

ISCTE  IUL
Instituto Universitário de Lisboa

Department of Information Science and Technology

**The use of Sensor Networks to
create smart environments**

André Filipe Xavier da Glória

A Dissertation presented in partial fulfillment of the Requirements
for the Degree of
Master in Telecommunications and Computer Engineering

Supervisor

Prof. Dr. Francisco António Bucho Cercas, Full Professor
ISCTE-IUL

Co-Supervisor

Prof. Dr. Nuno Manuel Branco Souto, Assistant Professor
ISCTE-IUL

June, 2017

Resumo

A *Internet of Things* está a atingir o mundo de modo a tornar-se a próxima grande revolução depois da Internet, com quase todos os objectos a estarem ligados para recolher dados e permitir o controlo através de dispositivos móveis. Mas esta revolução depara-se com vários desafios devido à falta de standards no que toca a comunicações ou sensores.

Nesta dissertação apresentamos uma proposta para um sistema dedicado a criar ambientes inteligentes usando redes de sensores, com uma aplicação prática desenvolvida para oferecer automação, eficiência e versatilidade, permitindo uma monitorização e controlo remoto seguro em tempo real de qualquer objecto ou ambiente, melhorando assim a experiência do utilizador e a eficiência das tarefas levando a redução de custos. O sistema desenvolvido, que inclui *software* e *hardware*, usa algoritmos adaptáveis com Inteligência Artificial e dispositivos IoT de baixo custo, utilizando os melhores protocolos de comunicação, permitindo que o mesmo seja apropriado e facilmente adaptado para qualquer especificação por qualquer pessoa.

Avaliamos os melhores métodos de comunicação e dispositivos necessários para a implementação e demonstramos como criar todos os nós da rede, incluindo a construção de IoT Gateway e Sensor Node personalizados. Demonstramos também a eficácia do sistema desenvolvido através da aplicação do mesmo em casos reais.

As principais contribuições do nosso estudo passam pelo desenho e implementação de uma nova arquitectura para projectos adaptáveis de IoT com foco na eficiência do objecto, incluindo a demonstração pratica, tal como um estudo comparativo sobre os melhores protocolos de comunicação para dispositivos IoT de baixo custo.

Palavras-chave: Internet of Things, Redes de Sensores, Ambientes Inteligentes, Comunicações, Sistemas Embebidos, Eficiência.

Abstract

Internet of Things is taking the world in order to be the next big thing since the Internet, with almost every object being connected to gather data and allow control through mobile and web devices. But this revolution has some barriers with the lack of standardization in communications or sensors.

In this dissertation we present a proposal of a system dedicated to creating smart environments using sensor networks, with a practical application developed to achieve automation, efficiency and versatility, allowing real-time monitoring and remote control of any object or environment improving user experience, tasks efficiency and leading to costs reduction. The developed system, that includes software and hardware, is based on adaptive and Artificial Intelligence algorithms and low cost IoT devices, taking advantage of the best communication protocols, allowing the developed system to be suited and easily adapted to any specification by any person.

We evaluate the best communication and devices for the desired implementation and demonstrate how to create all the network nodes, including the build of a custom IoT Gateway and Sensor Node. We also demonstrate the efficiency of the developed system in real case scenarios.

The main contributions of our study are the design and implementation of a novel architecture for adaptive IoT projects focus on environment efficiency, with practical demonstration, as well as comparison study for the best suited communication protocols for low cost IoT devices.

Keywords: Internet of Things, Sensor Networks, Smart Environments, Communications, Embedded Systems, Efficiency.

Acknowledgements

I would like to thank my advisers Professor Francisco Cercas and Professor Nuno Souto for accepting my challenge to do an out of the box project, for all the knowledge and inspiration to complete this dissertation.

Although not part of the project, a special thanks is necessary for Professor Pedro Sebastião for all the help given throughout the time, not only in academic but also in professional and personal matters.

To my family, especially my parents, Sandra and Paulo, that always done the possible and the impossible for providing me the opportunity to reach my goals, for the support and understanding my choices. Without them it was not possible to get all these achievements and for that I will forever be grateful.

To my good friends André Marques and Daniel Fernandes, who for 5 years accompanied me in the various challenges that brought us where we are today.

Also to my friends António Raimundo, José Serro, Manuel Oliveira and Pedro Romano, that in some way help me through this project, and to the rest of my friends and colleagues.

Last, and because there is always that one person that influence you to reach for the stars, an exceptional thanks for my 1st grade Professor Fandy. Thanks for teaching that little kid that used to bring toy cars to play under his desk during class time how to read and do maths, and for always remind me not to give up and that grades does not define who you are. Although you are not here to see it, I know that you would be proud of the outcome.

Contents

Resumo	iii
Abstract	v
Acknowledgements	vii
List of Figures	xiii
List of Tables	xv
Abbreviations	xvii
1 Introduction	1
1.1 Objectives	2
1.2 Scientific Contribution	3
1.3 Structure of the Dissertation	4
2 State of the Art	7
2.1 Internet of Things	7
2.2 Sensor Networks	9
2.3 Communication protocols	10
2.3.1 Wired Network Technologies	11
2.3.1.1 Serial Peripheral Interface	11
2.3.1.2 Inter-Integrated Circuit	12
2.3.1.3 Universal Asynchronous Receiver and Transmitter	13
2.3.1.4 Power Line Communication	14
2.3.2 Wireless Network Technologies	14
2.3.2.1 IEEE 802.11 (Wi-Fi)	15
2.3.2.2 IEEE 802.15.1 (Bluetooth)	16
2.3.2.3 IEEE 802.15.4 (ZigBee)	17
2.3.2.4 LoRaWAN	18
2.3.3 Internet	19
2.3.3.1 Message Queuing Telemetry Transport	20
2.3.3.2 Constrained Application Protocol	21
2.3.4 Remarks	22
2.4 Controlling Platforms	22

2.4.1	Arduino	23
2.4.2	ESP8266	24
2.4.3	Raspberry Pi	25
2.4.4	BeagleBone	26
2.4.5	Intel Edison	26
2.4.6	Remarks	27
2.5	Related Work	28
2.5.1	IoT Gateway	28
2.5.2	Visualization Platforms	30
3	Communications Tests	31
3.1	Server side communications	31
3.1.1	Test Scenario	32
3.1.2	Results	32
3.2	Network side communications	35
3.2.1	Setup Complexity	36
3.2.2	Test Scenario	38
3.2.3	Results	39
3.3	Discussion	44
4	System Architecture	45
4.1	Software	46
4.1.1	Visualization Platforms	46
4.1.1.1	Monitoring and Control	48
4.1.1.2	Rules	49
4.1.1.3	Notifications	50
4.1.1.4	Personal Area	51
4.1.2	Support Scripts	51
4.1.2.1	Visualization Platform	52
4.1.2.2	Server	53
4.1.2.3	Aggregation Node	53
4.1.2.4	Sensor Node	55
4.1.3	Network Security	56
4.2	Hardware	57
4.2.1	Sensor Node	58
4.2.2	IoT Gateway	59
5	Application Scenarios	65
5.1	Implemented Environments	66
5.1.1	Solar Panel Control	66
5.1.1.1	Results	68
5.1.2	Class Room Control	69
5.1.2.1	Results	71
5.1.3	Vertical Garden	72
5.1.4	ISCTE Satellite Station	73

