University of Franche-Comté

UFR STGI - Master 2 - IDO

Deployment of a wireless sensor network (WSN) used to control the photo-voltaic production and the recharging of electric vehicles.

Internship Report

Author: ABDEL KHALEK Farah
Supervisors : Dr. NASSAR Jad
Dr. LEFEVERE Vincent

Lille, 2020

Acknowledgments

First of all, I would like to thank HEI - École des Hautes Études d'Ingénieur for welcoming me as part of their team during this internship. It was a considerable opportunity for starting my own career in the research and development field, as well as growing my network and meeting wonderful people who led me throughout this period.

Mainly, I would like to express my greatest gratitude to Dr. NASSAR Jad, my first supervisor, for supporting me and guiding me throughout the whole project. All of that beside his friendly attitude and extremely amiable behavior that helped me adjust to the pace of work and give my absolute best.

Along with the direction of Dr. NASSAR, my second supervisor was Dr. LEFEVERE Vincent, who also participated in carrying out the project at his esteemed organization. I would like to address him my deepest thanks for all the necessary advices and recommendations that he provided me with. Moreover, Dr. LEFEVERE was always there for me whenever I faced any technical issues, and enhanced my knowledge with his huge experience in computer science.

Other staff from HEI and the Catholic University of Lille also cooperated, each with their own expertise and skills, in order to complete the project successfully. I choose this moment to acknowledge their contribution gratefully.

I would also like to highlight the fact that the project was accomplished remotely, due to Covid-19, and that my supervisors managed to follow-up my work process every week, and kept me on the correct path during this tough period.

Last but not least, my greatest appreciation goes to Dr. MAKHOUL Abdallah, my master's supervisor, for giving me the chance of being part of this extremely interesting program overseas and therefore becoming the engineer I aim to be.

Abstract

The internship took place in the HEI laboratory - OMI department. It lasted for a period of six months, under the supervision of Dr. NASSAR Jad and Dr. LEFEVERE Vincent. The project's objective is to deploy a wireless sensor network (WSN) used to control the photo-voltaic production and the recharging of electric vehicles.

The increasing integration of renewable energy resources (e.g., photo-voltaic cells, wind farms) has turned traditional electric grids into smart ones. This evolution takes an active role in smart energy systems, especially when integrating a Wireless Sensor Network (WSN) to control and manage the grid. However, monitoring the whole grid with constrained devices is a challenge worth considering.

In this work, we introduce a WSN demonstrator for Smart Grids as part of the SoMel SoConnected project. It consists of Raspberry Pi and Zolertia Re-Mote sensor nodes deployed all over the building of the Catholic University of Lille and Yncréa Hauts de France. Thus, providing a WSN testbed in a real Smart Grid environment.

A performance evaluation is conducted in the wired and wireless architectures in order to test some of the metrics that could be evaluated in this testbed, particularly the end to end delay and the packet delivery ratio.

Index Terms - Smart Grids, Wireless Sensor Networks, Contiki OS, Zolertia RE-Mote, Raspberry Pi, MQTT , Modbus, ZigBee

Contents

1 Introduction **1**

2 State of the art **4**

3 Used Tools **5**

3.1 Zoletia Re-Mote 5

3.1.1 Definition and Programming 5

3.1.2 Flashing via USB 6

3.1.3 Difference Between Remote And Firefly 7

3.2 Contiki Operating System 8

3.2.1 Installation 9

3.2.2 Configuration 10

3.2.3 Flashing the Zolertia from Raspberry Pi 3 13

3.3 RF Explorer 17

3.3.1 Objective 17

3.3.2 Description of the main characteristics 17

3.3.3 RF Explorer Standard Model 18

3.3.4 Manage input power levels correctly 19

3.3.5 Spectrum Analyzer - Main Screen 20

3.3.6 RF Explorer Antennas 22

3.4 MODBUS Protocol 24

3.4.1 Introduction 24

3.4.2 Serial transmission modes 24

3.4.3 MODBUS TCP/IP 25

3.5 MQTT Protocol 28

3.6 Network Time Protocol 29

4 Global Architecture **31**

4.1 Hardware Architecture 32

4.2 Software Architecture 32

5 Example of Use Cases **35**

5.1 Sensor Deployment 36

5.2 Experiment Parameters 36

5.3 Performance Evaluation 38

6 Conclusion, Actual State and Future Work **40**

References **i**

List of Figures **iv**

List of Tables **v**

Introduction

The transformation of traditional power grids into "smarter" ones has become more and more common. This evolution, called the Smart Grids, is manifested by (1) the integration of renewable energy resources all over the grid [1], (2) a two-way communication between the utility and the customers and (3) automated decisions of the smart connected devices. Therefore, these changes require the ability to exchange a maximum of data over the network, in order to monitor and control the different heterogeneous decentralized energy resources.

A Wireless Sensor Network (WSN) distributed across the grid, on the different measuring and control points, is a potent and plausible solution [2].

However, sensor nodes are highly constrained devices. Their limitations in terms of power are due to their continuous radio transmission. Although WSN can drive a SG, it cannot always be powered by it due to the intermittent energy resources (i.e., renewable energy resources). Moreover, and on many levels in the power grid, such high voltage will make it challenging in terms of sensor placement and electric supply. For that, and in order to meet the Quality of Service (QoS) requirements of the different SG applications (e.g., delay tolerance, reliability, etc), a WSN controlling/monitoring a SG have to take these challenges into consideration [3].

Launched in 2013 by the Catholic University of Lille, LiveTREE represents the university's program for energy and social transition. It registers under Rev3 – the Third Industrial Revolution in the Hauts-de-France region through a collaborative approach between the students, personnel, residents, companies and collectivities. The project aims to reduce the campus' carbon footprint and transform it to a living laboratory of social innovation.

The innovative solutions envisioned focus on the technical and human aspects

of the transition and are tested in real conditions on the campus and its neighborhoods. The energy aspects are particularly explored in LiveTREE[1], notably with the SoMEL SoConnected project which is part of YOU & GRID[2]. The aim of this project is to enable the development of energy distribution networks to serve a territory through the connection of infrastructures. In light of that mission, the Faculties of the Catholic University of Lille and the Yncréa Hauts-de-France engineering school are developing self-consumption models of renewable energy in the non-residential sector. In this project, they test the technical feasibility and profitability of self-consumption models in the non-residential sector with building-integrated photo-voltaic, by piloting charges including recharging electric vehicles in private car parks.

As part of this project, a wireless sensor demonstrator is developed. It consists of 120 Raspberry Pi and Zolertia sensor nodes distributed all over the building of the University.

The aim of this demonstrator is to collect different sensor values from the photo-voltaic cells, electric vehicles, electric meters, etc. in order to control and pilot the energy production/consumption on campus. More precisely, two network architectures are implemented: the first one having a wired connection between the different measuring points and the sensors, based on Raspberry Pi version 3 (RPI) modules. The second one relies on low power wireless personal area (6LoWPAN) networks with Zolertia Re-Mote sensors. The collected data are sent over both networks via an MQTT broker/client communication and are stored in a database for a future use.

It is important to mention that the wired network is used to control, monitor and reprogram the wireless sensor network (e.g., detect transmission loss, provide a benchmark for wireless routing measurements, etc.).

[1]http://livetree.fr/

[2]https://www.enedis.fr/yougrid-northern-france

Figure 1: Example Of Sensor Deployment In One Floor In The University

We note that the physical installation of the sensors in the building is in process. Figure 1 shows an example of the sensor deployment on one floor of one of the buildings. We used the range-test application [4] in order to ensure an efficient and adequate placement of the Zolertia Re-Mote nodes..

The remainder of this work is organized as follows. Section 2 presents a summary of the previous work on WSN demonstrators. Section 3 describes the different tools utilized in the realisation of the project. It includes the Zolertia Re-Mote sensors, RF Explorer module, MQTT, MODBUS and Network Time protocols. Section 4.1 and 4.2 introduce the hardware and software tools used for the design and deployment of the demonstrator's WSN. Section 5 illustrates a performance comparison between a WSN and a wired network in a real-case environment. Finally, section 6 concludes the paper.

