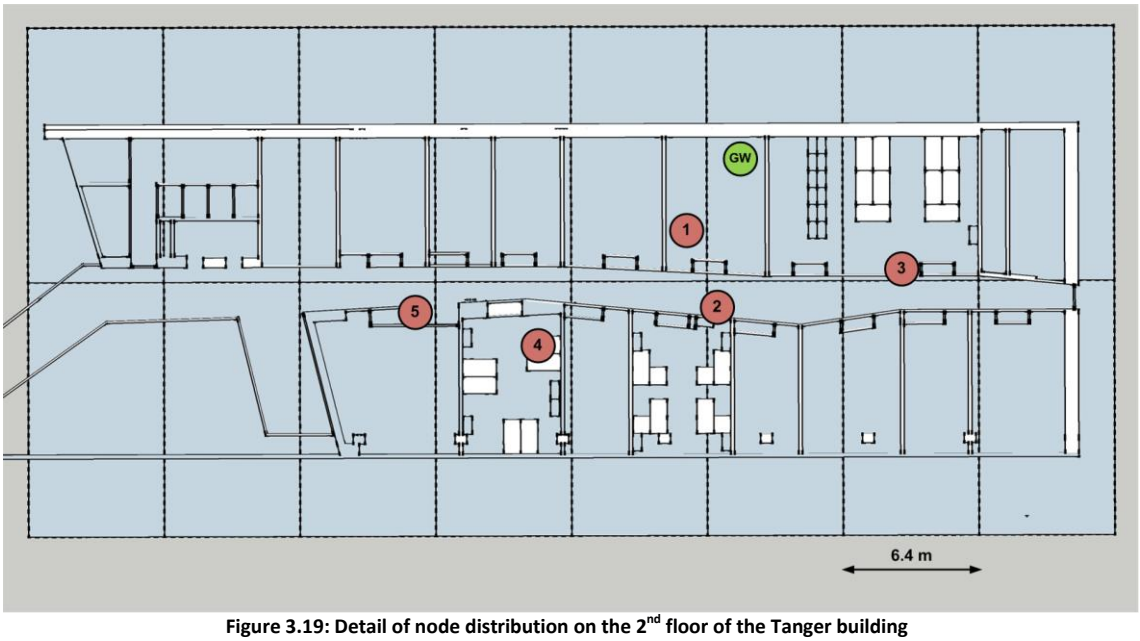


Figure 3.19: Detail of node distribution on the 2nd floor of the Tanger building

All tests were executed considering no mobility and with the same STAs' placement (see Figure 3.19). Detailed pictures of the STAs' placement can be observed from Figure 3.20 to Figure 3.25. All STAs were powered by an 800 mAh battery except the gateway, which was continuously powered by the PC. In addition, all STAs periodically took environmental measures through their connected sensors: a DHT22 temperature and humidity sensor, and a Groove light sensor v1.2.

[27] https://www.upf.edu/campus/en/comunicacio/tanger.html

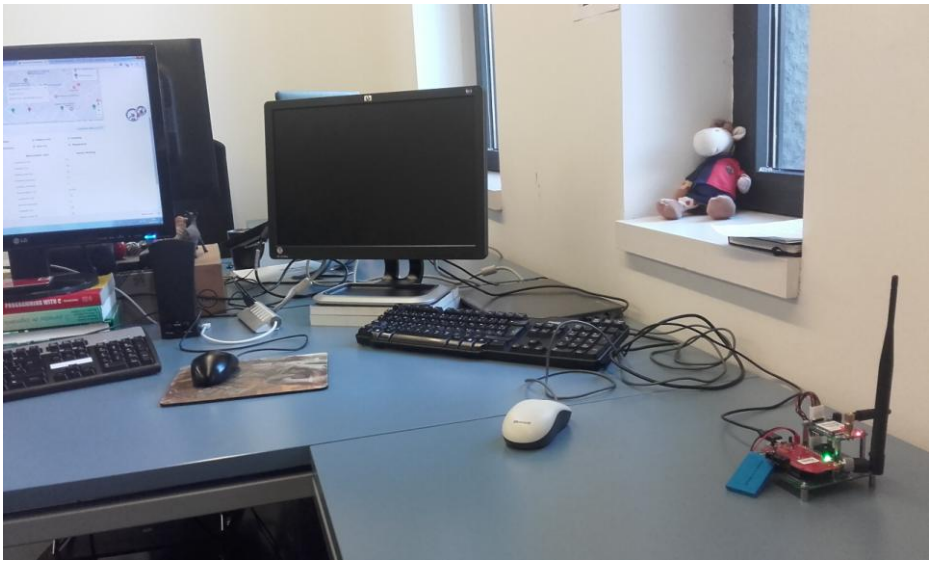


Figure 3.20: Placement of GW

Figure 3.21: Placement of STA #1

Figure 3.22: Placement of STA #2

Figure 3.23: Placement of STA #3

Figure 3.24: Placement of STA #4

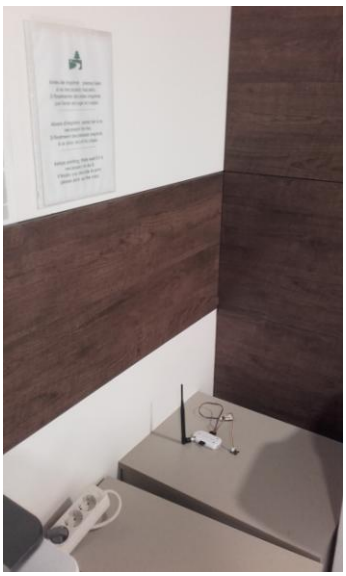


Figure 3.25: Placement of STA #5

Results and metrics of tests were directly obtained from the data published in the web dashboard. The own log record of the GW, based on its measures or thanks to the *statistics messages* periodically sent by STAs, has been also used. Statistics messages contain information about different metrics such as the number of packets sent and acknowledged, RTT delays, as well as power profiles of microprocessor and radio module. The test finished when all STAs ran out of battery; however, most of metrics shown in the current study only consider the period when all STAs remain active; so that the end of the observation time is set in the last statistic message containing data from every single STA. A summary of the main test parameters is provided in Table 3.15.

Table 3.15: Test configuration

Test configuration	Value
ENTOMATIC communication protocol version	v5.0
Platform	Zolertia RE-Mote
Operational frequency	868 MHz
Data rate	50 kbps
Packet size	≅ 50 bytes
Number of GW	1
Number of STAs collecting environmental data	5
Maximum distance between STAs	30 meters
Test start	March 14th at 13:15 hours
Test finish	March 21st at 20:10 hours
Observation start	March 14th at 13:15 hours
Observation finish (last statistic message with data from all STAs)	March 20th at 20:10 hours
Number of sensor measurements / day	71
Periodicity of sensor measurement	10 minutes
Daily beginning of active period	8:00 hours
Daily end of active period	20:00 hours
Re-association periodicity	Daily (at 8:00 hours)
Statistics periodicity	Daily (at 20:00 hours)

Under this configuration, Table 3.16 depicts the number and type of the beacons emitted by the GW during the test. Note that there are 466 data beacons (the ones which ask for sensor data) and 6 statistics beacons (the ones which gather information from each deployed device).

Considering that there are 5 STAs in the network producing data, this generates a load of $(466 + 6) \cdot 5 = 2.360$ packets expected by the GW. From these packets, only the ones corresponding to data will be retransmitted by the GW to the data receiver server; i.e., $466 \cdot 5 = 2.330$ data packets. As for the $6 \cdot 5 = 30$ statistics packets, they are only used for computing metrics in the GW and, if necessary, for activating an alarm in case the PDR received by the GW was lower than a certain quality threshold. In the current test, the minimum accepted PDR is 75%.

Table 3.16: Summary of beacons emitted by the GW

	OBSERVATION PERIOD							
Re-association beacons	**Day 1**	**Day 2**	**Day 3**	**Day 4**	**Day 5**	**Day 6**	**Day 7**	**Total**
Re-association	1	1	1	1	1	1	1	7
General beacons	**Day 1**	**Day 2**	**Day 3**	**Day 4**	**Day 5**	**Day 6**	**Day 7**	**Total**
Void	24	72	72	72	72	72	49	433
Data	40	71	71	71	71	71	71	466
Statistics	0	1	1	1	1	1	1	6
TOTAL	64	144	144	144	144	144	121	905

3.6.2.1 Association mechanism

The association mechanism of the WSN has been tested every single day of the simulation, so that the whole network is daily reconfigured at 8:00 hours. The different set of topologies is shown from Figure 3.26 to Figure 3.29. As expected, nodes closer to the GW such as STA #1, STA #2 and STA #3 are mainly located in lower rings (1 or 2), while nodes far away (STA #4 and STA #5) are allocated into higher rings (2 or 3). In addition, it is worth noting here that, regardless the network topology, STA #5 is always connected to STA #2, and STA #4 is always connected to STA #1.