List of Figures

2.1	IoT elements	8
2.2	Sensor network nodes	9
2.3	SPI bus topologies	12
2.4	I ² C connection between devices	13
2.5	IBSS and ESS configurations of Wi-Fi networks.	16
2.6	Bluetooth topology	17
2.7	IEEE 802.15.4 topologies	18
2.8	LoRaWAN topology	19
2.9	MQTT Publish/Subscribe process	20
2.10	CoAP message Types	21
2.11	Arduino boards	23
2.12	ESP8266 components	24
2.13	Raspberry Pi boards	25
2.14	BeagleBone Black	26
2.15	Intel Edison module and Kit for Arduino	27
3.1	Server side communication test scenario	33
3.2	HTTP POST Wireshark packet capture	33
3.3	MQTT QoS 0 Wireshark packet capture	33
3.4	MQTT QoS 2 Wireshark packet capture	34
3.5	Bandwidth consumption comparison	34
3.6	Transaction RTT comparison	34
3.7	Arduino Wireless Protoshield with XBee S2	37
3.8	XBee Explorer	37
3.9	Adafruit RFM69HCW	37
3.10	Wired test example	38
3.11	Test Scenario Model	38
3.12	Test scenario grounds	39
3.13	Wireless Test example	39
3.14	I ² C Delay vs Distance results	40
3.15	RS232 Delay vs Distance results	40
3.16	ZigBee Delay vs Distance results	41
3.17	LoRa Delay vs Distance results	41
3.18	I ² C Throughput vs Distance results	42
3.19	RS232 Throughput vs Distance results	42

3.20	ZigBee Throughput vs Distance results	43
3.21	LoRa Throughput vs Distance results	43
4.1	System Architecture	46
4.2	Database relational design	47
4.3	Screenshot of the web platform	47
4.4	Top section with sensor and actuator information	48
4.5	Bottom section with actuator schedule	49
4.6	Bottom section with sensor information	49
4.7	General Information Cell	49
4.8	Bottom section with rules information	50
4.9	Rules Bootstrap Modal	50
4.10	Notification Bootstrap Modal	51
4.11	Screenshot of the personal area	51
4.12	Visualization Platform Tasks pending MQTT Message Received	52
4.13	Scheduled Actions Management	53
4.14	Value and rules process on aggregation node	54
4.15	Network and User authentication process and initial process	55
4.16	Data and Actions exchange	56
4.17	Breaduino implementation	58
4.18	IoT Gateway first prototype	60
4.19	RS232 network connection	60
4.20	LoRa network connection	61
4.21	IoT Gateway second prototype	61
4.22	IoT Gateway PCB implementation	63
4.23	IoT Gateway with Wireless Communication	63
5.1	Solar node implementation	67
5.2	Solar Panel Results	68
5.3	Room Control scenario	70
5.4	Motion sensor view	70
5.5	Room Control result	71
5.6	Vertical Garden implementation	72
5.7	ISCTE Satellite Station Architecture	73
5.8	ISCTE Satellite Station Hardware	74
5.9	CPU Usage comparison between systems	75
5.10	Bandwidth Usage comparison between systems	75
A.1	I2C connections	85
A.2	RS232 connections	86
A.3	LoRa connections	86
B.1	Sensor Node Connections	87
B.2	IoT Gateway Connections	88
B.3	IoT Gateway PCB Projection	88

List of Tables

2.1	Major Wired communication protocols characteristics	11
2.2	Major Wireless communication protocols characteristics	15
2.3	Major IoT devices characteristics	28

Abbreviations

ACK	A cknowledgement
AP	A ccess P oint
API	A pplication P rogramming I nterface
BSS	B asic S ervice S et
CoAP	C onstrained A pplication P rotocol
CSMA/CD	C arrier S ense M ultiple A ccess with C ollision D etection
CSS	C hirp S pread S pectrum
DSSS	D irect S equence S pread S pectrum
FSK	F requency- S hift K eying
IBSS	I ndependent B asic S ervice S et
IDE	I ntegrated D evelopment E nvironment
IEEE	I nstitute of E lectrical and E lectronics E ngineers
I²C	I nter- I ntegrated C ircuit
I/O	I nput/ O utput
IoT	I nternet o f T hings
LAN	L ocal A rea N etwork
LR-WPAN	L ow- R ate W ireless P ersonal A rea N etwork
M2M	M achine-to(2)- M achine
MQTT	M essage Q ueuing T elemetry T ransport
OTA	O ver T he A ir
PAN	P ersonal A rea N etwork
PLC	P ower L ine C ommunication
QoS	Q uality o f S ervice
RTC	R eal T ime C lock

RTT	R ound- T rip T ime
SDK	S oftware D evelopment K it
SoC	S ystem o n C hip
SPI	S erial P eripheral I nterface
TCP	T ransmission C ontrol P rotocol
UART	U niversal A synchronous R eceiver and T ransmitter
UDP	U ser D atagram P rotocol
VPS	V irtual P rivate S erver
WLAN	W ireless L ocal A rea N etwork
WPAN	W ireless P ersonal A rea N etwork
WSN	W ireless S ensor N etwork N etwork

Chapter 1

Introduction

Nowadays the urge to connect everything to the Internet is growing, not just to send information to servers for processing and storage but also to provide full control of physical devices over the web. While humans will continue to connect their devices to the Web in greater numbers, by 2020 more than 200 billion smart devices are expected to be connected to the Internet, making Machine-to-Machine (M2M) communications up to 45% of the whole Internet traffic [1, 2, 3].

Examples such as Smart Homes, where users can control their thermostats or lights with a smartphone, are the basis for Internet of Things (IoT). IoT was designed to play a great role improving our quality of life and its applications are present in many of our day to day experience such as transportation, health care and industrial automation.

IoT has the ability to transform a simple physical device into a smart one, using the embedded technology and computational power. With the sensors and actuators available to guarantee the features of the device, it is possible to share information between devices and to put them to work together to improve the user experience. This will contribute to a bigger explosion coming from “things” connected to the Web that were not connected before, or didn’t exist, and now use their connection as a core feature.

People living in a smart world will be served by smart devices and IoT projects have the ability to do more than just connect the device to the Internet, they can be a big part of improving the efficiency or even adding new features such as Artificial Intelligence, transforming every common objects into connected ones, keeping track of people's cyber, physical and social aspects interconnecting them and intelligently creating a better experience.

But for every product of different companies there is an app or standard that can not be used outside those products. For example, in the smarthomes business there are hundreds of smart bulbs available and for each one there is an app, some of them paid, or a communication standard that does not work on or in connection with other devices, offering to the average user, that have multiple products from different manufactures, little or no flexibility, reduced scalability, high complexity and prices. Beside all the solutions available, that cover almost every topic, there is not a solution that singly gives all the information and allows full control.