State of the art

The heterogeneity of IoT applications leads to a large diversity of WSN testbeds, supporting different topologies and network layer protocol options. Many wireless sensor network testbeds are available to the community today. In [5], the authors present FIT IoT-LAB, an open testbed that consists of 2728 motes located in 6 different sites across France, offering an accurate open-access and open-source multi-user scientific tool. It provides repeatable mobility via electric toy trains, multi-site experiments and energy consumption measurement. The result is a heterogeneous testing environment, which covers a large spectrum of IoT use cases and applications. IoT-LAB is a one-of-a-kind facility, allowing anyone to test their solution at scale, experiment and fine-tune new networking concept. In [6], TWIST supports experiments with diverse node platforms, active power supply and creation of hierarchical and flat sensor networks control of the nodes. The self-configuration capability, the use of hardware with standardized interfaces and open-source software makes the TWIST architecture scalable, affordable, and easily applicable. In [7], SmartSantander offers advanced hardware, scale, functionalities proposed to the user, mobility via public buses Multi-site and real-life experiments. It deeply embeds heterogeneous IoT devices as a programmable experimentation substrate in a real life office environment and makes flexible experimentation with real end users possible. In [8], RealNet is a testbed which objective is to monitor environmental parameters from the air, water and ground, and to become a generic platform that serves different purposes. In [9], SDNWisebed: an SDN-empowered WSN testbed management system that enhances the WSN management functions with a stateful Software-Defined Networking solution is developed.

However, all of these demonstrators generally operate in controlled environments and are not so open to public.

Used Tools

To build our testbed, several hardware and software tools were implemented. Before describing the global architecture, we are focusing in this section on the importance of each material used, its necessity and the way it is implemented. We are discussing as well the appropriate software integrated with each material, and its role in creating the IoT testbed.

3.1 Zoletia Re-Mote

The Zolertia Re-Mote Revision B is the type of sensor deployed in this project.

3.1.1 Definition and Programming

This tool has several features listed below:

- An industrial-grade architecture.
- Ultra-low energy consumption.
- Protocols: UDP, TCP, HTTP, MQTT, LWM2M.
- Addressable over IPV6 above 6LoWPAN.

Figure 2: Zolertia Sensor

- This is a self-healing network of wireless mesh: if a route disappears, a given RE-Mote will find a new route and restore the network, no need to configure static routes manually.
- Uses 60 to 300 times less power than any WIFI device: 1 uA to 150 nA.
- This can range from 100 meters to 5-10-20 kilometers in one jump.
- OS: Contiki, RIOT, Openthread.
- Secure: SHA2, AES-128/256, ECC-128/256 and RSA for a secure key exchange.
- Flashing is done via USB and the compiled binary is transferred to the device where the bootloader ROM executes the image after it resets.

3.1.2 Flashing via USB

The flashing process via a USB cable is done using a series of commands to be used in a linux environment.

- pip install intelhex
- pip install python-magic
- Instant-Contiki is downloaded on a virtual machine
- The Zolertia Re-Mote sensor is enable in the VM
- In the directory /contiki/example/cc2538dk:
  make TARGET=cc2538dk cc2538-demo

To activate the Bootloader mode and thus be able to first flash the Zolertia module, pressing and holding the user button should be done, then pressing and holding

the reset button. Finally, releasing the reset button and the user button. These buttons are displayed in figure 3.

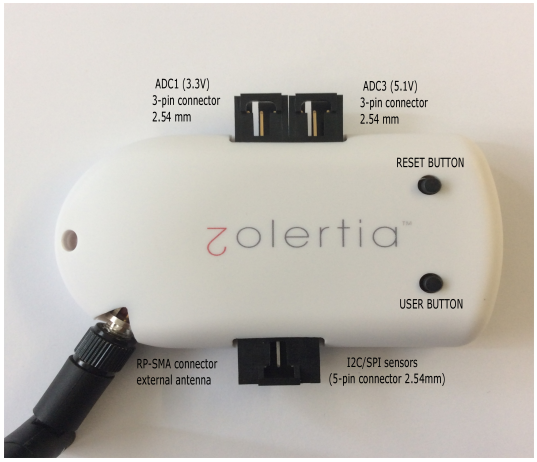

Figure 3: Zolertia Re-Mote Buttons And Pins

3.1.3 Difference Between Remote And Firefly

In this section, table 1 detailing the main differences between Re-Mote and Firefly nodes is presented. It focuses on the antenna type, consumption and power rates, form factor, micro-SD type, RTC, WDT, shutdown mode, utilisation technique and the development platform.

	REMOTE	**FIREFLY**
Antenna	Programmable RF switch for connecting an external antenna through the RP-SMA connector to either the 2,4 GHz or the Sub 1 GHz RF interface	Printed on-board PCB sub-1GHz antenna, preferably u. FL for external antennas under 1GHz/2.4GHz
Consumption	Ultra low	-
Power	Built-in battery charger (500 mA)	No battery charger
Form Factor	73 x 40 mm	53x25mm
Micro SD (external storage)	Yes	No
Real time clock RTC	Yes	No
Watch dog timer WDT	Yes	No
Shutdown mode	Yes (sleepy nodes)	No
Utilisation	Prototyping applications for wireless networking with 6LoWPAN communication protocol	Specific to IoT applications and services, with long distance operation and low power consumption
Development	Full framework for Hardware development	Bottom line growth

Table 1: Difference Between Zolertia And Firefly

3.2 Contiki Operating System

Contiki is a highly compact, open source, multitasking operating system with a small power usage for wireless sensor networks. Contiki is therefore well adapted to program a module in Zolertia. This section explains how to implement the Contiki operating system on a Zolertia, step by step.

3.2.1 Installation

There are several ways to install Contiki OS on the machine, either by downloading it on an Ubuntu-based VM, or by importing a VM called InstantContiki in which the Contiki OS is integrated. In this section, the first installation option is detailed, by creating a new virtual machine based on Ubuntu 32bits.

1. After the virtual machine is created and configured, the following command is used:
   git clone –recursive https://github.com/contiki-os/contiki.git
   It retrieves and downloads the Contiki repository's last working copy and all of the sub-modules. To get a permanent working copy:
   wget https://github.com/contiki-os/contiki/archive/3.0.zip

2. The downloaded file is unzipped, supposing in the /home/user/ directory, with the command:
   unzip 3.0.zip
   then the folder is renamed from contiki-3.0 to contiki using:
   mv contiki-3.0 contiki
   Next, the following link is typed and the DownloadLatestVersion button is clicked:
   https://sourceforge.net/projects/contiki/files/Instant%20Contiki/Instant%20Contiki%203.0/

3. After downloading the zip file and unzipping it, the disk in VMWare is opened and the "I COPIED IT" option is chosen.

3.2.2 Configuration

After the operating system has been mounted, it's time for the configuration. All applications for different platforms would need to be activated utilizing the specified commands.

- Ubuntu 14.04

  sudo apt-get install build-essential binutils-msp430 gcc-msp430 msp430-libc msp430mcu mspdebug binutils-avr gcc-avr gdb-avr avr-libc avrdude openjdk-7-jdk openjdk-7-jre ant libncurses5-dev

  sudo apt-get install gcc-arm-none-eabi

- Ubuntu 16.04

  sudo apt-get install build-essential binutils-msp430 gcc-msp430 msp430-libc msp430mcu mspdebug gcc-arm-none-eabi gdb-arm-none-eabi openjdk-8-jdk openjdk-8-jre ant libncurses5

In case of operating on a 64-bit computer, many issues may be found with the executable serialdump-linux, because it might have been optimized for 32-bit computers. To overcome this question a library should be installed using the link:
sudo apt-get install lib32ncurses5
An essential step to do is to install these libraries as well:
sudo pip install pyserial
sudo pip install intelhex sudo pip install python-magic
Compiling the code for the native device (to be used while there is no sensor attached to the laptop) is carried out on a first try. To do so, the hello-world example is chosen in the directory:
cd /contiki/examples/hello-world

Then the native platform is chosen using:

make TARGET=native

If there is a error when compiling that says " #include: No such file or directory", the following steps should be executed on Ubuntu:

sudo apt-get install libncurses5-dev for which "apt" for 32-bit should have been installed using:

sudo dpkg -i apt_0.8.16 exp12ubuntu10.10_i386.deb

Running the program is done using this command :

./hello-world.native

The output of this program is shown in figure 4.