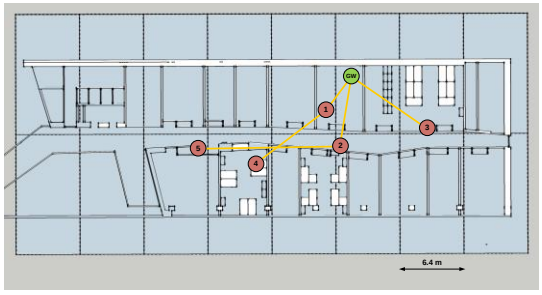


Figure 3.26: Network topology A

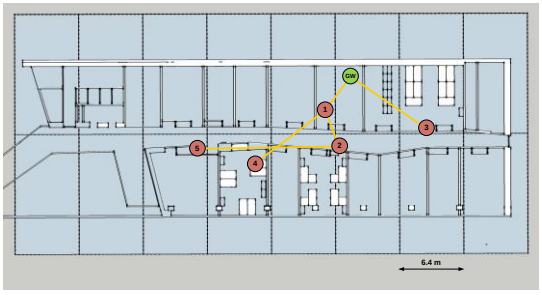


Figure 3.27: Network topology B

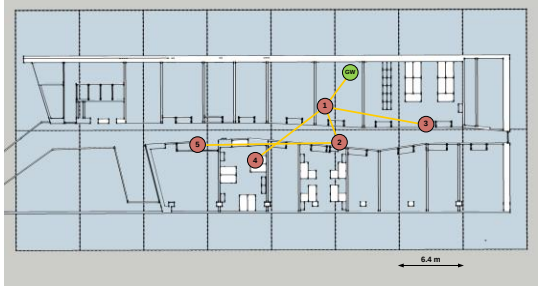


Figure 3.28: Network topology C

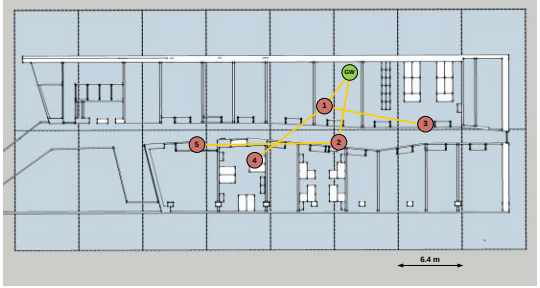


Figure 3.29: Network topology D

The frequency appearance of each topology is depicted in Table 3.17, with predominance of topologies C and D, where STA #1 is directly connected to the GW, STA #3 and STA #4 are connected to STA #1, STA #5 is connected to STA #2, and STA #2 is either connected to STA #1 or directly connected to the GW.

Table 3.17: Frequency appearance of each network topology

Network topology	Number of general beacons with this topology	Number of general beacons (%) with this topology
A	112	12.38%
B	144	15.91%
C	288	31.82%
D	361	39.89%
TOTAL	905	100%

A summary of topology metrics for the connected STAs is shown in Table 3.18 and Table 3.19 (in %), where the most important conclusion is that all STAs are always connected to the network, regardless the topology.

Table 3.18: Topology metrics for connected STAs

Stations	Beacons Connected	Number of beacons in ring			Number of beacons with children				Number of beacons with next hop					
		1	2	3	0	1	2	3	GW	STA #1	STA #2	STA #3	STA #4	STA #5
STA #1	905	905	0	0	0	112	505	288	905	0	0	0	0	0
STA #2	905	473	432	0	0	905	0	0	473	432	0	0	0	0
STA #3	905	256	649	0	905	0	0	0	256	649	0	0	0	0
STA #4	905	0	905	0	905	0	0	0	0	905	0	0	0	0
STA #5	905	0	473	432	905	0	0	0	0	0	905	0	0	0

Table 3.19: Topology metrics for connected STAs (in %)

Stations	Beacons Connected	% of beacons in ring			% of beacons with children				% of beacon with next hop					
		1	2	3	0	1	2	3	GW	STA #1	STA #2	STA #3	STA #4	STA #5
STA #1	100%	100%	0%	0%	0%	12.38 %	55.80 %	31.82 %	100%	0%	0%	0%	0%	0%
STA #2	100%	52.27 %	47.73 %	0%	0%	100%	0%	0%	52.27 %	47.73 %	0%	0%	0%	0%
STA #3	100%	28.29 %	71.71 %	0%	100%	0%	0%	0%	28.29 %	71.71 %	0%	0%	0%	0%
STA #4	100%	0%	100%	0%	100%	0%	0%	0%	0%	100%	0%	0%	0%	0%
STA #5	100%	0%	52.27 %	47.73 %	100%	0%	0%	0%	0%	0%	100%	0%	0%	0%

3.6.2.2 WSN reliability

The WSN reliability has been measured in the GW after taking into consideration its own gathered information as well as the information provided by STAs in their ***statistics packets*** (whose results are attached in Table 3.20). The first computed metric is the PDR (Packet Delivery Ratio), which is measured as the quotient between the number of received packets at the GW and the number of expected packets according to the associated STAs.

Table 3.20: Statistics table at the end of the test

	Packets sent	Packets acknowledged	ACKs sent	RTT link (ms.)	RTT e2e (s.)	CPU	LPM	TX	RX	PW max	PW min
STA #1	474	472	1048	445	3	0.1%	99%	0.001%	4%	30	25

STA #2	475	469	472	371	3	0.1%	99%	0.001%	4%	29	26
STA #3	472	458	0	357	8	0.1%	99%	0.001%	4%	30	21
STA #4	473	440	0	386	8	0.1%	99%	0.001%	4%	30	19
STA #5	473	472	0	270	8	0.1%	99%	0.001%	4%	30	24
Total	2367	2311	1520	-	-	-	-	-	-	-	-
Average	473.4	462.2	304	365.8	6	0.1%	99%	0.001%	4%	29.8	23

Each network's STA has 5 consecutive transmission windows to perform its own data transmission, so that if its data packet is not acknowledged in the corresponding e2e-ACK at the end of a transmission window, it can repeat the process up to 5 times. Figure 3.30 show the accumulated PDR in the GW of all the expected transmissions (2.360 if we have into consideration data and statistics packets) in function of the transmission window. Only in the first transmission window 99.62% of packets were properly received by the GW; in the second window the whole amount was acknowledged.

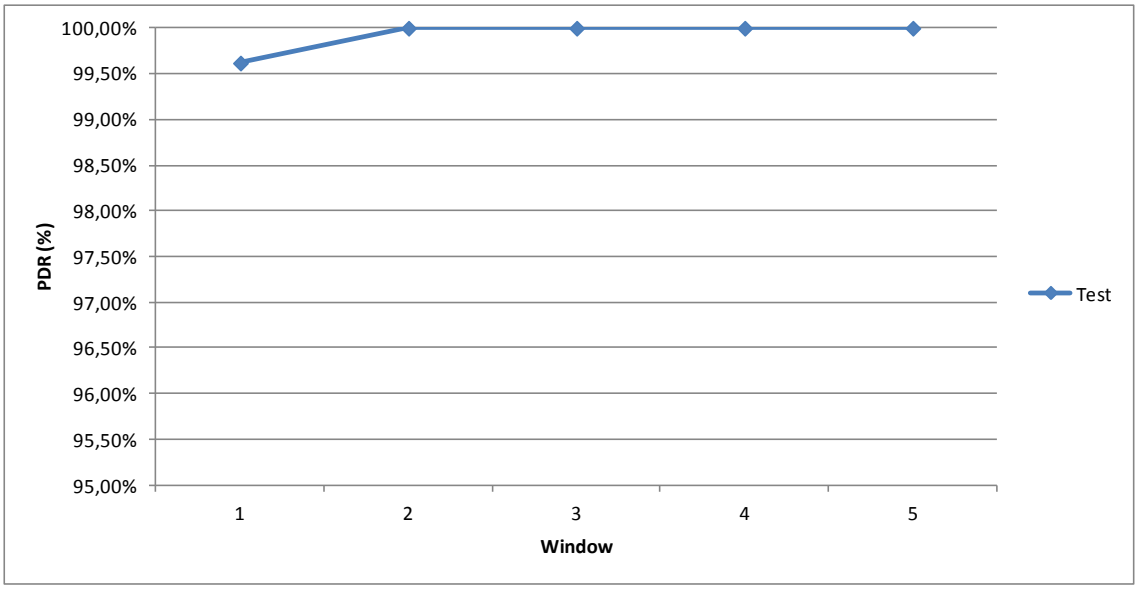


Figure 3.30: Accumulated Packet Delivery Ratio (PDR) after each transmission window

Table 3.21 depicts the main metrics related to the transmission reliability in the WSN during the executed test. The proposed system shows its suitability by providing 100% of success both in the CSR and the PDR metrics. The number of transmissions in the network per packet received at the GW, with a value of 1.003, reflects the good quality within the network's links, as STAs do not need to retransmit their own packets and they are almost always acknowledged at the end of the first transmission window.

Another mechanism that proves its suitability in this environment is the e2eACK, resulting in a reduction of the number of retransmissions. This can be observed, for instance, in the number of packets of STA #4 acknowledged by its immediate parent (see Table 3.20), so that a low value of this metric (440 packets directly acknowledged) does not directly imply the lost of the packet, but rather the lost of the ACK, and the subsequent acknowledgement of the packet in the corresponding e2eACK.

Lastly, the association process shows 2 rings on average as mean number of rings in the network, without any disassociation during the whole test and a mean association delay of 93.14 seconds. This value has been strongly affected by the re-association process of the last day, where STA #3 could not enter in the network at first attempt and it should wait until the first general beacon to get associated (i.e., it should wait more than 10 minutes to get associated).

Table 3.21: WSN reliability metrics

Metric	Value

Total number of duty cycles	472
Total number of duty cycles fully acknowledged	472
CSR (Cycle Stability Ratio)	100%
Total number of packets received at the GW	2360
Total number of packets expected at the GW	2360
PDR (Packet Delivery Ratio) after 5 windows	100%
Mean number of transmissions in the network per packet received at the GW	1.003
Mean number of ACKs per STA and cycle	0.64
Mean number of rings	2
Total number of STAs disassociations	0
Mean association delay	93.14 seconds

3.6.2.3 Link GW-Server reliability

The measure of the reliability in communications between the GW and the data receiver server has been quantified by computing the PDR of the link. In this case, as the test consisted of 466 data beacons and 5 STAs, the data receiver server should have received $466 \cdot 5 = 2330$ data packets.