1.1 Objectives

The goal for this project is to create an IoT system that can be applied to any object or environment, with little or none modifications, offering the user an easy full remote control over objects. With this project it will be possible for objects that require manual control and that can only be accessed locally, to have fully remote control via an online platform.

The system, which will be built from the ground up, will be developed in order to be based on three critical areas: automation, efficiency and versatility.

In the automation area, it will allow the user to have full remote and secure control and monitoring of sensor networks, being able to see current and past data from sensors and allow control over actuators, all over a web platform, anywhere in the world. The efficiency area is critical, because automation does not lead to efficiency, e.g. an automatic irrigation system will turn on even if it is raining, so with the use of rules set by the user and Artificial Intelligence algorithms, that

see if the actions are really needed at that moment, it allows a more effective and cost saving functionality, leading to potential gains, such as energy or water savings. Last but not least, the versatility area will allow the system to adapt to any non-smart object or environment and user specifications with a simple and easy process.

To achieve this goal, it is necessary to develop a set of specific modules that, when connected, create a system that can be applied to fulfill any requirements. For each module there is a set of target features:

1. Visualization Platform: Development of an Online Platform that connects the user to the sensor network, allowing data visualization and control over the network;
2. Sensor Network: Design and specification, including the study of the best architecture, to each node incorporated in the system;
3. Communications: Evaluation of the best communication methods, using wired or wireless techniques, to connect the several nodes in the system (server – aggregation node; aggregation node – sensors).

Although the main goal is to create a low cost system that is flexible in the sense that it can be easily adapted to any specification, the practical functionality will be tested and evaluated in a solar panel, garden and room control applications, through previously determined specifications.

1.2 Scientific Contribution

This dissertation presents the following contributions:

- Reviews and evaluates which communication schemes to use in order to ensure maximum flexibility, availability, energy efficiency and implementation simplicity;

- Introduce a new solution to sustain and control a full intelligent system using embedded platforms and sensor networks;
- Demonstrates the successful application of the system on real case studies;

The results obtained within the development of this project result into two scientific articles published in international telecommunications conferences, thus contributing to its dissemination to the scientific community.

- A. Glória, F. Cercas and N. Souto, “Design and implementation of an IoT gateway to create smart environments”, The 8th International Conference on Ambient Systems, Networks and Technologies (ANT 2017), 2017;
- A. Glória, F. Cercas and N. Souto, “Comparison of Communication Protocols for Low Cost Internet of Things Devices”, The 2nd International South-East European Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM 2017), 2017;

Also, during our research the project was presented in several technological events, resulting in four keynote presentations and one demonstration:

- "The use of Wireless Sensor Networks to create smart environments" - Gen-uino Day Lisbon 2016 & IoT Summit 2016 & 22nd RTCM Meeting;
- "Smart Environments using Raspberry Pi" - Raspberry JAM (November 2016);
- "Create Smart Environments using Arduino" - Arduino Day Lisbon 2017.

1.3 Structure of the Dissertation

In Chapter 2, we review various approaches and solutions studied for creating smart environments using sensor networks. In Chapter 3, we discuss the results of

our communication and devices research. In Chapter 4, we present our developed system architecture, with details on how to design and develop all the nodes. In Chapter 5, we experiment the solution created in real case scenarios and discuss the implementation results. Finally, in Chapter 6, we discuss the overall result of the dissertation and how this project can be improved.

Chapter 2

State of the Art

While IoT is becoming a day-by-day experience for everybody, it has proven challenging to create a single solution that can work in any object or environment allowing the user to fully control it. In this chapter, we review the various approaches that have been studied for creating smart environments using sensor networks. In Section 2.1, we start by comprehending how IoT is organized. In Section 2.2, we define what is a sensor network and how we can use it. In Section 2.3, we focus on the communication techniques used in our experiments. In Section 2.4, we compare the controllers. Finally, in Section 2.5, we discuss the main challenges faced by researchers in the mixed fields of IoT, sensor networks and smart environments, what solutions have been proposed, and how our approach differs.

2.1 Internet of Things

The Internet is one of the most important development of mankind and IoT will represent the next evolution of the Internet [2, 3]. With the capability of gathering, analyzing and distributing the data, IoT consists in the connection between the Internet and a range of devices and sensors.

IoT, as shown in Figure 2.1, can be divided in six elements [4] that help us understanding its real meaning and functionality, i.e., *identification*, *sensing*, *communication*, *computing*, *services* and *semantics*.

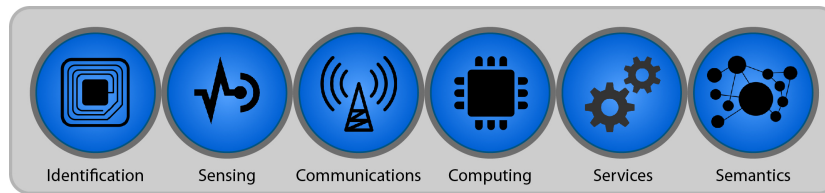


FIGURE 2.1: IoT elements

Identification is needed in IoT to name and match services with their demand. IoT objects need to be addressed so it is critical to know the difference between object ID, such as “Lum1” for a particular luminosity sensor, and its address within the network.

Sensing is the ability to gather data from the environment and sending it to be analyzed. Data can be temperature, luminosity or anything that can be measured.

Communication connects objects to trade data and perform specific services. For IoT, the most usual are Wireless communication protocols but Wired protocols are also used.

Computing represents the brain and computational ability of IoT, using hardware processing (e.g, microcontrollers, microprocessors, system on chips (SoCs)) and software applications to perform tasks.

Services in IoT are divided in four categories: identify-related services, that lay the ground to others services since every real world object needs to be identified in the virtual world; information aggregation services, that gather and summarize the raw information which needs to be processed; collaborative services, that uses the obtained data to make decisions; and ubiquitous services, that offer the collaborative-aware services to anyone on demand, anytime and anywhere.

Semantics is the ability to collect knowledge intelligently so as to provide the required services, including discovering resources, utilizing resources, modeling information, recognizing and analyzing data.

There are still some barriers [2] that can prevent the fast development of IoT such as the introduction of IPv6, with the need of sensors to have a unique address and the shortage of IPv4 address; the need of common standards, especially in the areas of security and communications; and power consumption, as the sensors require power it will be necessary to create energy from the environment. But given the potential of IoT, it is only a matter of time for resolving these issues.

2.2 Sensor Networks

IoT relies on the ability of gathering, transporting and processing data so the need of sensor networks its crucial. These networks can be any device equipped with sensors, from smartphones and cars to hobbyist microcontrollers such as Arduinos.

Sensor Networks [5] are composed by a number of sensing nodes communicating in a multi-hop fashion. As presented in Figure 2.2, there are three types of sensing nodes [6]:

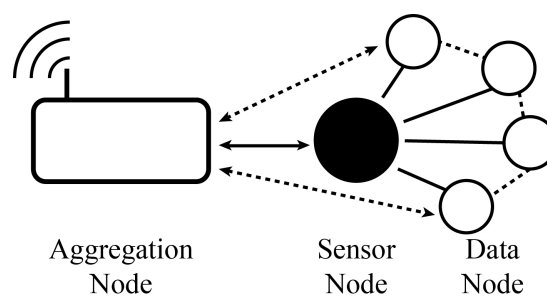


FIGURE 2.2: Sensor network nodes

1. Sensor Node: The lowest level of a Sensor Network is simply composed by a sensor and a communication mechanism. They gather the information and pass it to another node.