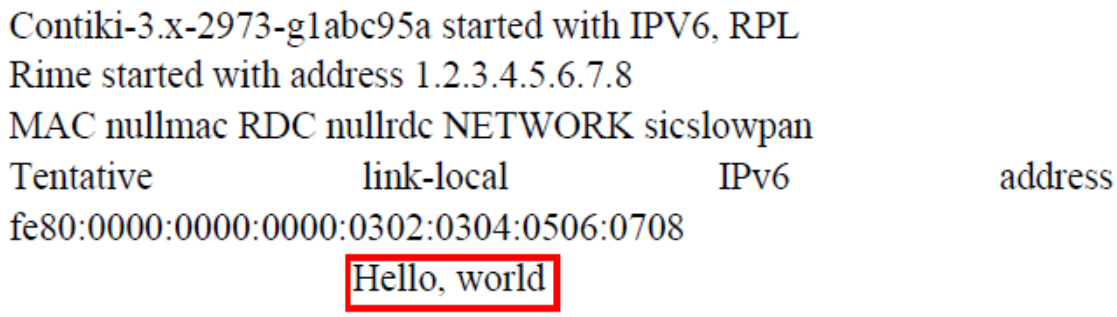

Figure 4: Hello-World Example

To run the code on the Zolertia module, it will be necessary to modify the target platform, so TARGET= native is replaced by TARGET= zoul

cd /contiki/examples/hello-world

make TARGET=zoul hello-world

To save the target, this command is executed:

make TARGET=zoul savetarget

Before continuing, this command must be typed so that the current user can access the device connected to the USB (in more technical terms, non-root access

to /ttyUSB0 should be permitted) by doing :

sudo usermod -a -G dialout $USER

Checking the used port is done using: *ls -l /dev/ttyUSB0*

The display should look like this:

crw-rw—T 1 root dialout 188, 0 Feb 12 12:01 /dev/ttyUSB0

Then, the machine is restarted using :

sudo reboot

At the end, the hello-world program is compiled:

make hello-world.upload

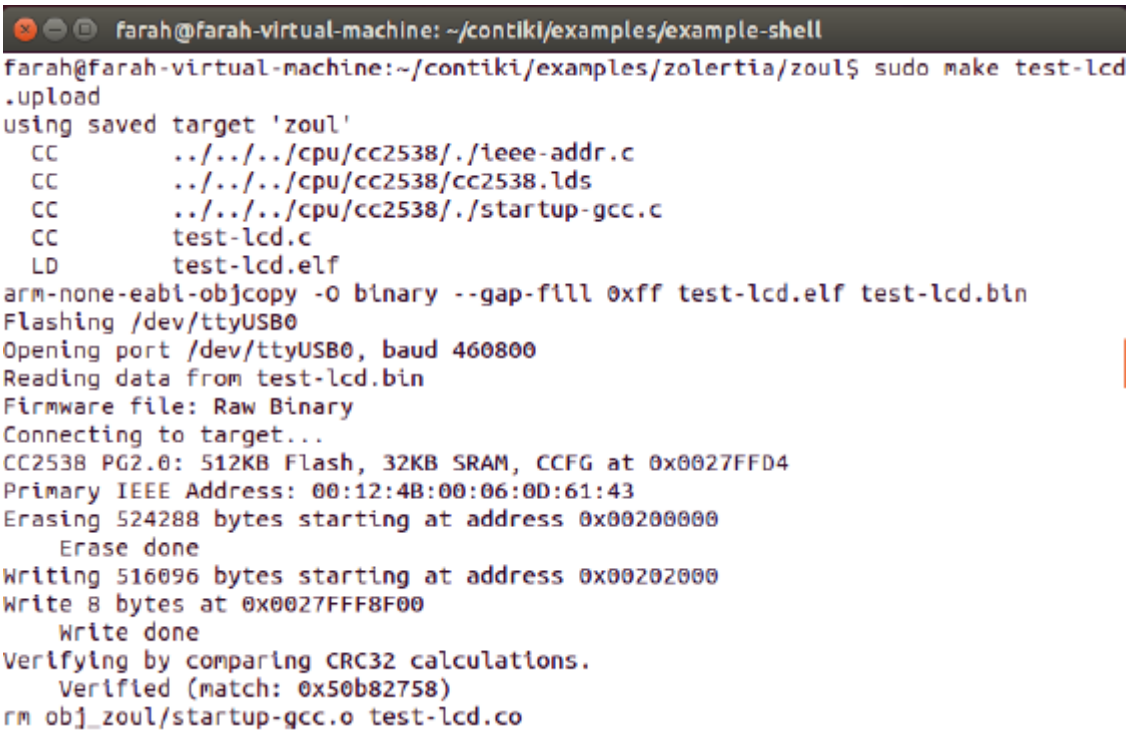

Figure 5: Compilation Output

Figure 5 is the show that the user would imagine. While the message is flickering, he should be able to see the Zolertia blinking. If the *make login* command

is used, he'll be able to visualize the execution of the code on his terminal. But an error is displayed which is presented in figure 6.

```
using saved target 'zoul'
../../tools/sky/serialdump-linux -b115200 /dev/ttyUSB0
../../tools/sky/serialdump-linux: 1: ../../tools/sky/serialdump-linux: Syntax error:
")" unexpected
../../platform/zoul/Makefile.zoul:115: recipe for target 'login' failed
make: *** [login] Error 2
```

Figure 6: Error Display

To solve it, these commands are necessary:
cd tools/sky && rm serialdump-linux && make serialdump && mv serialdump serialdump-linux

3.2.3 Flashing the Zolertia from Raspberry Pi 3

The goal is to manage the Zolertia from Rasbperry, then it will be necessary to install, following the same steps, the operating system Contiki on the Raspberry Pi.

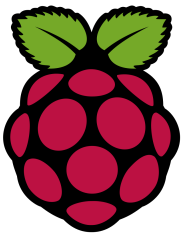

Figure 7: Raspbian Buster Software

Raspbian is the official operating system for all models of the Raspberry Pi. It is a Linux distribution. Rather than a brand new OS, Raspbian is based on the

Debian OS. It is optimized for the Raspberry Pi. Below are the steps needed for the flashing process:

1. The SD card is inserted into the adapter and the adapter into a USB port on the PC.

2. From the link :
   https://www.raspberrypi.org/downloads/raspbian/
   RASPBIAN Buster LITE is downloaded in zip version. The lite version is used here because the rapsberry GUI is not needed (interface with a mouse). Instead, the command prompt (also called shell) in its bash version is used.

3. The Raspbian operating system is in the zip file on the computer. Now, to write it to the SD card, a simple copy/paste can't be done. This is because linux files are organized in a different way than Windows files. So flashing the card is mandatory, using this command:
   https://etcher.io/

4. The software is then downloaded.

5. Flashing the sd card process takes a few minutes. Note that there is a "Validating" step where etcher verifies that the flash was successful. Once finished, the key is disconnected.

6. A delay of 5 seconds is needed to plug it back in: this will force Windows to re-check the contents of the memory card.

7. In the file explorer, one or two new disks appears. The one to be interested in is called "boot". In fact: the memory card has been partitioned i.e., divided "software" into two logical subspaces. One is called boot and is readable by Windows and linux. The other one has no imposed name and is unreadable

by Windows. When trying to access it, Windows will ask to format it. The boot partition has been set up by Raspbian to allow editing from Windows of the Raspbian configuration files. One will be able, for example, to configure ssh and wifi from now on from Windows.

8. Since working with windows, Notepad++ is installed from the link
   https://notepad-plus-plus.org/downloads/
   and then launched.

9. A new file in Notepad++ is then created. In the Edit menu – Convert line breaks – Convert to unix format (LF). This file is saved under boot by choosing the "all types" option with the name : "ssh". This file will be understood by Raspbian: it will start the ssh server when the Raspberry is powered up to allow you to connect to it with Putty on Windows or from the terminal on MAC.

10. To tell the Raspberry Pi to automatically connect to a specific Wifi network, one needs to edit a file called: wpa_supplicant.conf.
    To do so, a file with Notepad++ is created under the name "wpa_supplicant.conf" (always with quotes + unix end of line + always in the "boot" disk) with the following content:
    network= { ssid="..." psk="..." }

11. As linux files are organized in a different way than Windows files, it is necessary to download a specific software to be able to access the Linux configuration files in order to modify them. It is installed from the link below:
    https://www.paragon-software.com/home/linuxfs-windows/.
    Once finished, the "rootfs" partition appears in the file explorer.

12. Configuring the wlan0 interface : In the interfaces folder, the following lines

are added to tell the Raspberry Pi to allow wlan (wireless connection) as the network connection method and to use /etc/wpa_supplicant/wpa_supplicant.conf as the configuration file.

- auto wlan0
- allow-hotplug wlan0
- iface wlan0 inet dhcp
- wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
- iface default inet dhcp

Now, the SD card can be unplugged from the PC, and plugged into the Raspberry, then the Raspberry can be powered on.

13. For a remote connection to the Rasperry Pi, the Putty software can be utilized under windows, or Terminal under MAC or Linux: ssh pi@IP_ADDRESS. When the connection is established, it will be necessary to identify the user on the Raspberry with the appropriate coordinates:
Username: ...
Password: ...

Figure 8: Putty Software

14. In what follows, it will be necessary to install Contiki on Raspbian in the same way as explained in the previous section.