Table 3.22: GW-Server reliability metrics

Metric	Value
Total number of data cycles	466
Total number of data cycles fully acknowledged	463
CSR (Cycle Stability Ratio)	99.36%
Total number of packets received at the server	2330
Total number of packets expected at the server	2323
PDR (Packet Delivery Ratio)	99.69%
Mean re-association time (consisting of 2 HTTP frames)	73.14 seconds
Mean data transmission time (consisting of 2 HTTP frames)	74.03 seconds

As described in subsection 3.3.2.3, frames sent from the GW to the receiver server consist of up to 3 data payloads from different STAs, so that in the current deployment with 5 STAs, it is necessary to send 2 frames to the server per cycle: the first one with 3 data payloads, and the second one with 2 data payloads. After comparing the records from both the GW and the server, only 5 data payloads were lost during the whole test, resulting in an accumulated PDR = 99.69%. As for the CSR, the ratio of data cycles fully received, it is equally very high (99.36%).

Further analysis of gateway logs shows that the fourth day of the test (March 17th), two frames from the GW to the server were lost: the first one at 17:20 containing 2 data payloads corresponding to STAs #4 and #5, and the second one at 17:30 containing 3 data payloads corresponding to STAs #1, #2 and #3. On March 20th, another frame from the GW to the server was lost at 14:20 containing 2 data payloads corresponding to STAs #3 and #5

As for the transmission time between the GW and the data receiver server, to inform the server about a new network topology (i.e., after each new re-association period) lasts, on average, 73.14 seconds which correspond to 2 HTTP frames. Similarly, the transmission of a whole cycle of STAs' payloads represents 2 HTTP frames lasting 74.03 seconds on average.

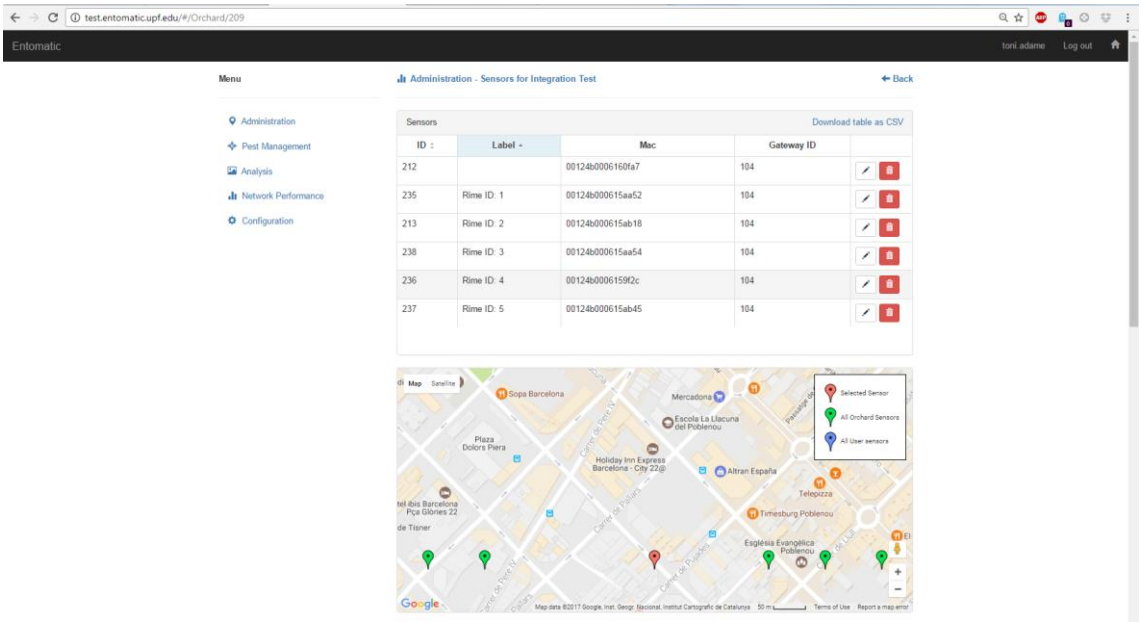


Figure 3.31: Capture of the test deployment in the data receiver server
(note that position of STAs in map is illustrative to easily differentiate one device from another)

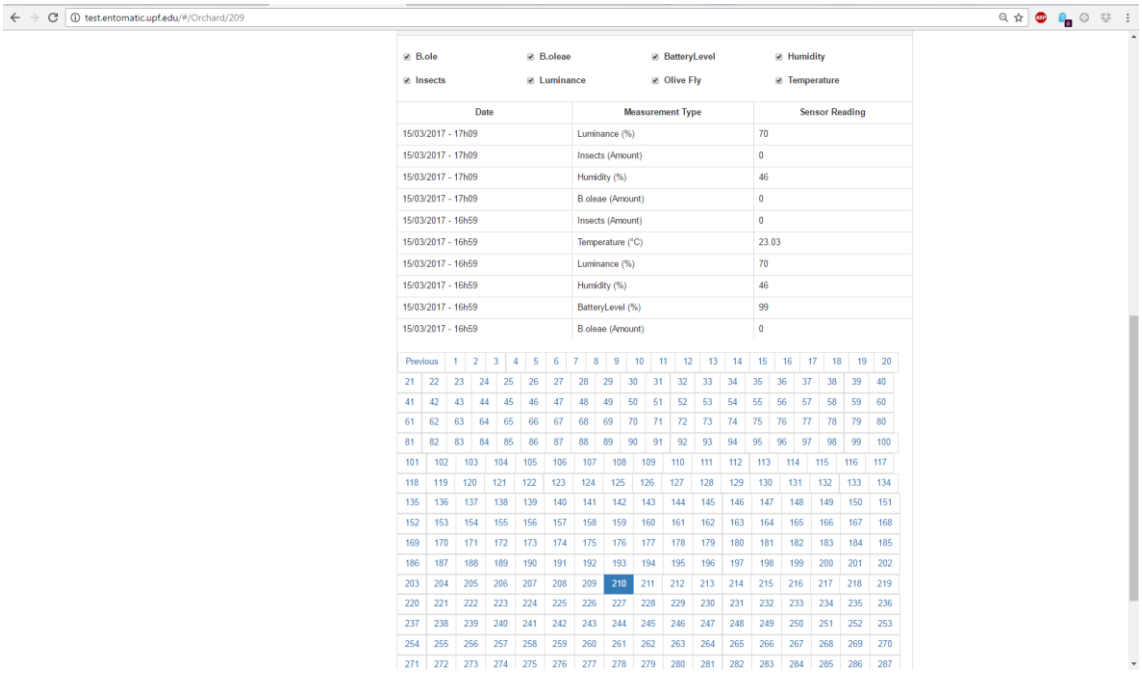


Figure 3.32: Capture of data obtained in the data receiver server during the test

The access to the platform is available in the following webpage:

http://test.entomatic.upf.edu

by using the following credentials:

Username: **toni.adame**
Password: **entomatic**

The log of the reported measurements is available by clicking:

1. Administration
2. Organisations -> National
3. Users -> Toni Adame Vázquez
4. Orchards -> Integration Test

5. View Sensors
6. And then clicking on the different STAs

3.6.2.4 Alarms

As described in subsection 3.3.2.4, four different alarms have been defined in the system with their corresponding thresholds depicted in Table 3.23. During the test, only the alarm related to the low-battery of an STA has been set off. However, the rest of alarms have been tested in other internal simulations, where the corresponding values have been explicitly altered in order to prove the alarm usage.

Table 3.23: Alarm thresholds

Alarm	Threshold	Value
Malfunctioning network	Minimum value of PDR (%) before setting off an alarm	75%
Malfunctioning device	Maximum temperature allowed before setting off an alarm	60
	Minimum temperature allowed before setting off an alarm	0
	Maximum humidity allowed before setting off an alarm	99
	Minimum humidity allowed before setting off an alarm	0
	Maximum luminance allowed before setting off an alarm	99
	Minimum luminance allowed before setting off an alarm	0
	Maximum number of detected olive flies before setting off an alarm of malfunctioning sensor	99
	Minimum number of detected olive flies before setting off an alarm of malfunctioning sensor	0
Device running out of battery	Minimum battery level allowed before setting off an alarm	95%
High population of olive flies	Maximum number of detected olive flies before setting off an alarm of high population	25 flies

As stated above, the single cause for activating an alarm in the current test has been a STA running out of battery. Table 3.24 compiles the metrics regarding the activated alarms, being 100% of them produced as consequence of a battery level under 95%. The value of this threshold was established after modeling the abrupt battery level drop when the Zolertia RE-Mote is about to be exhausted (see Figure 2.91 and Figure 2.92, and also the battery level behavior in the current test in Figure 3.34).

To sum up, 72 alarms were generated during the test, all of them caused by the low level of batteries. After receiving these alarms from STAs, the GW was able to retransmit all this information to the data receiver server without any loss (see Table 3.24). As for the distribution of these alarms (see Table 3.25 and Figure 3.33), STAs #2 and #3 together represent more than 60% of the total amount of alarms.