2. Data Node: Data nodes are sensor nodes that store data besides sending it to another one. Data storage can be used to perform autonomous tasks or keep track of information when communications are down. These nodes require additional devices with more capabilities, such as storage, where SD cards or databases are usually used.
3. Aggregation Node: These nodes don't read sensors, they typically have a communication device and a gateway to the server. They're used to collect data from one or more sensor nodes and send it to the servers.

Sensor Networks are design to be energy efficient, because energy is the hardest resource to find in the locations where the network is installed; scalable, given the high number of nodes that a network can have; reliable, in some situations the network can send warnings in case something is wrong; and robust, since sensor nodes can be exposed to failures or environmental hazards [5].

2.3 Communication protocols

As mentioned before, communication is one of the main elements of IoT, as the need to trade data between devices is crucial to the efficiency of an IoT project. Besides the word Internet being part of IoT, not all the situations rely on Internet related protocols. With the increased use of Sensor Networks and applications in the most diverse environments, the need of different protocols, both wired and wireless, is growing.

In this section we will discuss and compare, in detail, the various protocols including the major features of each one, in order to choose the best one for each point of communication.

2.3.1 Wired Network Technologies

Wired protocols are still used to connect devices in places with none or little Internet access, since they are more reliable, secure and can transfer data at higher rates. The most common wired technologies are Serial Communication or UART, mainly in the form of USB, RS232 or RS485, and also SPI, I²C and PLC. Beside these, CAN bus can be used for networks that need a higher reliability.

Table 2.1 shows some key characteristics of the most common wired protocols. A more detailed description, as well as advantages and disadvantages, of each one can be found in the following subsections.

TABLE 2.1: Major Wired communication protocols characteristics

Feature	SPI	I ² C	RS232	RS485	PLC	CAN
Data Rate [Mbps]	20	3.4	0.02	10	14	1
Range [m]	100	10	15	60	3000	40
Nodes	3	1024	256	256	x	127
Complexity	Medium	Low	Low	Low	Medium	High

2.3.1.1 Serial Peripheral Interface

SPI was introduced by Motorola in 1979, being a protocol designed for communication between integrated circuits and slow communication with on-board peripherals [7]. It uses four wires to connect to microcontroller peripherals, each one associated to a signal line, as shown in Figure 2.3:

- SCLK: a clock signal sent from the bus master to all slaves. This synchronizes all SPI signals;
- SS_n: a slave select signal for each slave, used by the master to select the slave to communicate;
- MOSI: a data line from master to slave;
- MISO: a data line from slave to master.

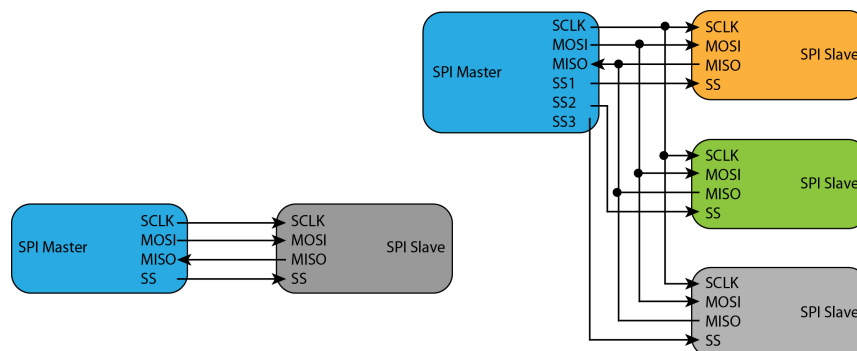


FIGURE 2.3: SPI bus topologies [7]

As a single master communication protocol, SPI has one central device that starts all the communications with the slaves. To send data to a specific slave, the master pulls the corresponding SS line low, activating a clock signal in a frequency usable for both. The MOSI line is used by the master to put information while sampling the MISO line to get information from slaves.

The SPI protocol does not define a maximum data rate or addressing scheme and does not have an acknowledgment mechanism. I/O voltages and standard used by the devices are not important for the SPI protocol [8].

Advantages

- Offers a variety of data rates;
- Cost-effective;
- Low power consumption;

Disadvantages

- Configuration done manually;
- Limited number of nodes;
- Require extra hardware for each slave

2.3.1.2 Inter-Integrated Circuit

I²C was developed in 1982 by Philips with the purpose of easily connecting a CPU to a peripheral chip in a television [7]. As SPI, this protocol was design for communication between integrated circuits and slow communication with on-board peripherals.

In order to reduce the number of wiring printed in the PCB and avoid additional logic, I²C uses just two wires for connecting the peripherals to the microcontroller, called Serial Data (SDA) and Serial Clock (SCL) [8]. In this two lines it is possible to connect multiple masters and slaves, making I²C a multi-master

protocol. For this, the protocol defines an unique 7-bit slave address for each device, used to connect to the bus, 8 bytes containing data and a few bytes for the communication control. As in Figure 2.4, besides this two lines, physically, the bus also needs a ground connection. Both active lines are bi-directional making the device that starts the data transfer on the bus a master, being the other slaves.

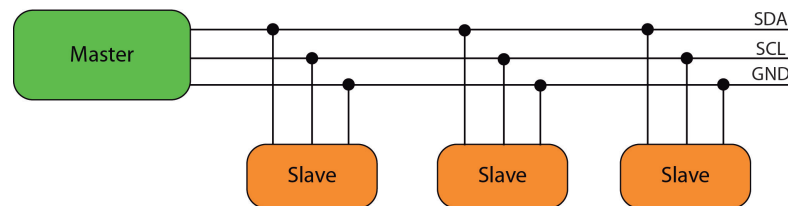


FIGURE 2.4: I²C connection between devices

I²C has a range of data rates [7] in which we can choose to transmit. *Standart mode*, at 100 kb/s, *fast mode*, at 400 kb/s, and *high speed mode*, at 3.4 Mb/s, are the main data rates for I²C. But some variants of the protocol add *low speed mode*, at 10 kb/s, and *fast mode +*, at 1 Mb/s, as valid speeds.

Advantages	Disadvantages
<ul style="list-style-type: none"> - Easy to connect - Widely supported - Automatically configured - Low power consumption 	<ul style="list-style-type: none"> - Does not support long distance or High Speed communication - Number of nodes is limited by the address space on the bus

2.3.1.3 Universal Asynchronous Receiver and Transmitter

The UART is a programmable integrated circuit capable of interfacing serial devices [9]. This interface provide operations such as converting the bytes to a single serial bit stream for outbound transmission, taking bytes of data and transmits them in a sequential fashion of individual bits, that will then be re-assembled in the destination by a second UART, adding a start and stop bits to signal the beginning and end of a data word, and a parity bit for error detection, with no need for a clock signal since UART is asynchronous but with both ends of the line operating with the same baud rate [9, 10, 11].