3.3 RF Explorer

3.3.1 Objective

This is an excellent monitoring tool used to track wireless and communication networks and troubleshoot them.

3.3.2 Description of the main characteristics

- Tank size and light weight with solid metal shell.
- The spectrum analyzer calculator contains Peak Max, Max Hold, Normal, Overwrite and Averaging modes.
- Free lifetime firmware updates available.
- Open to new features the user is demanding.
- High capacity Lithium ion battery with continuous operation up to 16 hours, rechargeable via USB.
- Microsoft Windows software is Open Source and free.
- The Mac OS client is also Open Source and free.
- The communication protocol USB is open to customized implementations and extensions.
- Different models of RF Explorer are available :
  - High frequency general purpose model: 15-2700 and 4850-6100MHz.
  - System broadband general purpose: 15-2700MHz.

  - UHF-ISM versions wideband: 50Khz-960MHz and 2350-2550MHz.
  - ISM versions for the Narrowband: 2.4GHz, 433MHz, 868MHz or 915MHz.

- Expandable: With the internal expansion port the base unit of the RF Explorer, platform can be conveniently extended to additional bands.
- Many versions come with an integrated RF signal generator.

3.3.3 RF Explorer Standard Model

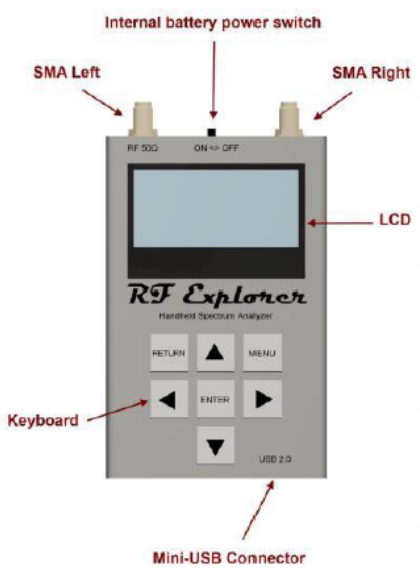

Figure 9: RF Explorer

The standard RF Explorer model, shown in figure 9, comes with 7 working buttons on the front side. The package also includes a typical SMA style RF connector with at least one 50 ohm impedance. For extended range and functionality, all models have a left SMA connector mounted and as an alternative, some models have a second SMA connector in the proper position. When a compatible mini USB cable is attached, the device will automatically start running on USB power.

Important: If the USB cable is attached, the internal battery control switch must be in the ON place for battery charging. If the switch is in the OFF position, the internal battery does not charge.

3.3.4 Manage input power levels correctly

RF Explorer is specially designed to track low-power signals using antennas. Any extra steps can be performed directly. External attenuators should be used for better performance when working with powerful signals.

Measurable Input Range	**Recommanded External Attenuator**
Lower than -30dBm / 1 micro Watt	No need
-30 to 0dBm / 1 microWatt to 1 milliWatt	30dB external or internal
-30 to 10dBm / 1 microWatt to 1 milli-Watt	40dB external
0 to 33dBm / 1 milliWatt to 2 Watt	60dB external

Table 2: Input Power Levels

Note that by using spectrum analysers, there are three different types of power levels to be aware of, from the lowest to the highest:

- Measurable input range: This range encompasses all rates that the analyzer can calculate accurately without distortion or incorrect readings. Table 2 above shows typical RF Explorer levels.
- Secure but not detectable range: The instrument will not be harmed within this range, the calculation will be skewed and, in most cases, inaccurate.
- Damage range: The instrument may be permanently damaged within this range, and readings can at any time be incorrect.

3.3.5 Spectrum Analyzer - Main Screen

In this section, the main screen features are displayed in details, in its standard and advanced modes. The RF Explorer switches automatically to spectrum analyser mode when started.

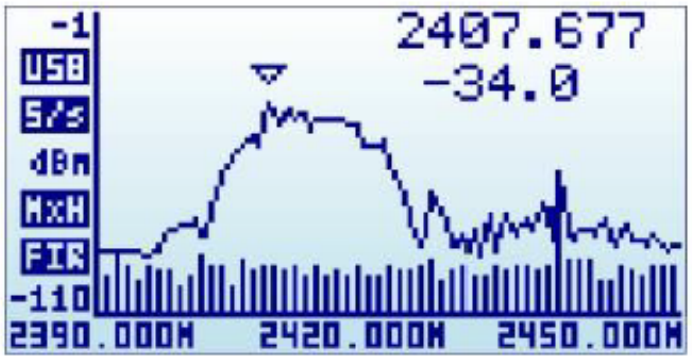

Figure 10: Spectrum Analyzer Main Screen

The X axis reflects the MHz frequency, and the Y axis shows the actual power obtained in dBm or dBuV (selectable). In the above case, the frequency range is 2390 MHz to 2450 MHz (a range of 60 MHz), and the visual amplitude ranges between -1 dBm and -110 dBm.

A small triangle-shaped marker is shown automatically at the main screen. It sets out the initial frequency max, with the first line of text being the frequency in MHz and the magnitude at that stage the second. The available metrics are read as follows:

- USB / battery status: If a valid 5V USB link is open, this indicator will display USB1. Alternatively, when the RF Explorer power switch is set to ON, a battery icon with a charge level indicator will be shown. When both connections are activated then the USB and battery will alternate and the battery will charge in this scenario.

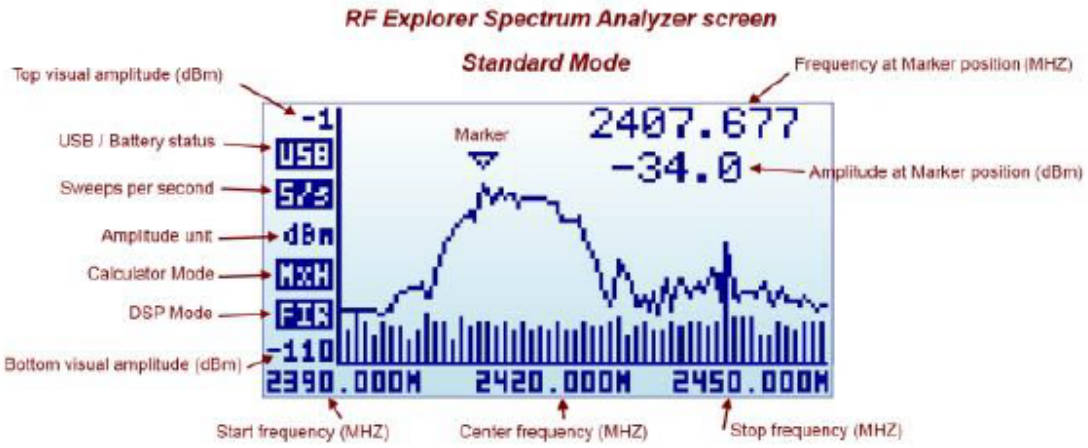

Figure 11: Standard Mode

- Sweeps per second: This is about the amount of full screen scans per second. There are 5 sweeps per second in the above example, or, in equivalent terms, one sweep per 200 milliseconds.

- Calculator mode: This indicator can have different values in the frequency screen, as shown by the calculator mode.

- DSP Mode: This variable indicates the analyser's actual value used. The suggested DSP is auto in the frequency menu, so that RF Explorer selects the best choice as shown below.

- FST: Fast mode. This is available in all versions as standard mode.

- FIR: Filter mode. This mode is available only in the modules 15-2700MHZ and Plus, and is the recommended mode for those versions.

- Marker: The marker has multiple operating modes. Using the [Return] key to bring the analyzer screen into advanced mode.

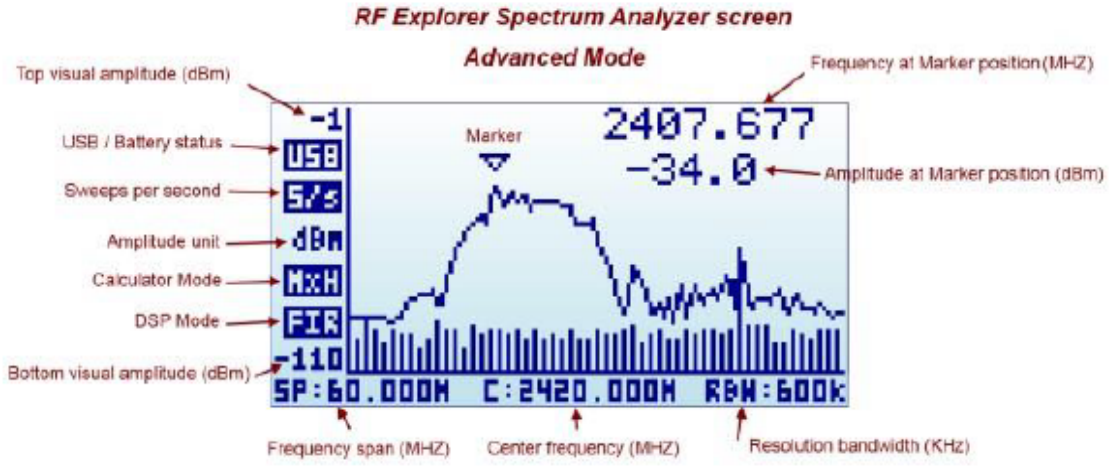

Figure 12: Advanced Mode

3.3.6 RF Explorer Antennas

Figure 13: Two Types Of Antennas

1. Nagoya Telescopic NA-773

   It offers a strong response with all frequencies under 1GHz . Using this antenna across all 15-1000MHz frequency ranges.