Table 3.24: Alarm metrics

Metric	Value
Generated alarms	72
Low-battery alarms	72 (100%)
Received alarms by the server	72
PDR (Packet Delivery Ratio)	100%

Table 3.25: Distribution of low-battery alarms among STAs

Station	Number of low-battery alarms	Number of low-battery alarms (%)
STA #1	13	18.06%
STA #2	21	29.17%
STA #3	23	31.94%
STA #4	5	6.94%
STA #5	10	13.89%
TOTAL	72	100%

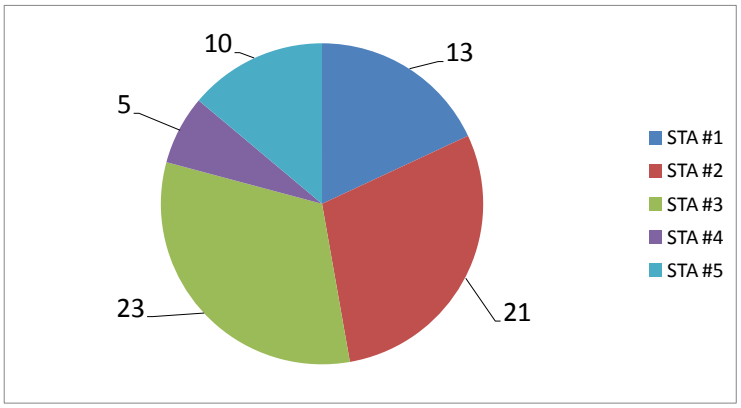


Figure 3.33: Low-battery alarm distribution among the different STAs

3.6.2.5 Battery lifetime

Battery lifetime of the STAs considered in the test has been measured as the time difference between the last message received by each STA at the server and the beginning of the test. From the results shown in Table 3.26, we can observe that almost all STAs have been operative for more than 7 consecutive days.

Table 3.26: STAs' battery lifetime

Stations	Time of test start	Time of the last message received at the server	Battery lifetime	Battery lifetime (minutes)
STA #1	14th March at 13:15 hours	21st March at 16:49 hours	7 days, 3 hours and 34 minutes	10294 minutes
STA #2	14th March at 13:15 hours	21st March at 13:49 hours	7 days and 34 minutes	10114 minutes
STA #3	14th March at 13:15 hours	21st March at 13:09 hours	6 days, 23 hours and 54 minutes	10074 minutes
STA #4	14th March at 13:15 hours	21st March at 19:59 hours	7 days, 6 hours and 44 minutes	10484 minutes
STA #5	14th March at 13:15 hours	21st March at 17:39 hours	7 days, 4 hours and 24 minutes	10344 minutes
Average	14th March at 13:15 hours	21st March at 16:17 hours	7 days, 3 hours and 2 minutes	10262 minutes

According to the results obtained in the test aimed to validate the communication protocols of the WSN (see Table 2.53), the battery lifetime of STAs was 5.29 days when setting a period between beacons of 3 minutes. Although the current test enlarges this period up to 10 minutes, the use by STAs of two sensors (and their corresponding current consumption) and the lower density of deployed nodes (making them to use higher power transmission levels) results in a reduction of the battery lifetime.

Lastly, the battery level chart from Figure 3.34 shows the high stability of this value (always between 98% and 99%) for all STAs, except from STA #4, whose battery level fluctuates between the two mentioned values, due to its own discharge process. It is worth noting here that the battery discharge process is not

lineal, and previous studies on this issue (see subsection 2.10.1.2) show how fast drops this value when the battery is about to get exhausted.

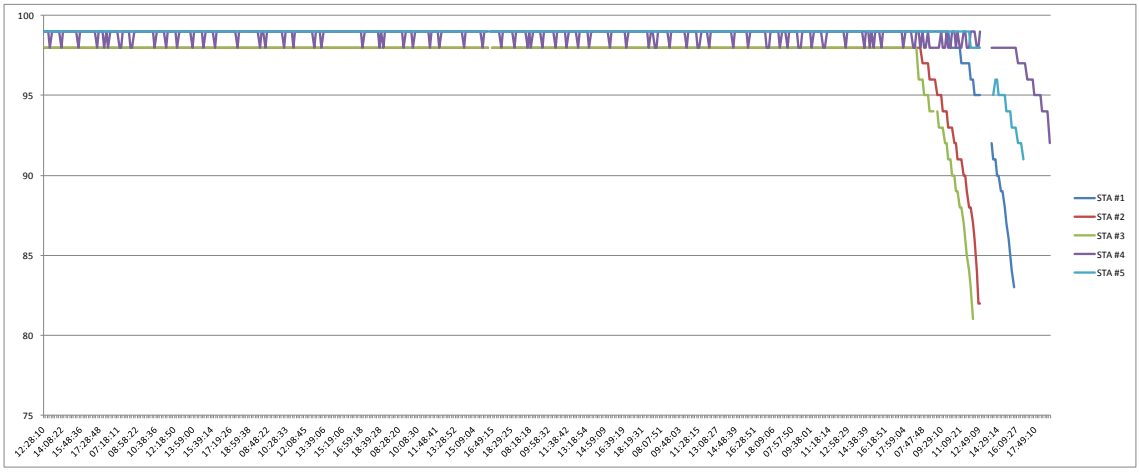


Figure 3.34: Battery level chart

3.6.2.6 Energy consumption

The model employed to estimate the energy consumption of STAs is described in subsection 2.4.3. This model split the consumption in the Zolertia RE-Mote into two main areas: first, the consumption in the TI CC2538 microcontroller; and second, the consumption in the TI CC1200 RF transceiver. Table 3.27 shows the time distribution of each operational state for the STAs involved in the test and its corresponding current consumption.

Table 3.27: Time distribution among the different states

Source	Mode	% of time	Current consumption
TI CC2538 Microcontroller	**CPU**	0.1%	$I_{CPU} = 13\ mA$
	LPM (Low Power Mode)	99%	$I_{LPM} = 0.4\ \mu A$
TI CC1200 RF Transceiver	**TX (Transmitting)**	0.001%	$I_{TX} = 39 - 61\ mA$
	RX (Receiving)	4%	$I_{RX} = 19\ mA$
	SL (Sleeping)	96%	$I_{SL} = 0.12\ \mu A$

According to the current consumption values from Table 3.27 and the following equation, we can obtain the main theoretical parameters regarding the current consumption depicted in Table 3.28.

$$E = t_{CPU} \cdot V_{DD} \cdot I_{CPU} + t_{LPM} \cdot V_{DD} \cdot I_{LPM} + t_{TX} \cdot V_{DD} \cdot I_{TX} + t_{RX} \cdot V_{DD} \cdot I_{RX} + t_{SL} \cdot V_{DD} \cdot I_{SL} = \\ = V_{DD} \cdot (t_{CPU} \cdot I_{CPU} + t_{LPM} \cdot I_{LPM} + t_{TX} \cdot I_{TX} + t_{RX} \cdot I_{RX} + t_{SL} \cdot I_{SL})$$

Table 3.28: Current consumption metrics

Metric	Value
Total energy consumption (E)	489.75 mAh
Mean current consumption	2.86 mA
Battery lifetime in hours for an 800 mAh battery	279.33 hours
Battery lifetime in days for an 800 mAh battery	11.64 days
Energy consumed per bit of payload delivered	172.76 mJ/bit

It is worth noting here that the deviation between the estimated battery lifetime (11.64 days) and the actual battery lifetime, on average, for the STAs in the test (7.13 days) is produced by external factors which can affect battery life[28] as well as by the use of environmental sensors. An α deviation factor is then defined and obtained as follows:

$$\alpha = \frac{7.13\ days}{11.64\ days} = 0.61$$

As a consequence, and now having into consideration all the elements of the system (microprocessor, transceiver, sensors and circuitry), more accurate battery lifetime estimations on different network settings can be computed by using this deviation factor α. In this case, we can update the battery lifetime estimated values from Table 2.53 with the aforementioned α deviation factor and obtain the following values (see Table 3.29):

Table 3.29: Update of lifetime of an 800 mAh battery

		Estimated battery lifetime (days)		Estimated battery lifetime with α deviation factor (days)	
		$T_p = 1\ h.$	$T_p = 4\ h.$	$T_p = 1\ h.$	$T_p = 4\ h.$
X-MAC Multi-hop	$E_{0/0}$	105.17	413.16	64.15	252.03
	$E_{10/5}$	101.04	397.21	61.63	242.30
	$E_{20/10}$	90.10	354.85	54.96	216.46
	$E_{30/15}$	87.38	344.31	53.30	210.03

From these results it can be concluded that for a feasible configuration of STAs in the ENTOMATIC main use case, with the GW sending beacons every 4 hours, the devices would be operative for 252.03 days (i.e., 8.4 months), far more than the stated requirement of 6 months of uninterrupted and unattended operation.

3.6.2.7 Sensor measurements

The STAs deployed in the described test sent every 10 minutes (within the configured active period from 8:00 in the morning to 20:00 at night) a data payload with 3 environmental measurements (temperature, humidity and luminance) and their current battery level (in %). Note that the 'events' and the 'flies' frame is filled with '0' due to the unavailability of traps at this stage of the project.

Type	Application packets byte index																			
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Data packet	*rime*		*seq*	*events*	*flies*	*temp*		*hum*	*light*	*bat*										

Figure 3.35: Data packet frame sent by STAs

By way of illustration, in the following lines are depicted the different charts corresponding to the measures sent by the 5 network STA's during one whole day of the simulation (the 15th March 2017). It is worth noting here that the horizontal axis of charts is represented in GMT (Greenwich Mean Time).

Data consistency of measured temperature can be observed in Figure 3.36, where STAs located in closed offices (STA #1 and STA #4) report higher values than the rest, due to the human presence. No significant changes are observed as a result of the permanent use of thermostats in the building, so that temperatures are always comprised between 21 °C and 25 °C.