There are four different standarts for UARTs: RS232, RS423, RS422 and RS485. The RS232 has a Simplex or Full Duplex operation, being able to achieve

up to 15 meters at 20 kbits/s; as for the RS485 has an Half-Duplex differential capable of achieving up to 60 meters at 10Mbits/s [10].

Advantages	Disadvantages
- Widely supported - No clock signal needed - Robust to errors	- Limited size of 9 bits

2.3.1.4 Power Line Communication

PLC is a technology for carrying data as well as electric power on power lines. It does not require a separate communication line, since power lines are established widely, giving the devices an easy connection simply through plugging the power cord into an electrical outlet [12]. The use of power line cables for transporting data is possible with the application of certain modulation techniques that allow high speed communication with minimal interference [8]. Typically PLC devices operate using X10 protocol, that modulates a digital signals in a carrier wave of 120 kHz into the electric wiring at the transmitter, during zero crossings. To send a command to a receiver, each is assign an address within the system, and first the address is sent followed by the command, all through signals transmitted over the household wiring and decoded at the receiver [13].

Advantages	Disadvantages
- Preexisting connections allows low cost setup with no additional hardware; - High Data transfer;	- Data signals are susceptible to attenuation and to electromagnetic interference;

2.3.2 Wireless Network Technologies

Wireless communication protocols, offer all what is needed to work with the advantage of a wireless low space environment. The most common wireless technologies are Wi-Fi, ZigBee, Bluetooth, with new technologies like 6LoWPAN, LoRa and even LTE-A being increasingly used.

Present in Table 2.2 are some key characteristics of the most common wireless protocols. A more detailed description, as well as advantages and disadvantages, of each one can be found in the following subsections.

TABLE 2.2: Major Wireless communication protocols characteristics

Feature	Wi-Fi	Bluetooth	ZigBee	LoRa
Based Data Rate [Mbps]	11	1	0.25	0.011
Carrier Frequency [GHz]	2.45	2.45	2.45	0.433
Range [m]	1-100	10	10-100	20000
Nodes	32	7	65540	-
Power Consumption [mA]	100-350	1-35	1-10	1-10
Complexity	High	Medium	Low	Medium
Security	WPA/WPA2	128 bit	128 bit	128 bit

2.3.2.1 IEEE 802.11 (Wi-Fi)

Wi-Fi, a standard which conforms to IEEE 802.11, is a bidirectional radio frequency wireless protocol with the aim to connect devices inside the wireless local area network (WLAN). Approved in 1997, the IEEE 802.11 has seen some changes since the initial signal rate of 1Mbps, with standard b and g getting up to 11 and 54Mbps at 2,4GHz, respectively, the introduction of 5GHz on Europe with standard h and speeds up to 1Gbps at 5GHz in standard ac [14]. Also standards i and n brought new functionalities such as new encryption and MIMO physical layer.

The IEEE 802.11 architecture is composed by a number of components that interact providing a WLAN that can support station mobility, using a Basic Service Set (BSS), making devices capable of communicating directly within BSS range. Based on these, the IEEE 802.11 is also capable of implementing independent BSS (IBSS) where devices can connect directly without an access point (AP), an operation called ad hoc network, since the LAN is formed without pre-planning and is only active for the time needed. Also an extended form of network built

with multiple BSSs can be created, with multiple APs controlled by a distribution system (DS) being used to interconnect the BSSs, allowing the creation of an ESS network with arbitrary size and complexity.

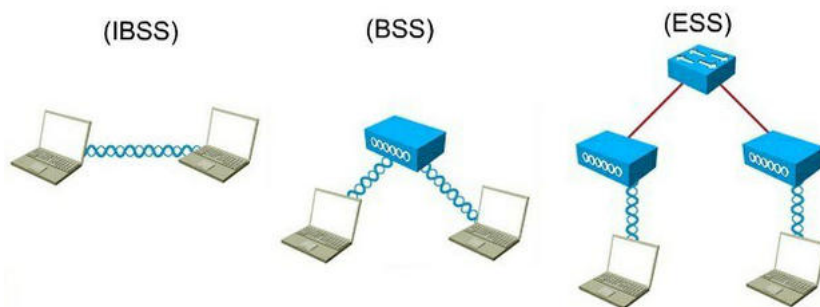


FIGURE 2.5: IBSS and ESS configurations of Wi-Fi networks

Communication over Wi-Fi inside a WLAN mostly occur with TCP/IP packets, being each device identified by its IP or MAC address.

Advantages	Disadvantages
<ul style="list-style-type: none"> - Highly secure connection, with 128-bit AES encryption; - Supports networking over power lines, coaxial cables and phone lines; 	<ul style="list-style-type: none"> - Consumes a significant amount of power;

2.3.2.2 IEEE 802.15.1 (Bluetooth)

Introduced in 1994 by Ericsson Mobile Communications, Bluetooth is a standard for wireless communication based on short-range radio system (WPAN). It was approved in 2005 as an IEEE standard, the 802.15.1 [14].

Bluetooth defines not only the radio interface but also the communication stack that allows devices to find each other. A Bluetooth device can operate either as a master or a slave, up to 7 nodes plus the master, in a star topology, as shown in Figure 2.6. Masters and slaves can switch roles in order to participate and interconnect with other networks, being slaves only capable to communicate with their master in a point-to-point fashion, starting by listening for an already running master and letting him know his address, whereas the master's transmissions may be either point-to-point or point-to-multipoint, that also defines the unique

frequency sequence and synchronize the network with its clock [14, 15]. In order to reduce power consumption a slave can be in the parked or stand-by modes.

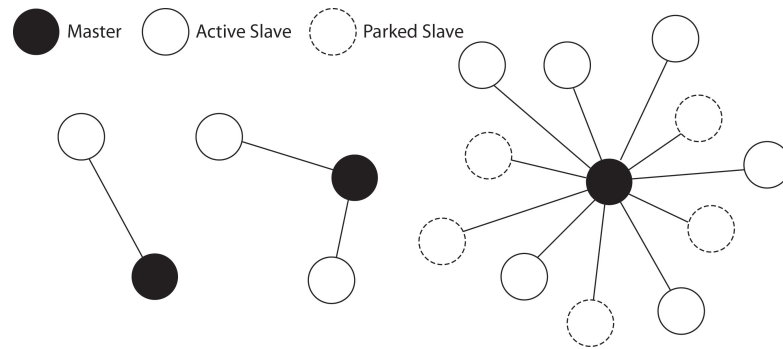


FIGURE 2.6: Bluetooth topology

Bluetooth is currently in version 4.0 or Bluetooth Low Energy, a short range radio with minimal amount of power to operate for longer periods of time compared with previous versions. Besides the low power consumption, the main feature of BLE is that have 10 times more range than normal Bluetooth, about 100 meters, making BLE a good candidate for IoT [4]. Some tests are already made for version 5.0, and previous versions allow several types of connection, with different combinations of available bandwidth, error protection and quality of service.