2. Whip dipole antennas

   RF Explorer 868 M: Contains an 868MHz optimized 2dBi antenna.

3. RF Explorer Near Field antennas

   RF Explorer Near Field Antennas is a group of 4 high-performance antennas equipped for the most challenging diagnostic RF tasks:

Figure 14: Near Field Antennas

The antenna comes with a semi-flexible RF cable of high quality; this cable acts as the antenna manager.

Figure 15: Semi-Flexible RF Cable

3.4 MODBUS Protocol

3.4.1 Introduction

This protocol specifies a message structure to be understood and used by controllers regardless of the form of network they are interacting with. It explains the mechanism that a controller uses to seek access to another computer, how it will respond to requests from other devices and how it can identify and disclose errors. The Modbus protocol is applied on Modicon controllers. It is a master-slave technology, where the transaction (request) may only be made by one person(master). The others (slaves) respond by sending the required data to the master or by performing the requested action in the specification. In this project, the Raspberry Pi 3 is considered the master, while the photo-voltaic panels are treated as the slaves. The master can address individual slaves, or submit a broadcast message to all slaves addressed to them individually.

3.4.2 Serial transmission modes

Two serial transmission modes are available. For all machines in a Modbus network, the serial mode and parameters must be the same, because those two modes are incompatible.

1. ASCII: Every 8 bits are an ASCII character. Its advantage relies in longer time cycles without creating errors between characters.

Start	Slave Address	Function Code	Data	LRC check	End
1 char	2 char	2 char	N char	2 char	2 char

Table 3: ASCII Message Frame

2. RTU: Every 8 bits comprises 2 hexadecimal 4-bit characters. Its advantage relies in a good throughput of results.

Start	Slave Address	Function Code	Data	CRC check	End
T1-T4	8 bits	8 bits	Nx8 bits	16 bits	T1-T4

Table 4: RTU Message Format

3.4.3 MODBUS TCP/IP

Modbus TCP / IP, instead of master-slave, uses the term client-server. The TCP / IP network consists of a client connecting to a switch, or a set of switches, of which all network servers are linked as well. The Internet Protocol (IP) addresses are used by the Modbus TCP / IP network and include a subnet mask. The default gateway is facultative.

Masters are the clients, and the slave is the server. Thus, it is the clients that have to read and write to the Modbus server. Every client will use the TCP protocol (server IP address, port 502) to connect to the server. Note that one server may hold several masters (clients), and clients cannot connect to other clients. It is important to mention that servers cannot create requests on their own as well.

Figure 16 is a representation of a client-server model.

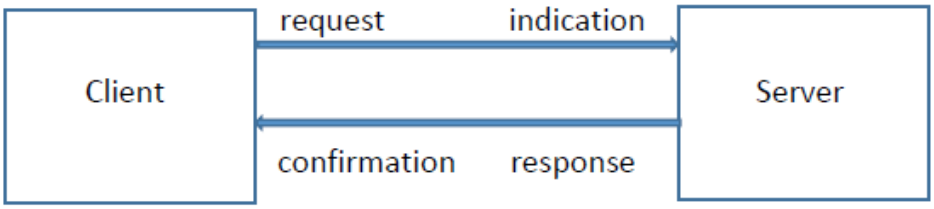

Figure 16: Client-Server Model

It is centered on four messaging types:

1. Modbus request

2. Modbus confirmation

3. Modbus indication

4. Modbus response

The Modbus request is the client's message sent to trigger a connection across the network. The indication is the same request but from the server side. The latter's response is the Modbus response. And the confirmation is the same response received from the client side.

The message format of this type of communication differs from the previous one. To start a transaction, the device builds the Application Data Unit (ADU).

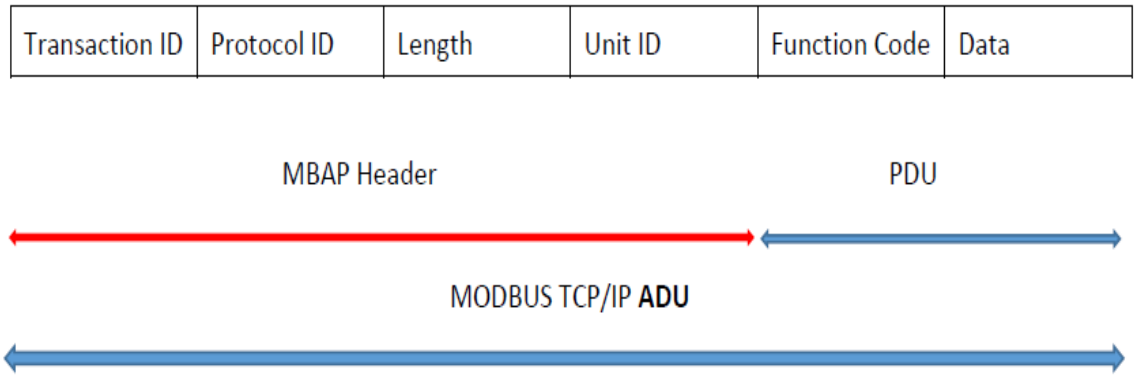

Figure 17: TCP/IP Message Format

The Transation ID field is used for transaction pairing. The Modbus server copies the request's transaction identifier into the response. The Protocol ID is used for multiplexing intra-system operations. The value 0x0000 defines the Modbus Protocol. The Length specifies the number of bits to follow, including the Unit ID field and the Data. The Unit ID field is used for purposes of intra-system routing. This is usually used to communicate with a serial line slave Modbus or Modbus+ through a gateway, router, or bridge between the TCP / IP Ethernet network and a serial line from Modbus. In the request, this field is specified by the Modbus client and must be returned by the server with the same value in the response.

Moreover, Modbus function codes are integer codes telling the slave what table to reach and how to read or write to that table. Each function code refers to a specified address range of the data table. Table 5 below illustrates several function codes that may be encountered. In the case of an exception response, the func-

01	Read coil status
02	Read input status
03	Read holding registers
04	Read input registers
05	Write single coil status
06	Write single register
0A	Multiple coil write
0B	Multiple register write

Table 5: Modbus Function Codes

tion code server becomes equal to: function code client + 80H. Some exception examples are presented in table 6.

01	Illegal code
02	Illegal address
03	Illegal data value
04	Server failure
05	Acknowledge
06	Server busy
0A	Gateway problem

Table 6: Function Code With Exceptions

If the transaction identifier in the header MBAP is not referring to the current Modbus transaction, the reply shall be refused. Otherwise, an analysis of the response is needed. Therefore, the following steps are established:

- Check the Identifier protocol (= 0x0000).
- The unit identifier (0x00FF) is meaningless and must be ignored if the device is directly connected to the TCP / IP network.

- The Device Identifier (= 0x00FF) identifies the server if the latter is linked to a serial line subnet and the response is from a bridge, router or gateway.

In the PDU packet, if the function code server is equal to the function code client and is in a correct format, or if the function code server is equal to an exception code, it then leads to a positive response. Otherwise, if the function code server differs from the function code client or it is in an incorrect format, it leads to a negative response.

3.5 MQTT Protocol

There are protocols that ensure communication between several heterogeneous or homogeneous applications in an asynchronous way. One of the most widespread protocols that works this way is the MQTT.

As its name indicates, it is a protocol that guarantees communication between many applications, whether similar or not, through the capability of transporting messages between them. It's widely used in the Machine to Machine (M2M) and IoT domain.