[28] Digi-Key Electronics – Battery Life Calculator - http://www.digikey.es/en/resources/conversion-calculators/conversion-calculator-battery-life

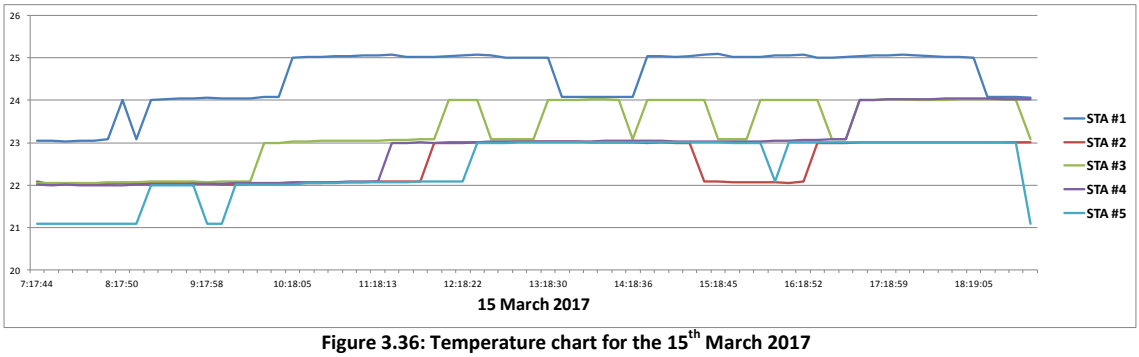


Figure 3.36: Temperature chart for the 15th March 2017

As for the humidity values from Figure 3.37, they are comprised between 34% and 48%, being their variations more similar in STAs #2, #3, and #4.

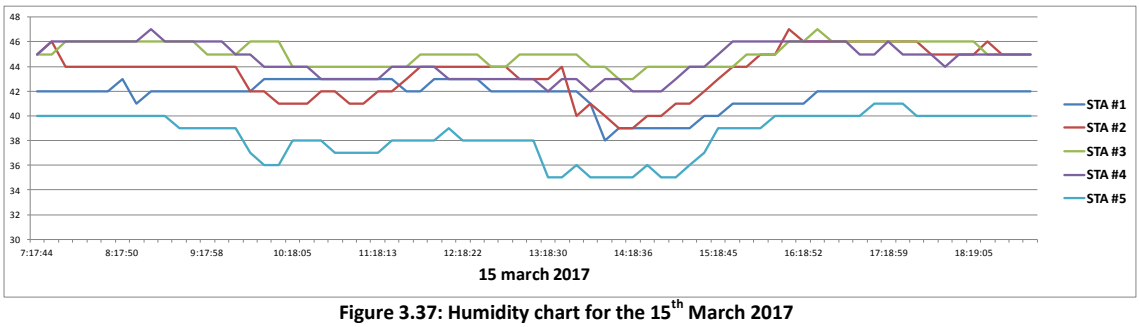


Figure 3.37: Humidity chart for the 15th March 2017

From the luminance chart (see Figure 3.38) more interesting conclusions can be extracted. For instance, the increase of this value during the first morning hours in occupied offices (see STA #1 and STA #4) when lights are for first time switched on and its corresponding descent in the afternoon, where lights are switched off.

STAs #2, #3 and #5, placed in a corridor, do not experience these variations, except from 10:30 GMT to 12:30 GMT, when maintenance tasks in the lightning system were conducted. Their actual values during that day are closely related to its placement; for instance, STA #2 is the best illuminated (40%-50%) due to its position under a light bulb of the corridor (see Figure 3.22). On the other hand, STAs #3 and #5 are rather hidden and do not have a direct light source in the surroundings.

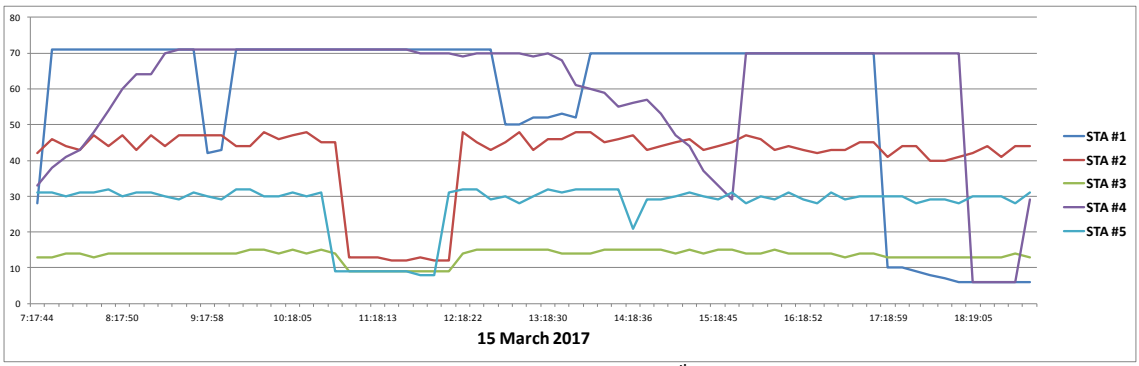


Figure 3.38: Luminance chart for the 15th March 2017

4 TEST EVALUATION SUMMARY

The following section compiles the main results obtained from the different tests performed with the Zolertia RE-Mote platform during the execution of Work Package 4 (WP4) "Design & Development of WSN".

4.1 WSN COMMUNICATION

A full description of this test is provided in subsection 2.11.2.2.

4.1.1 RELIABILITY

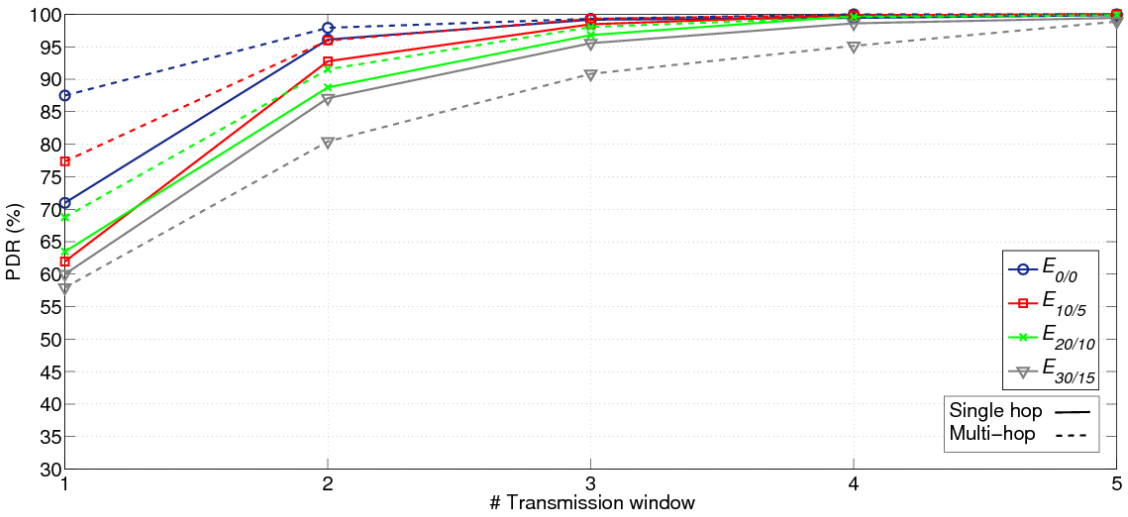


Figure 4.1: Packet delivery ratio (PDR) in the proposed testbed for X-MAC layer

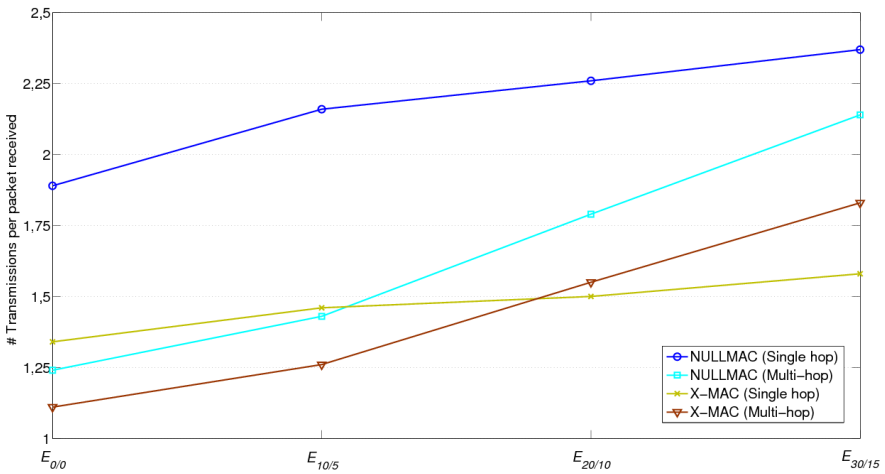


Figure 4.2: Number of transmissions per packet received in the proposed testbed

4.1.2 ENERGY CONSUMPTION

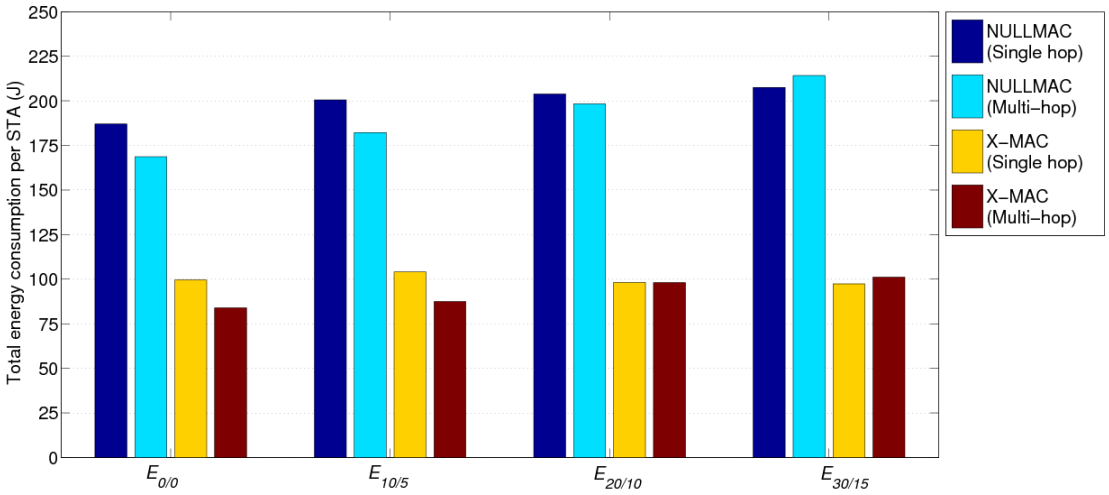


Figure 4.3: Average total energy consumption per STA after 20 beacons when $T_p = 3\ min.$