Advantages	Disadvantages
<ul style="list-style-type: none"> - One coordinator can control a numerous amount of slaves; - Widely supported; - Self-organizing network; 	<ul style="list-style-type: none"> - Pairing all the network can be a bit complex; - Only has star topology; - Limited number of nodes;

2.3.2.3 IEEE 802.15.4 (ZigBee)

Introduced in 2002, ZigBee implements the IEEE 802.15.4 protocol. Created for low-rate wireless private areas networks (LR-WPAN) it is one of the most used communication protocols for IoT and WSNs due to its low power consumption, low data rate, low cost and high message throughput. It can also provide high reliability, security, with both encryption and authentication services, working with different platforms and handling up to 65000 nodes [4, 8]. The IEEE 802.15.4 works in three frequencies, with a DSSS method, capable of transferring data at

250 kbps at 2.4 GHz, the most usual. To ensure that collisions are avoid, it uses the CSMA/CA protocol [4].

This standard can support two types of network nodes, Full Function Device (FFD) and Reduced Function Device (RFD). The FFD can serve as a PAN coordinator, a coordinator or just a normal node, and it is responsible for creating, controlling and maintain the network. They also store the routing table, being able to connect with every node using all network topologies. The RFD are simple nodes with low resources, being just able to communicate with the coordinator in a star topology [4, 15]. ZigBee topologies, as shown in Figure 2.7, are formed using a combination of specific nodes.

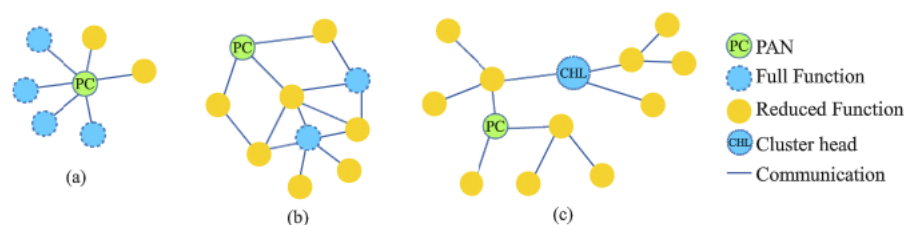


FIGURE 2.7: IEEE 802.15.4 topologies. (a) Star, (b) Peer-to-Peer, (c) Cluster-tree

Advantages	Disadvantages
<ul style="list-style-type: none"> - Low power consumption; - One coordinator can control a numerous amount of slaves - Self-organizing network capabilities - Highly secure, with 128-bit AES encryption 	<ul style="list-style-type: none"> - Requires additional hardware; - Low packet size, only 127 bytes per message (250 kbps) - Is incompatible with other network protocols and lacks Internet Protocol support

2.3.2.4 LoRaWAN

LoRaWAN is a bidirectional communication protocol that uses the LoRa physical layer in order to provide low power long-range communications. To achieve this, LoRa is based on CSS modulation that have the same low power characteristics as FSK modulation (present in great part of the other wireless communication protocols) but with a significant increase in the communication range. With a single base station it is possible to cover up to hundreds of square kilometers [16].

LoRaWAN uses a star topology in order to maintain the low power long communication viable, reducing complexity and increasing network capacity and lifetime as opposed by a mesh topology. This is possible by using an adaptive data rate and multichannel multi-modem transceiver in the gateway, providing the capability to receive simultaneous messages [16].

As presented in Figure 2.8, the nodes are not associated with a specific gateway. As said the data is transmitted to multiple gateways, that then forward the received packets from the end-node to the network server, that will then filter the redundant packets, do security checks and tell a specific gateway to send the acknowledgment [16].

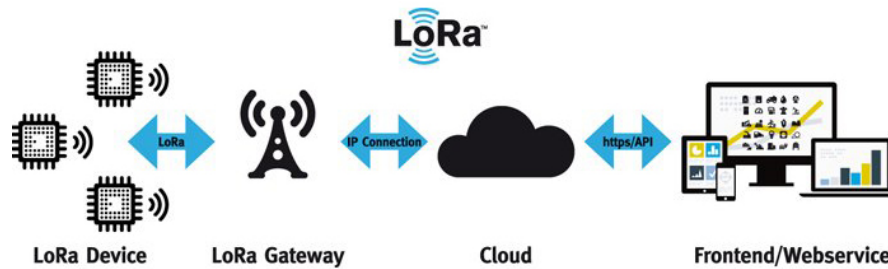


FIGURE 2.8: LoRaWAN topology

Advantages	Disadvantages
<ul style="list-style-type: none"> - Low power consumption; - Long communication range - Highly secure, with 128-bit AES encryption 	<ul style="list-style-type: none"> - Low packet size, only 55 bytes per message (11 kbps);

2.3.3 Internet

In order to transfer the data from the sensors to the servers, where they will be stored and analyzed, and with the servers mostly out-seas or in cloud, without the option to use wired or wireless protocol, it is necessary to have a protocol that support Internet connection. With the main function of this protocols being the communication between the server and the aggregation node and given the low size of the packets, the chosen protocols must have good performance, low

bandwidth consumption and latency. With this in mind, we select MQTT and CoAP, both design for IoT, as the Internet protocols.

2.3.3.1 Message Queuing Telemetry Transport

MQTT is a messaging protocol [4, 17, 18] created in 1999 and standardized in 2013 at OASIS, that aims at connecting embedded devices and networks with applications and middleware. Built on top of the TCP protocol, it uses a publish/subscribe pattern, with a routing mechanism (one-to-one, one-to-many, many-to-many), to provide flexibility and simplicity transition making MQTT an optimal connection protocol for the IoT and M2M, being suitable for small, cheap, low power and low memory devices with low bandwidth networks. Compared to HTTP, MQTT is designed to have a lower protocol overhead.

MQTT consists of three components: subscriber, publisher and broker. A device registers as a subscriber for specific topics of interest in order to get the information published to that topic. The publisher acts as a generator of data for some topics, transmitting that information to the subscriber through the broker. Figure 2.9 illustrates the process used by MQTT in the subscribe/publish method.

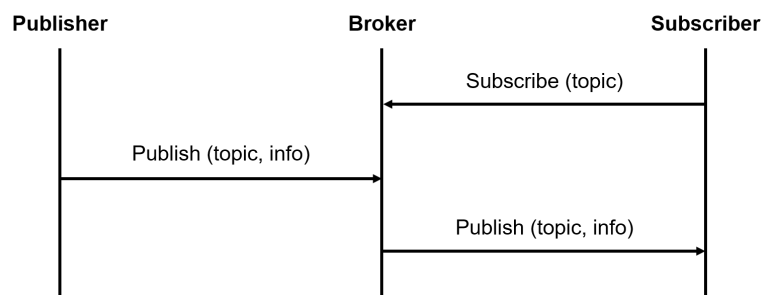


FIGURE 2.9: MQTT Publish/Subscribe process

To ensure that MQTT is reliable it uses three Quality of Service (QoS) levels [19]. In level 0 the message is delivered at most once and no acknowledgment of reception is required. In level 1 the message is delivered at least once and confirmation of message reception is required. In level 2, a four-way handshake mechanism is used for the delivery of a message exactly once. To provide the best

communication between clients MQTT protocol uses a set of features [18], for example, the Keep Alive time, in order to keep unused connections from wasting data; a durable session, that stores messages when the subscriber is disconnected; and a will, sent to all subscribers when the publisher leaves.