Basically, when MQTT is used for communication, there will be two types of endpoints applications. The first one sending messages to the rest known as publisher / sender. The other applications receiving the corresponding messages known as subscriber / consumer / receiver. In fact, using this protocol, the two applications will not communicate directly with each other. There is an intermediary program between them which is the Message Broker. The process that will take place is the following: A publisher will transmit data (in the form of a string message) to the broker. The latter will take the collected message and send it to all the subscribers. Note that even if a consumer is not available, there won't be any data loss. The broker will stock the data in a queue to be re-transmitted at a later stage. This is what makes the MQTT protocol asynchronous.

Moreover, it is very important for the broker to be able to filter the messages so that he knows to which consumer he has to forward them. This is where MQTT topics come into play. In fact, they are all represented by a single string and differentiated by a "/" separator. The publisher must associate the topics with the messages he wants to publish, and the consumer must specify the topics of the messages he wants to receive. In this way, when a message is received, the broker will only send it to the right consumers.

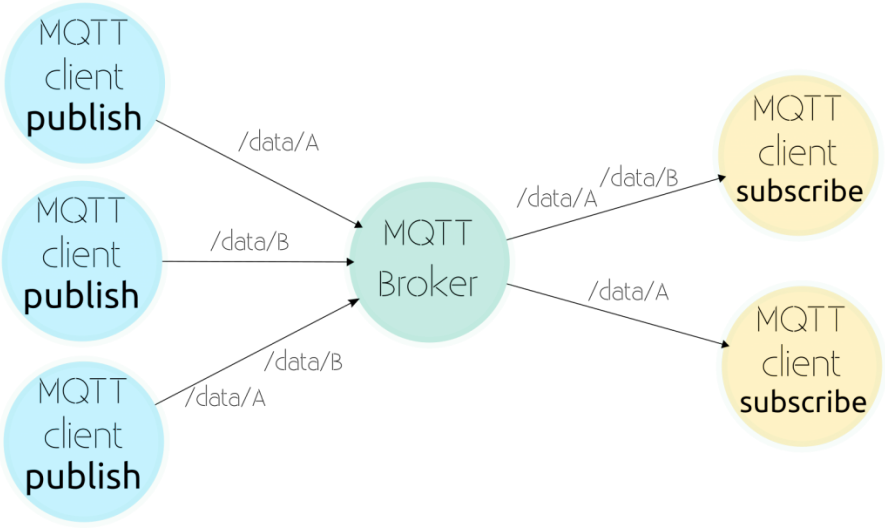

Figure 18: MQTT Architecture

Mosquitto is a free and open source broker used in this project.

3.6 Network Time Protocol

There are several ways to fetch the current date and time. One of the simplest and easiest solution is to use an NTP. It refers to Network Time Protocol. It is a protocol usually used by computers systems in order to provide them with synchronized clocks from the same reference. This common reference is UTC (Coordinated Universal Time).

The most popular architecture that can be used with NTP is the client-server architecture.

Below are listed the steps that describe how the NTP client communicates with the NTP server in order to receive the current date and time: (1) First, the NTP client connects to the NTP server through UDP (User Datagram Protocol) using the port 123. So, NTP doesn't wait for a properly established connection. (2) Then, the NTP client sends a request packet to the NTP server. (3) After, when the NTP server receives the client's request, it responds with a packet containing multiple information such as UNIX timestamp, accuracy, etc. (4) Finally, when the NTP client receives the response packet, it figures out the current date and time.

Regarding the NTP server, it can be any desired time server, but if "pool.ntp.org" is chosen, it will auto select the most time server geographically nearest to the NTP client.

It should be noted that the current timestamp could have been gotten by adding an additional chip to the prototype which is an RTC (Real time clock) chip. However, the RTC needs to be adjusted manually frequently and its response is not always accurate.

Global Architecture

Our demonstrator aims to provide a WSN testbed in a real SG environement. Collected data are provided by PV cells, electric vehicles, electrical equipment, etc. The communication is ensured via 120 sensor nodes deployed all over the buildings of the university. In the following subsections, we will briefly describe the network's hardware and software architectures.

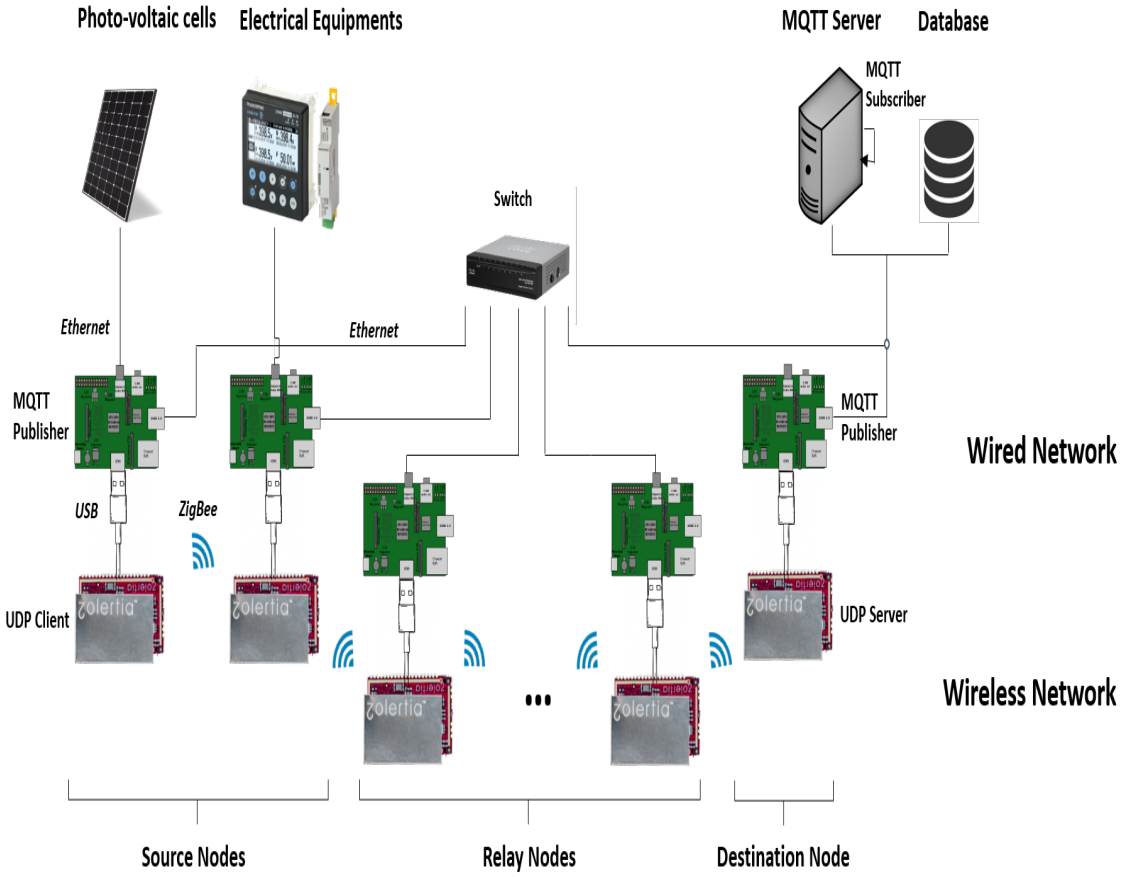

Figure 19: Global Architecture Overview

4.1 Hardware Architecture

The testbed consists of RPI-Zolertia sensor nodes, communicating over two types of networks (wired and wireless). In figure 19, the global architecture of our testbed and both wired and wireless networks are presented. The wired network serves as a controller to reprogram the wireless nodes, collect data from the different electrical equipment and store it in the database on the server. In a detailed view, a sensor node in our architecture consists of a RPI-Zolertia couple. It is a master-slave type of relation, where RPIs control and manage the wireless network, while Zolertia devices assume the role of processing and routing the received information from the different energy sources.

The wireless network is composed of three types of nodes: (1) *source*, (2) *relay* and (3) *destination nodes*. *Source nodes* are responsible of reading data from the RPIs, processing and transmitting it to a one hop relay node. *Relay nodes* receive the data from the *source nodes* and then take care of path computing, processing, and communicating to *destination nodes* across the wireless network. Finally, *destination nodes* process the received data packets to execute specific tasks (i.e., storing them in the database via the RPI). It is important to note that the Zolertia family is composed of different generations (e.g., Z1, Re-Mote, Firefly). In our demonstrator, we used Zolertia Re-Mote as a fast development tool for testing our IoT application. However, in the final network architecture, Firefly nodes [10], a more advanced generation, were the ones implemented.