Table 4.1: Lifetime of an 800 mAh battery running the described testbed

			Battery lifetime (days)		
			$T_p = 3\ min.$	$T_p = 1\ h.$	$T_p = 4\ h.$
NULLMAC	Single hop	$E_{0/0}$	2.37	47.35	187.87
		$E_{10/5}$	2.21	44.19	175.41
		$E_{20/10}$	2.18	43.46	172.53
		$E_{30/15}$	2.14	42.70	169.53
	Multi-hop	$E_{0/0}$	2.63	52.51	208.17
		$E_{10/5}$	2.44	48.60	192.79
		$E_{20/10}$	2.24	44.66	177.27
		$E_{30/15}$	2.07	41.35	164.23
X-MAC	Single hop	$E_{0/0}$	4.46	88.75	349.64
		$E_{10/5}$	4.27	85.00	335.07
		$E_{20/10}$	4.52	89.99	354.46
		$E_{30/15}$	4.56	90.79	357.55
	Multi-hop	$E_{0/0}$	5.29	105.17	413.16
		$E_{10/5}$	5.08	101.04	397.21
		$E_{20/10}$	4.53	90.10	354.85
		$E_{30/15}$	4.39	87.38	344.31

4.1.3 RESILIENCE AGAINST FAILURES

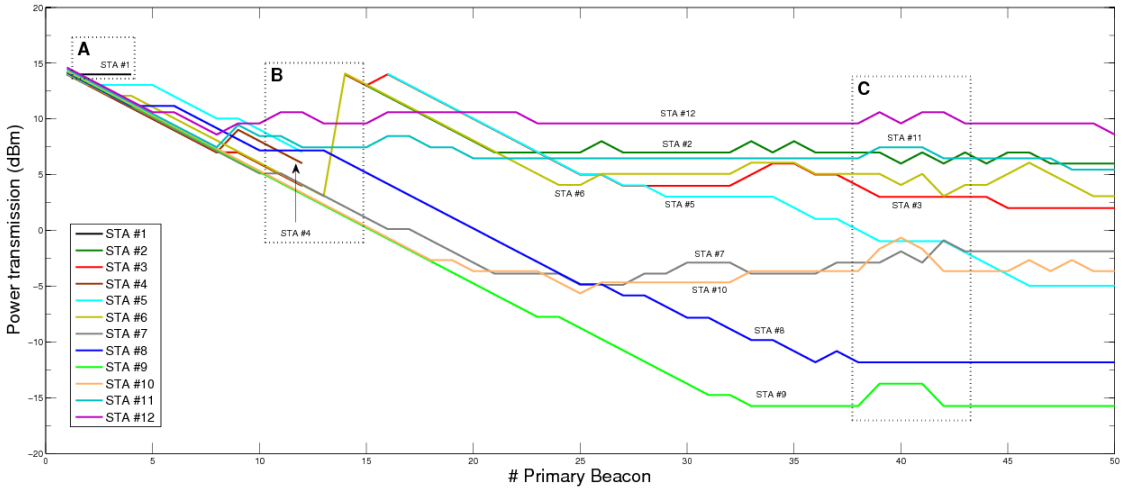


Figure 4.4: Temporal evolution of STAs' power transmission level

4.1.4 RANGE COVERAGE

A full description of this test is provided in subsection 2.11.3.

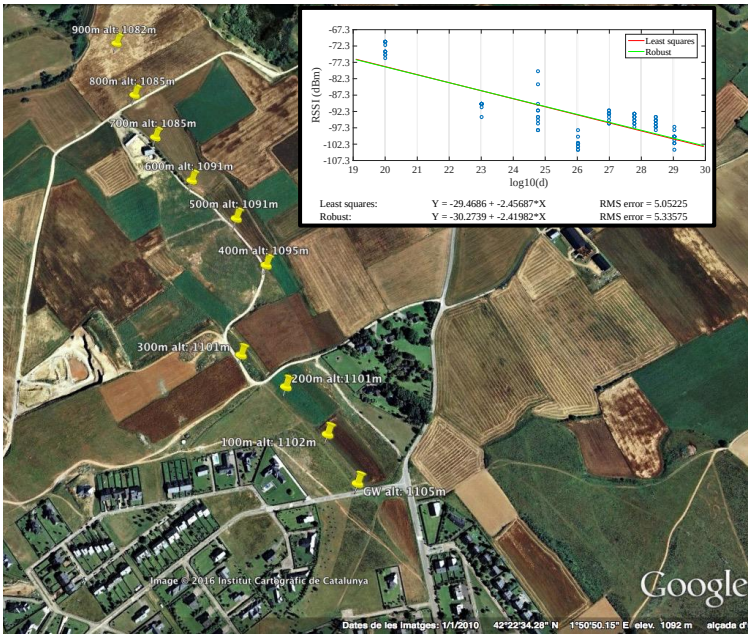


Figure 4.5: Test performed in La Cerdanya

4.2 WSN + GPRS COMMUNICATION

A full description of this test is provided in subsection 3.6.2.

4.2.1 RELIABILITY

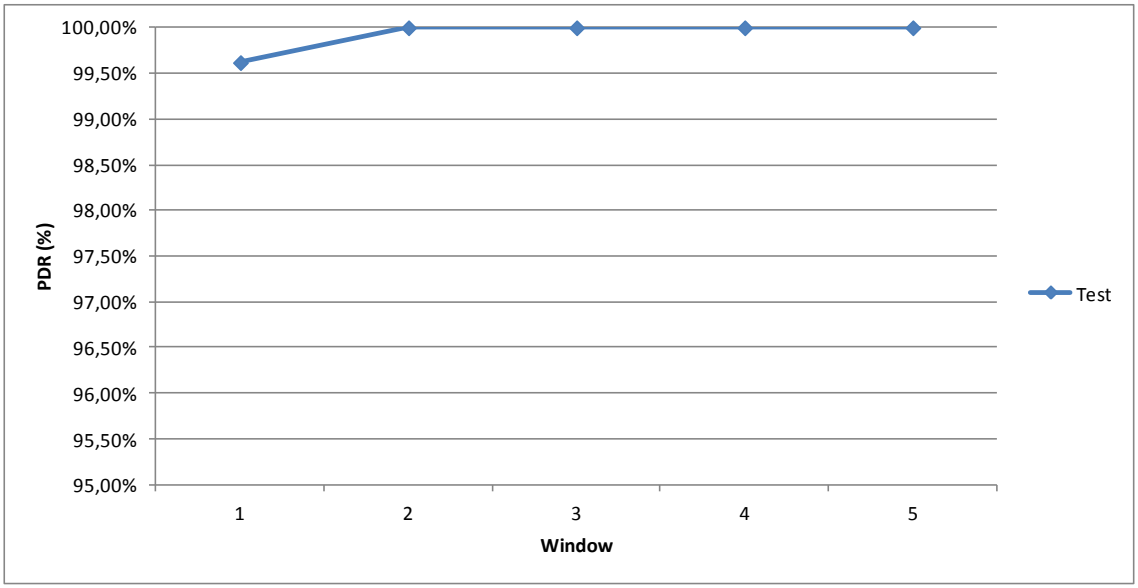


Figure 4.6: Accumulated Packet Delivery Ratio (PDR) after each transmission window

Table 4.2: GW-Server reliability metrics

Metric	Value
Total number of data cycles	466
Total number of data cycles fully acknowledged	463
CSR (Cycle Stability Ratio)	99.36%
Total number of packets received at the server	2330
Total number of packets expected at the server	2323
PDR (Packet Delivery Ratio)	99.69%
Mean re-association time (consisting of 2 HTTP frames)	73.14 seconds
Mean data transmission time (consisting of 2 HTTP frames)	74.03 seconds

4.2.2 ALARMS

Table 4.3: Alarm metrics

Metric	Value
Generated alarms	72
Low-battery alarms	72 (100%)
Received alarms by the server	72
PDR (Packet Delivery Ratio)	100%