2.3.3.2 Constrained Application Protocol

CoAP is an application layer protocol [4] created in 2013 by the IETF Constrained RESTful Environments working group for IoT applications. It is based on REST on top of HTTP functionalities, representing a simpler way to exchange data between clients and server over HTTP. It is a re-design of HTTP functions taking into account that embedded devices have low processing power and energy consumption [18]. Also bearing in mind these type of devices, CoAP is built on top of UDP, which as a lower overhead than TCP. CoAP includes a built-in pub/sub mechanism which allows a client to subscribe and receive notifications to resources in the server. For this, it uses two layers [4, 18]: the transaction layer, which is responsible for detecting duplicates and providing reliable communications over UDP; and the request/response layer, that handles the REST communications. With this dual layer approach, CoAP does not need to use TCP protocol to ensure reliability and congestion control, with a process illustrated in Figure 2.10. Also, CoAP enables asynchronous communication which helps web applications where servers are not able to respond to request immediately.

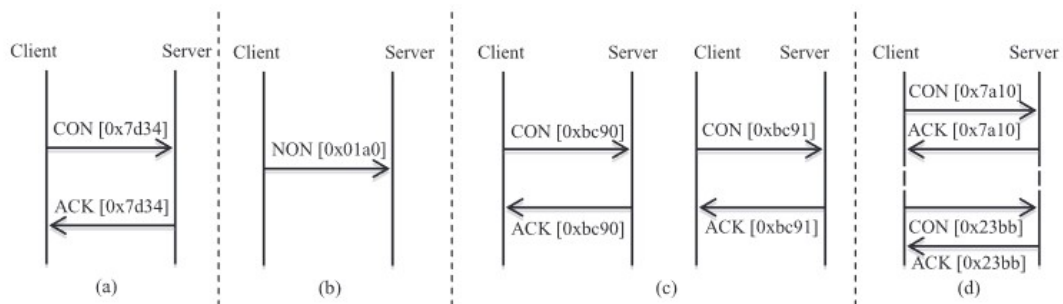


FIGURE 2.10: CoAP message Types

One of the main goals of CoAP was to minimize the packet fragmentation, making typical CoAP message size between 10 to 20 bytes. To ensure the best communication CoAP uses a set of features [4, 18], such as security, using DTLS to guarantee integrity and confidentiality of exchanged messages; and resource discovery, as the server uses known URI to provide resource discovery for the client.

2.3.4 Remarks

As device to device communications, after the description in the previous sections, the advantages and disadvantages presented and without the possibility to test every protocol and with a specification requiring a low-cost, low-power, high range, multi-node communication protocol that also has a low complexity hardware configuration, the chosen protocols to do further tests are I²C, RS232, ZigBee and LoRaWAN.

As for the device to server communication, after the description in the previous sections, the advantages and disadvantages presented, and bearing in mind that we need a good performance, low bandwidth consumption and latency suitable for low power, low memory and low bandwidth devices, and capable of running in a JavaScript client environment the chosen protocol to do further tests and implementations on the system is MQTT.

2.4 Controlling Platforms

IoT relies on group of devices connected over a network controlled by an application using a communication protocol. These devices can be anything capable of retrieving and storing data, but the most usual are microcontrollers or SoC devices. SoC [20] is a replacement for a computer, based on an electronic chip or integrated circuit containing the whole computing system. With this approach the cost of producing a computing system has dropped making this devices much cheaper for

the end consumer. The most common examples of this platforms are the Arduino or ESP8266 based boards, the Raspberry Pi or BeagleBone microcomputers and the Intel Edison.

2.4.1 Arduino

Arduino, created in Italy in 2005 [21], is an open-source electronics single board based on easy implementable hardware and software [22, 23]. It is a simple input/output board and a Processing language development environment [24], with several programming ways and languages, being the most common its own Integrated Development Environment (IDE) and C++ language.

The most common Arduino boards are based on the ATmega microcontroller, eg. the Uno and the Mega. These boards, on Figure 2.11, are capable of interfacing different peripherals, sensor and wireless communication devices through input/output pins, providing a simple and low cost platform for Research and electronic projects.

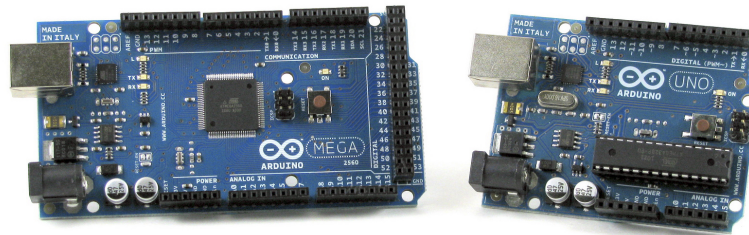


FIGURE 2.11: Arduino boards

According to [24, 21], Arduino boards are unique in nature because of features, that justify their adaptability, such as Open Source Software and Hardware, cross-platform compatible, easy to program via the Arduino IDE and USB which is flexible and robust for all sort of users and projects. Arduino boards have the disadvantages of low memory, single thread operation and are not capable of running an OS.

2.4.2 ESP8266

The ESP8266 chip is a low-cost WiFi module capable of providing full-stack TCP/IP and a microcontroller unit, produced by Espressif Systems in 2014. This module when connected to other devices, such as an Arduino board, allows a connection to an WiFi network and simple TCP/IP data transaction [25].

There are several releases of the chip, with different features, but the most usual is the ESP-12 [26], a WiFi module based on the ESP8266 core processor, providing the ability to interface with sensors and actuators through I/O pins and offering a complete and self-contained WiFi (IEEE802.11 b/g/n) networking solution as a low-power, low-cost and minimal space device.

Although the chip can work by itself, pinout is not compatible with most breadboard or PCB formats, so it is usual to see the chip integrated in a breakout board, as seen in Fig. 2.12. The Adafruit Huzzah ESP8266 Breakout is designed to allow an easy access to the ESP-12 chip and also adds voltage regulator, FTDI pinout to program the chip, among other features.

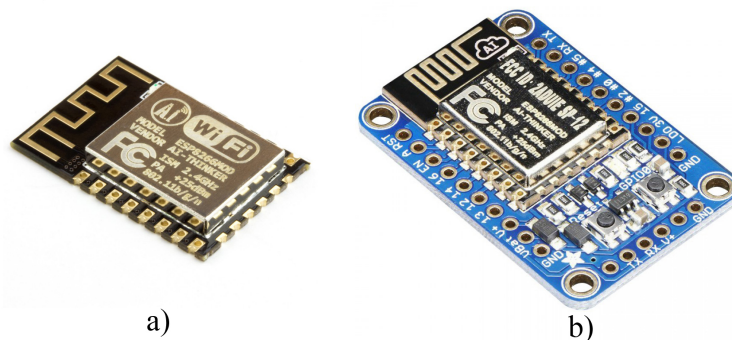


FIGURE 2.12: ESP8266 components: a) ESP-12 chip; b) Adafruit Huzzah ESP8266 breakout

It is a great solution when an WiFi connection is needed on a microcontroller project, being the main choice due to its low-cost, low consumption and reduced size, but it has the disadvantage of needing additional hardware to make it user-friendly.