4.2 Software Architecture

As already mentioned, our sensor node consists of a RPI-Zolertia couple. Firstly, RPI nodes are programmed using Python language. They serve first of all to reprogram/reflash the Zolertia nodes. Moreover, they communicate with the PV

cells and the electrical equipment using the Modbus TCP/IP Protocol [11]. It is used to establish a master-slave Ethernet communication between different devices. Each slave in the network is assigned a unique unit address and stores data as a numerical format in tables called registers. The different Modbus equipment are all connected to a measurement center (D-50 AGBT) via a Digiware Bus. Once the RPI node connects to that center, it chooses the corresponding slaves to address, as well as the desired registers to read from. In other words, the RPI node will choose the data type that it needs to get from the electrical equipment in order to transmit it afterwards to the Zolertia node. Its worth mentioning that we used ModbusDoctor, a real-time simulator.

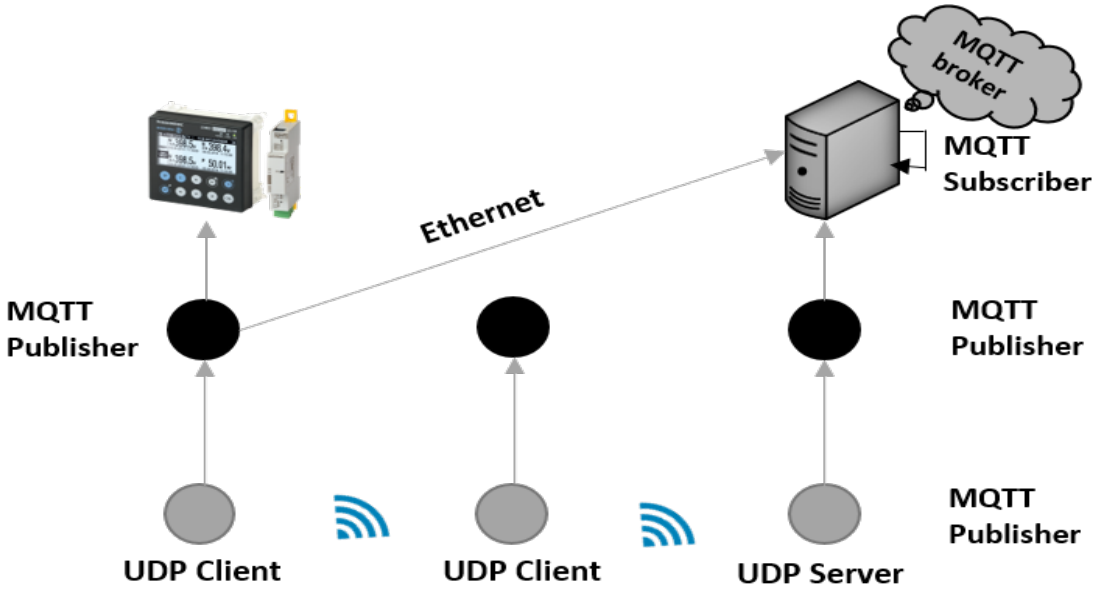

Figure 20: Software Architecture Overview

Next, and after the data collection from the electric equipment on the RPI level, these data are transmitted to the Zolertia nodes via a serial communication. These nodes are programmed to receive and transmit data in radio signals, taking into account their limited energy resources. Contiki [12], a largely used operating system for memory-constrained devices, is implemented on these sensor nodes,

thus enabling the connection to IPv4/IPv6. For that, devices are configured to meet the aforementioned standards and to reach every node in the network. This is done by implementing a 6LoWPAN [13] network. This protocol delegates all the capabilities of IPv6 on a constrained device. On the application layer, and in order to distribute the collected data more efficiently, Message Queue telemetry transport (MQTT) [14], a lightweight publish and subscribe protocol, is implemented on both the Zolertia and RPI nodes. A local broker running on the server side is responsible for handling the publishing and subscribing events. On the other side, each sensor node (wired and wireless) publishes the captured values in distinct topics. Data is then handled and inserted in the database by the subscriber.

Example of Use Cases

In order to test our network, a small use case scenario composed of three RPI-Zolertia sensor nodes (source, relay and destination) is studied. These nodes were deployed in a typical office environment in the university's building.
It it important to note that the current use case is just an example to show some of the metrics that can be evaluated/measured in our testbed. For that the comparison between the wireless and the wired networks is not intended to evaluate or to compare the performance of one or the other.

The network architecture implemented is similar to the one depicted in figure 19. On one hand, a reliable, non-intrusive and real-time system is implemented using RPI modules. They are connected all together by an SLM2008 Cisco switch, thus establishing a wired network. Thereafter, each module is given access to the database by authenticating to a web service. It then connects to the Modbus measurement center, and fetch the PV and electrical equipment's real-time values. Once fetched, these values are inserted via MQTT in the database. On the other hand, a wireless low-power network is also implemented, using the Zolertia Re-Mote sensors, where each one is connected to a RPI module. In this case, three scenarios take place: the first concerns the Zolertia *source node* connected to the RPI source. It retrieves the previously collected values (by the RPI from the PV cells and electrical equipment) and initiates the propagation process to the rest of the nodes by sending the data packets over the network. The second scenario illustrates the *relay node*, which mission consists of routing the packets from source to destination via the wireless medium. On the receiving edge, the *destination node* transmits the packet to the connected RPI, through serial cable, to be inserted in the database for a future use.

5.1 Sensor Deployment

For an efficient deployment of the Zolertia Re-Mote nodes in the building, the range-test application [4] was configured on two test sensors. The latter were both connected to LCD screens, in order to visualize the number of sent and received packets.

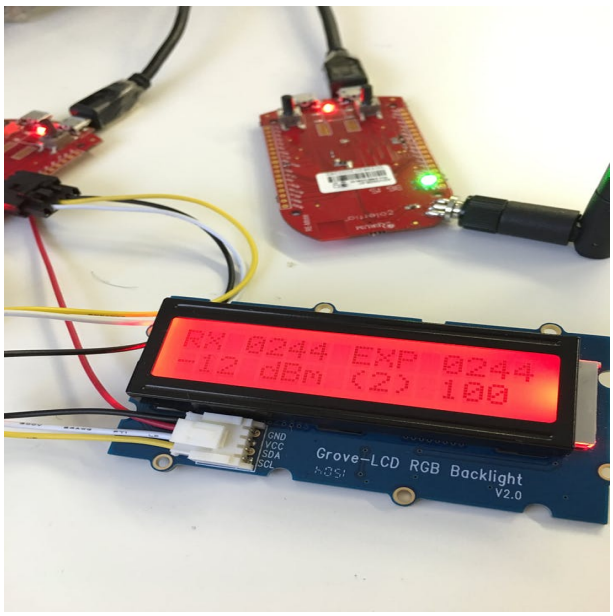

Figure 21: Range-Test Application

In the end, the analog quality of the signals were tested with the RF Explorer tool (more details about this module in section 3.3). Thereafter, the final sensor deployment schema was the one combining the optimal radio signal strength indication (RSSI), packet reception ratio (PRR), packet error rate (PER) and link quality indication (LQI).

5.2 Experiment Parameters

In order to evaluate our approach, the experiment was performed on Contiki OS using the RPI-Zolertia sensor nodes as already mentioned. Experimentation pa-

rameters are presented in Table 7. Tests were repeated three times for more accurate results. The topology consists of three sensor nodes placed at a distance of approximately 15 meters from each other. They are configured to transmit their radio signals on channel 26, on the 2.4 GHz frequency band. The client sends the collected measurements as UDP packets [15] every 5 seconds using the RPL routing protocol [16] on the network layer. A UDP server running on the destination node, sets up the UDP connection and waits for packets from clients.

Parameters	**Values**
Sensors	Zolertia Re-Mote
Operating System	Contiki
Communication Protocols	RPL, UDP, MQTT
Channel	26
Frequency	2.4 GHz
Distance between nodes	Approximately 15 meters
Number of nodes	1 client, 1 relay and 1 server
Sending interval	1 packet every 5 seconds

Table 7: Parameters Of The Experimentation

To guarantee the results reliability from the described scenarios, the RPI modules were all synchronized with a local Network Time Protocol (NTP) server.

5.3 Performance Evaluation

Figure 22 shows the percentage of packet loss during a period of 75 minutes. We notice that, in the wireless network, it is a non-stable curve, given the distinct signals that may interfere in the experiment's environment (e.g., WiFi). As expected, the wired network does not present any packet loss.

Figure 23 shows the resulted delay between the sent and received packets from the source to the destination node, in both the wired and wireless networks. We observe a delay ranging from 1.2 to 1.7 seconds in the wireless network. However, no delay (negligible) is recorded in the wired one.