4.2.3 BATTERY LIFETIME

Table 4.4: STAs' battery lifetime

Stations	Time of test start	Time of the last message received at the server	Battery lifetime	Battery lifetime (minutes)
STA #1	14th March at 13:15 hours	21st March at 16:49 hours	7 days, 3 hours and 34 minutes	10294 minutes
STA #2	14th March at 13:15 hours	21st March at 13:49 hours	7 days and 34 minutes	10114 minutes
STA #3	14th March at 13:15 hours	21st March at 13:09 hours	6 days, 23 hours and 54 minutes	10074 minutes
STA #4	14th March at 13:15 hours	21st March at 19:59 hours	7 days, 6 hours and 44 minutes	10484 minutes
STA #5	14th March at 13:15 hours	21st March at 17:39 hours	7 days, 4 hours and 24 minutes	10344 minutes
Average	14th March at 13:15 hours	21st March at 16:17 hours	7 days, 3 hours and 2 minutes	10262 minutes

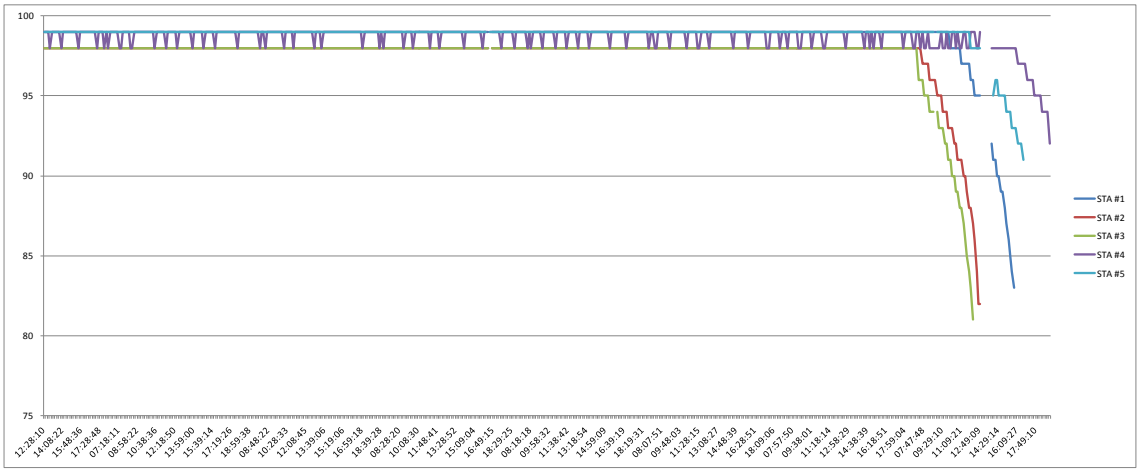


Figure 4.7: Battery level chart

4.2.4 ENERGY CONSUMPTION

Table 4.5: Update of lifetime of an 800 mAh battery

		Estimated battery lifetime (days)		Estimated battery lifetime with α deviation factor (days)	
		$T_p = 1\ h.$	$T_p = 4\ h.$	$T_p = 1\ h.$	$T_p = 4\ h.$
X-MAC Multi-hop	$E_{0/0}$	105.17	413.16	64.15	252.03
	$E_{10/5}$	101.04	397.21	61.63	242.30
	$E_{20/10}$	90.10	354.85	54.96	216.46
	$E_{30/15}$	87.38	344.31	53.30	210.03

4.2.5 SENSOR MEASUREMENTS

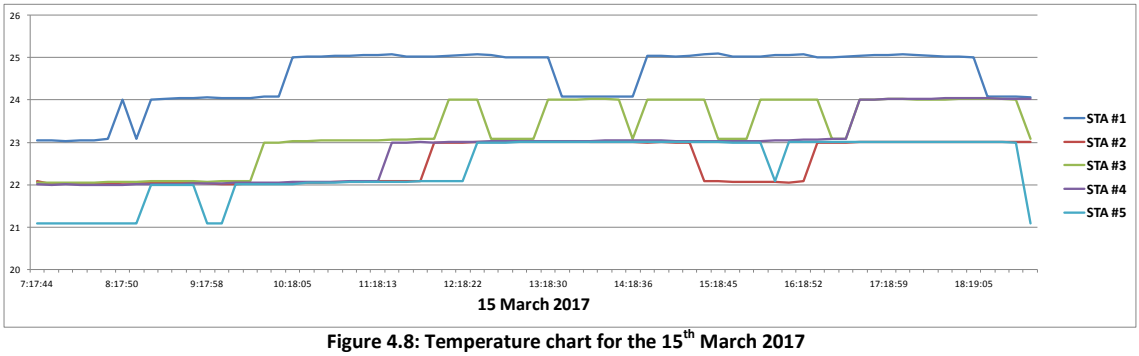


Figure 4.8: Temperature chart for the 15th March 2017

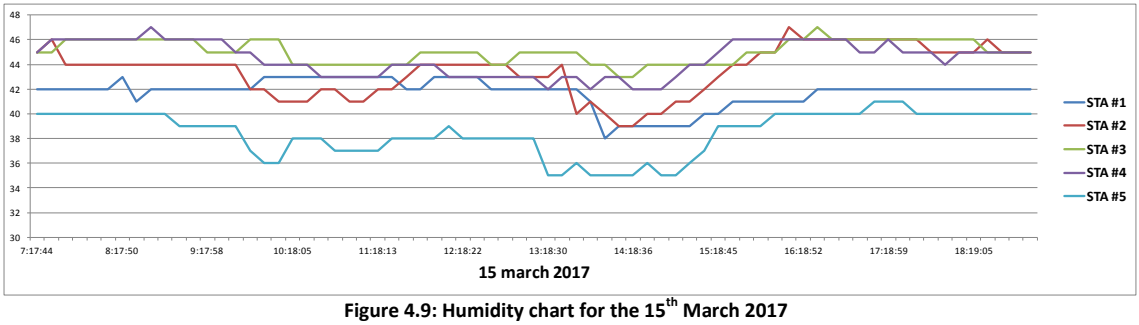


Figure 4.9: Humidity chart for the 15th March 2017

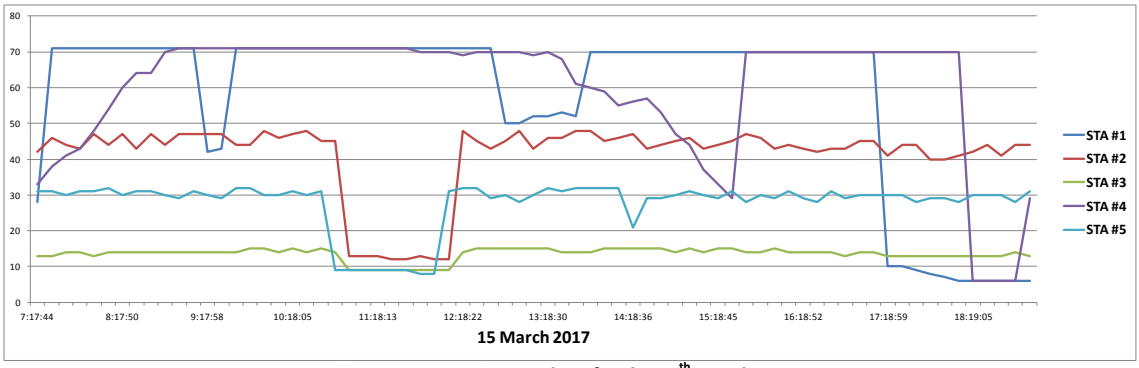


Figure 4.10: Luminance chart for the 15th March 2017

5 CONCLUSIONS

This document presented a series of mechanisms aiming to ensure ENTOMATIC communications' reliability in a self-managed environment (consisting of a WSN and a GPRS link) while reducing energy consumption as much as possible.

Single-hop topologies have become the de facto system to transmit data in current LPWANs mainly due to the need for network simplicity and robustness, and the fear of consuming too much energy in processing tasks and/or maintaining complex routing mechanisms. However, the ENTOMATIC communication protocol presented proves the suitability of alternative uplink multi-hop communication approaches in LPWANs. Distributed among three OSI layers (MAC, network and transport), the multiple mechanisms contained in the current work ensure network reliability and resilience against failures in uplink transmissions while keeping low energy consumption.

Results from a real WSN testbed show uplink PDR values above 95% for all considered settings, with faster achievement of this level when using multi-hop topologies with multiple transmission windows. In addition, multi-hop topologies outperform single-hop ones in terms of energy consumption in the considered non error-prone scenarios, with up to 15% improvement (which could be even much higher in larger networks) and values as low as 50 mJ/bit when employing an underlying duty-cycled MAC layer.

Similarly, network auto-configuration and resilience have been successfully put to the test after forcing the shutdown of some network STAs. No direct intervention is therefore required after switching off both the gateway and the deployed wireless modules, unless a massive error in the WSN occurs; still the network would be able to detect it and notify the user by sending a special alarm to the data receiver server via GPRS.

The results obtained from an end-to-end deployment (i.e., the WSN and the GPRS link between the GW and the data receiver server) are equally suitable for the ENTOMATIC application scenario. After placing 5 unattended STAs performing environmental monitoring, the accumulative PDR within the WSN was 99.62% after the first transmission window and 100% after the second one. As for the GPRS link, communication was also reliable, with a PDR of 99.69%.

The new transmission schedule system was validated, so that the network can be configured to send data only during the predetermined time periods, thus avoiding to send data in non-targeted periods; for instance, at night, when olive fruit flies sleep and cannot be detected by traps.

The operation of the temperature and humidity (DHT22) and luminance (Groove light) sensor was also validated after placing some of them in different locations within a dynamic environment such as a building. Similarly, the alarms produced by deviations in the information obtained by these sensors have been also tested, proving the usefulness of the proposed thresholds. Other alarms, such as the corresponding to a low reliability level of the network or a low battery level of devices, have been also validated in the test environment.

Lastly, the effort in enlarging battery lifetime of deployed wireless modules allows to achieve values of unattended operation in a feasible ENTOMATIC application scenario (i.e., with nodes sending data every 4 hours) from 210.03 days in an error-prone channel to 252.03 days in an ideal channel, thus fulfilling the initial stated requirement of at least 6 months of unattended operation. At this point it is worth noting that the optoelectronic sensor counts with its own battery, so that it has no impact on the estimated operating time of the communication system.

6 REFERENCES

[1] M. A. E. Villegas, S. Yee Tang and Y. Qian, “Wireless Sensor Network Communication Architecture for Wide-Area Large Scale Soil Moisture Estimation and Wetlands Monitoring,” University of Puerto Rico at Mayaguez (WALSAIP research project).

[2] J. Labs, “Background Perspective on Wireless Micro-networking,” [Online]. Available: http://www.jlhlabs.com/LowPowerWireless.htm.

[3] RERUM consortium, "RERUM (REliable, Resilient and secUre IoT for sMart city applications) main webpage," 2016. [Online]. Available: https://ict-rerum.eu/.

[4] A. Phani and K. A. Janakiram D., "Operating systems for wireless sensor networks: A survey technical report," Indian Institute of Technology Madras, Adi Mallikarjuna Reddy V AVU Phani Kumar, D Janakiram, and G Ashok Kumar, 2007.

[5] M. O. Farooq and T. Kunz, "Operating systems for wireless sensor networks: A survey," *Sensors,* vol. 11(6), pp. 5900-5930, 2011.

[6] A. Dunkels, B. Grönvall and T. Voigt, "Contiki-a lightweight and flexible operating system for tiny networked sensors," in *29th Annual IEEE International Conference on Local Computer Network*, 2004.

[7] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne and T. Voigt, "Cross-level sensor network simulation with cooja," in *IEEE conference on Local computer networks*, 2006.

[8] F. Österlind, "A sensor network simulator for the Contiki OS," SICS Research Report, 2006.

[9] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne and T. Voigt, "Cross-level sensor network simulation with Cooja," in *IEEE conference on Local computer networks*, 2006.

[10] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo and D. Gay, "TinyOS: An operating system for sensor networks," in *Ambient intelligence*, Springer Berlin Heidelberg, 2005, pp. 115-148.

[11] P. Levis and N. Lee, "Tossim: A simulator for tinyos networks," UC Berkeley, 2003.

[12] R. Goyette, "An analysis and description of the inner workings of the freertos kerne," Carleton University 5, 2007.

[13] E. Baccelli, O. Hahm, M. Günes, M. Wählisch and T. C. Schmidt, "RIOT OS: Towards an OS for the Internet of Things," in *IEEE Conference on Computer Communications*, 2013.

[14] R. Lajara, J. Pelegrí-Sebastiá and J. J. Perez Solano, "Power consumption analysis of operating systems for wireless sensor networks," *Sensors,* vol. 10, no. 6, pp. 5809-5826, 2010.

[15] A. Dunkels, J. Eriksson, N. Finne and N. Tsiftes, "Powertrace: Network-level power profiling for low-power wireless networks," Swedish Institute of Computer Science, 2011.

[16] T. Instruments, "CC2538 Powerful Wireless Microcontroller System-On-Chip for 2.4-GHz IEEE 802.15.4," 2015.

[17] I. C. Society, "Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)," The Institute of Electrical and Electronics Engineers, Inc, New York, 2003.

[18] P. Movva, "Application Report: AN125 - CC2538 + CC1200 Evaluation Module," 2013.

[19] A. Hazmi, J. Rinne and M. Valkama, "Feasibility Study of IEEE 802.11ah Radio Technology for loT and M2M use Cases," in *2012 IEEE Globecom Workshops*, 2012.

[20] A. Goldsmith, Wireless communications, Cambridge university press, 2005.

[21] A. Bachir, M. Dohler, T. Watteyne and K. K. Leung, "MAC essentials for wireless sensor networks," *IEEE Communications Surveys & Tutorials,* vol. 12, no. 2, pp. 222-248, 2010.

[22] A. Bachir, M. Dohler, T. Watteyne and K. K. Leun, "MAC essentials for wireless sensor networks," *IEEE Communications Surveys & Tutorials,* vol. 12, no. 2, pp. 222-248, 2010.

[23] C. Cano, B. Bellalta, A. Sfairopoulou and M. Oliver, "Low energy operation in WSNs: A survey of preamble sampling MAC protocols," *Computer Networks,* vol. 55, no. 15, pp. 3351-3363, 2011.

[24] K. Arisha, M. Youssef and M. Younis, "Energy-aware TDMA-based MAC for sensor network," in *System-level power optimization for wireless multimedia communication*, Springer US, 2002, pp. 21-40.

[25] S. C. Ergen and P. Varaiya, "PEDAMACS: Power efficient and delay aware medium access protocol for sensor networks," *IEEE Transactions on Mobile Computing,* vol. 5, no. 7, pp. 920-930, 2006.

[26] M. Ringwald and K. Romer, "BitMAC: A deterministic, collision-free, and robust MAC protocol for sensor networks," in *Proceeedings of the IEEE Second European Workshop on Wireless Sensor Networks*, 2005.

[27] M. I. Brownfield, K. Mehrjoo, A. S. Fayez and N. J. Davis IV, "Wireless sensor network energy-adaptive MAC protocol," Wireless Sensor Network Energy-Adaptive MAC Protocol, 2006.

[28] M. Buettner, G. V. Yee, E. Anderson and R. Han, "X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks," in *4th international conference on Embedded networked sensor systems*, 2006.

[29] A. Dunkels, "The contikimac radio duty cycling protocol," 2011.

[30] C. A. Boano, T. Voigt, N. Tsiftes, L. Mottola, K. Römer and M. A. Zúñiga, "Making sensornet MAC protocols robust against interference," in *Wireless sensor networks*, Springer Berlin Heidelberg, 2010, pp. 272-288.

[31] A. Keshavarzian, H. Lee and L. Venkatraman, "Wakeup scheduling in wireless sensor networks," in *ACM Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*, 2006.

[32] A. Dunkels, "Rime - a lightweight layered communication stack for sensor networks," in *Proceedings of the European Conference on Wireless Sensor Networks (EWSN)*, Delft, The Netherlands, 2007.

[33] S. Chaudhary, N. Singh, A. Pathak and A. Vatsa, "Energy Efficient Techniques for Data aggregation and collection in WS," *Int. J. Comp. Sci. Eng. Appl.(IJCSEA),* vol. 2,

no. 4 , pp. 37-40, 2012.

[34] R. Bista and J.-W. Chan, "Privacy-preserving data aggregation protocols for wireless sensor networks: a survey," *Sensors,* vol. 10, no. 5, pp. 4577-4601, 2010.

[35] B. Krishnamachari, D. Estrin and S. Wicker, "The impact of data aggregation in wireless sensor networks," in *IEEE 22nd International Conference on Distributed Computing Systems Workshops*, 2002.

[36] J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin and D. Ganesan, "Building efficient wireless sensor networks with low-level naming," *ACM SIGOPS Operating Systems Review,* vol. 35, no. 5, pp. 146-159, 2001.

[37] R. Rajagopalan and P. K. Varshney, "Data aggregation techniques in sensor networks: A survey," *Electrical Engineering and Computer Science,* 2006.

[38] S. Sirsikar and S. Anavatti, "Issues of Data Aggregation Methods in Wireless Sensor Network: A Survey," *Procedia Computer Science,* vol. 49, pp. 194-201, 2015.

[39] T. Adame, "Design of the ENTOMATIC gateway," 2015.

[40] W. R. Heinzelman, J. Kulik and H. Balakrishnan, "Adaptive protocols for information dissemination in wireless sensor networks," in *5th annual ACM/IEEE international conference on Mobile computing and networking*, 1999.

[41] J. Kulik, W. Heinzelman and H. Balakrishnan, "Negotiation-based protocols for disseminating," *Wireless Networks,* vol. 8, no. 2/3, pp. 169-185, 2002.

[42] C. Intanagonwiwat, R. Govindan and D. Estrin, "Directed diffusion: a scalable and robust communication paradigm for sensor networks," in *Proceedings of the 6th annual international conference on Mobile computing and networking. ACM*, 2000.

[43] D. Braginsky and D. Estrin, "Rumor routing algorthim for sensor networks," in *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications. AC*, 2002.

[44] R. C. Shah and J. M. Rabaey, "Energy aware routing for low energy ad hoc sensor networks," in *Wireless Communications and Networking Conference, WCNC2002*, 2002.

[45] A. Dunkels, J. Alonso and T. Voigt, "Making TCP/IP viable for wireless sensor networks," *SICS Research Report,* 2003.

[46] C. Wang, K. Sohraby, B. Li, M. Daneshmand and Y. Hu, "A Survey of Transport Protocols for Wireless Sensor Networks," *IEEE Network,* vol. 20, no. 3, pp. 34-40, 2006.

[47] A. Dunkels, J. Alonso, T. Voigt and H. Ritter, "Distributed TCP caching for wireless sensor networks," SICS Research Report, 2004.

[48] T. S. Company, "Datasheet SHT1x (SHT10, SHT11, SHT15)," 2008.

[49] AOSONG, "Temperature and humidity module DHT22/AM2302 Product Manual," [Online]. Available: http://akizukidenshi.com/download/ds/aosong/AM2302.pdf.

[50] S. S. O. &. E. C. Ltd., "GL55 Series Photoresistor".

[51] E. B. M. Inc., "Nickel Metal Hydride (NiMH) Handbook and Application Manual," 2010.

[52] SIMCom, "SIM900 AT Command Manual v1.05," 2011.