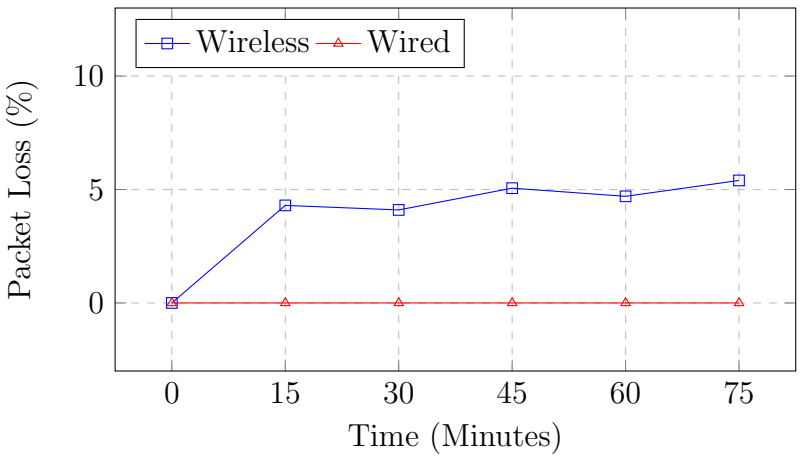

Figure 22: Packet Loss Percentage For Wireless And Wired Networks

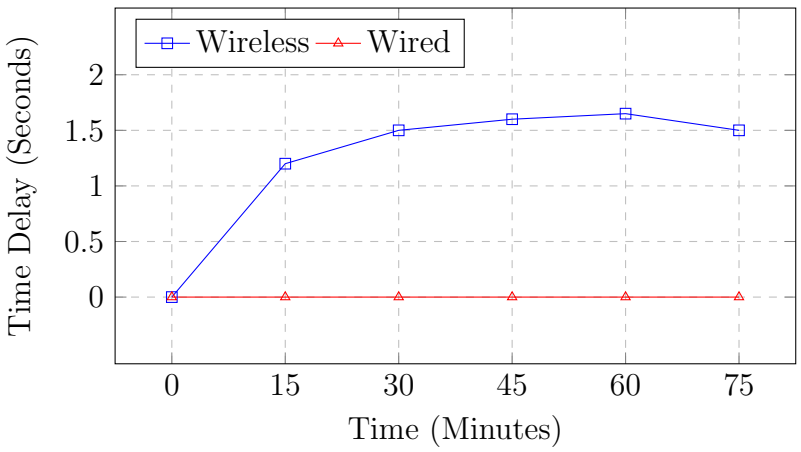

Figure 23: Time Delay For Wireless And Wired Networks

Conclusion, Actual State and Future Work

In this work, we introduced a SG WSN testbed that was developed in the SoMEL SoConnected project. It consists of a set of 120 RPI-Zolertia sensor nodes that are being deployed all over the building of the Catholic University of Lille and Yncréa Hauts de France. The purpose of developing this demonstrator is to monitor and control the energy production and consumption on campus via a WSN. This testbed is still under-deployment, and will be open to public access once fully implemented.

However, a live experiment was done using 3 sensor nodes (source, relay and destination), in order to evaluate our testbed. The results show a packet loss percentage (varying between 4% and 5% starting the minute 15 of the experiment) in the wireless network. Unlike the wired one where no packet loss percentage is noticeable. Another metric was studied in order to observe the time delay from the source to the destination. Results show a delay of approximately 1.5 seconds in the wireless network, and a negligible delay in the wired one.

In the near future, we will continue the physical deployment of the sensor nodes and the development of a dedicated website open to public in order to be able to launch experiments in a user friendly environment. This access will be done while ensuring the security of our equipment (i.e., developing the corresponding API to control the access). This network of sensors will make it possible to experiment, in real conditions (radio interference, crossing thick walls, loss of signal, etc.), specific self-adaptive communication protocols necessary for managing a Smart Grid such as the differentiated routing of control orders and/or production or electricity consumption measurements.

References

[1] N. Phuangpornpitak and S. Tia, "Opportunities and challenges of integrating renewable energy in smart grid system," *Energy Procedia*, vol. 34, pp. 282–290, 2013.

[2] S. Rekik, N. Baccour, M. Jmaiel, and K. Drira, "Wireless sensor network based smart grid communications: Challenges, protocol optimizations, and validation platforms," *Wireless Personal Communications*, vol. 95, no. 4, pp. 4025–4047, 2017.

[3] J. Nassar, M. Berthomé, J. Dubrulle, N. Gouvy, N. Mitton, and B. Quoitin, "Multiple instances qos routing in rpl: Application to smart grids," *Sensors*, vol. 18, no. 8, p. 2472, 2018.

[4] A. Lignan, "Range tests made easy with the re-mote and lcd," Tech. Rep., 2016. [Online]. Available: https://www.hackster.io/alinan/range-tests-made-easy-with-the-re-mote-and-lcd-6e78b3

[5] C. Adjih, E. Baccelli, E. Fleury, G. Harter, N. Mitton, T. Noel, R. Pissard-Gibollet, F. Saint-Marcel, G. Schreiner, J. Vandaele *et al.*, "Fit iot-lab: A large scale open experimental iot testbed," in *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*. IEEE, 2015, pp. 459–464.

[6] V. Handziski, A. Köpke, A. Willig, and A. Wolisz, "Twist: a scalable and reconfigurable testbed for wireless indoor experiments with sensor networks," in *Proceedings of the 2nd international workshop on Multi-hop ad hoc networks: from theory to reality*, 2006, pp. 63–70.

[7] M. Nati, A. Gluhak, H. Abangar, and W. Headley, "Smartcampus: A user-centric testbed for internet of things experimentation," in *2013 16th In-*

ternational Symposium on Wireless Personal Multimedia Communications (WPMC). IEEE, 2013, pp. 1–6.

[8] J. Albesa, R. Casas, M. T. Penella, and M. Gasulla, "Realnet: An environmental wsn testbed," in *2007 International Conference on Sensor Technologies and Applications (SENSORCOMM 2007)*. IEEE, 2007, pp. 502–507.

[9] J. Schaerer, Z. Zhao, J. Carrera, S. Zumbrunn, and T. Braun, "Sdnwisebed: A software-defined wsn testbed," in *International Conference on Ad-Hoc Networks and Wireless*. Springer, 2019, pp. 317–329.

[10] N. J. Mouri, "Iot protocols and security," *Information Technology*, 2019.

[11] A. Swales *et al.*, "Open modbus/tcp specification," *Schneider Electric*, vol. 29, 1999.

[12] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki-a lightweight and flexible operating system for tiny networked sensors," in *29th annual IEEE international conference on local computer networks*. IEEE, 2004, pp. 455–462.

[13] N. H. Kumar, S. Baskaran, S. Hariraj, and V. Krishnan, "An autonomous aquaponics system using 6lowpan based wsn," in *2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*. IEEE, 2016, pp. 125–132.

[14] U. Hunkeler, H. L. Truong, and A. Stanford-Clark, "Mqtt-s—a publish/subscribe protocol for wireless sensor networks," in *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE'08)*. IEEE, 2008, pp. 791–798.

[15] G. Oikonomou and I. Phillips, "Experiences from porting the contiki operating system to a popular hardware platform," in *2011 International Conference on*

Distributed Computing in Sensor Systems and Workshops (DCOSS). IEEE, 2011, pp. 1–6.

[16] T. Clausen, U. Herberg, and M. Philipp, "A critical evaluation of the ipv6 routing protocol for low power and lossy networks (rpl)," in *2011 IEEE 7th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. IEEE, 2011, pp. 365–372.

List of Figures

1 Example Of Sensor Deployment In One Floor In The University . . 3
2 Zolertia Sensor 5
3 Zolertia Re-Mote Buttons And Pins 7
4 Hello-World Example 11
5 Compilation Output 12
6 Error Display 13
7 Raspbian Buster Software 13
8 Putty Software 16
9 RF Explorer 18
10 Spectrum Analyzer Main Screen 20
11 Standard Mode 21
12 Advanced Mode 22
13 Two Types Of Antennas 22
14 Near Field Antennas 23
15 Semi-Flexible RF Cable 23
16 Client-Server Model 25
17 TCP/IP Message Format 26
18 MQTT Architecture 29
19 Global Architecture Overview 31
20 Software Architecture Overview 33
21 Range-Test Application 36
22 Packet Loss Percentage For Wireless And Wired Networks 38
23 Time Delay For Wireless And Wired Networks 39

List of Tables

1 Difference Between Zolertia And Firefly 8
2 Input Power Levels 19
3 ASCII Message Frame 24
4 RTU Message Format 25
5 Modbus Function Codes 27
6 Function Code With Exceptions 27
7 Parameters Of The Experimentation 37