2.4.3 Raspberry Pi

The Raspberry Pi [6, 20] is a single board credit card size computer created by Raspberry Pi Foundation in the United Kingdom in 2009. These boards are a low-cost computer with enough processing power and memory, that support programmable I/O ports and the use of standard peripherals. As a computer, the Raspberry Pi has an Operating System running on a SD card, mainly a Linux OS but more recently it can also run an IoT version of Windows 10.

The Raspberry Pi was design to facilitate the computer science, providing an inexpensive way to allow full control and customization over a piece of hardware. There are already several releases of the board, being last ones the Raspberry Pi 3, with integrated WiFi and BLE and a low-power functionality, and the Raspberry Pi Zero W, offering the same functionalities as the 3 but with a low-size and even more low-power.

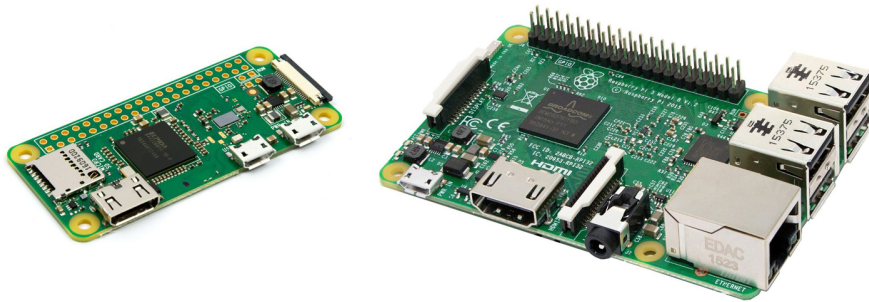


FIGURE 2.13: Raspberry Pi boards

The low cost, low size computer with several programmable pins and communication capabilities are the major advantages for Raspberry Pi boards. As a disadvantage Raspberry Pi has the lack of analog ports.

2.4.4 BeagleBone

BeagleBone was created by group of passionate individuals, including several employees of Texas Instruments, who wanted to create a powerful, open-source embedded device. Therefore BeagleBone boards are low-cost single-board computers based on low-power Texas Instruments processors with an ARM Cortex-A series core with all of the expandability of today's desktop machines [27].

BeagleBone is capable of running several Linux distributions or even Android and is also capable of acting like an Arduino through I/O pins, making their boards simplified physical computers and/or networked-enabled devices with a super-simple out-of-box learning experience.

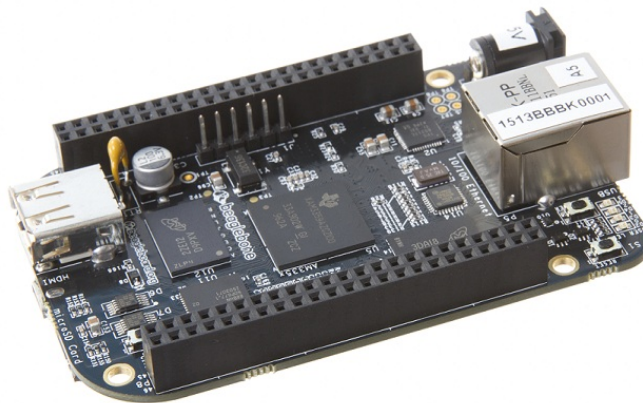


FIGURE 2.14: BeagleBone Black

2.4.5 Intel Edison

The Intel Edison module is a SD-card-sized computing chip based on the Intel Atom processor, designed for IoT and wearable projects. Launch in 2014 by Intel, it contains a high-speed, dual-core processing unit with integrated WiFi, Bluetooth low energy, storage and memory, and a range of I/O options [28].

Because of its small footprint and low power consumption it is meant to be embedded in devices or development boards, but due to his 70-pin DF40 Series

header connector connectivity with most boards must be done using a special piece of hardware. For this, Intel provides the Intel Edison Kit for Arduino that allows a quick and easy conversion to an Arduino like board, with exposed headers for I/O pins [28].

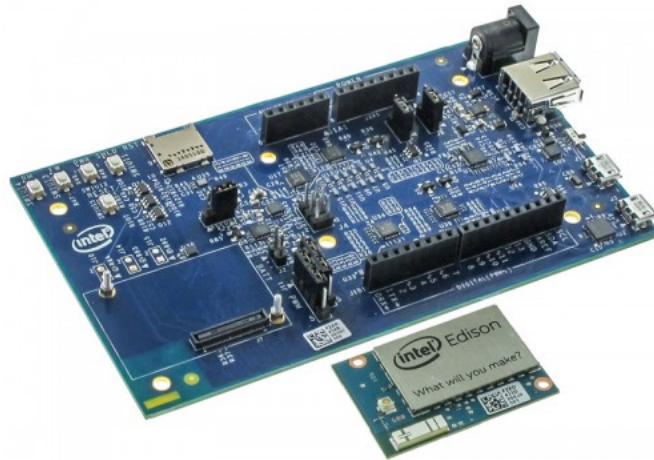


FIGURE 2.15: Intel Edison module and Kit for Arduino

The Edison module provides a small, low-power solution with a big processing power, but the need to have additional hardware and price makes it a more unlikely solution.

2.4.6 Remarks

As stated, there are multiple choices for IoT devices capable of creating sensor networks. Bearing in mind that the system needs to be low-cost, low-power consumption and capable of adapting to multiple necessities, like environmental, user specifications, different sensors, actuators or communication protocols, these are the main focus when choosing a board to integrate in the system. Also features like dimensions, programming language, available firmware and online support are points to have in mind.

TABLE 2.3: Major IoT devices characteristics [29, 30, 31, 32, 27, 33, 34]

Parameter	Arduino Uno	Arduino Mega	RPi Zero W	RPi 3	BeagleBoard Black	Huzzah ESP8266	Intel Edison
Processor	ATmega328P	ATmega1280	BCM2836 32 bits	BCM2837 ARM Cortex-A53 64 bits	AM3358 ARM Cortex-A8	ESP8266	Intel Atom
Operating System	-	-	Debian (Linux)	Debian (Linux)	Debian (Linux)	-	-
Programming Language	C++ (Arduino IDE)	C++ (Arduino IDE)	Any language	Any language	Any language	MicroPython, NodeMCU, C++ (Arduino IDE)	C++ (Arduino IDE), Python, Node.js, JavaScript
Integrated Communication Protocols	UART, SPI, I2C	UART, SPI, I2C	UART, SPI, I2C, WiFi, BLE	UART, SPI, I2C, WiFi, BLE	UART, SPI, I2C, CAN Bus	UART, I2C, SPI, WiFi	UART, SPI, I2C, WiFi, BLE
Analog Pins	6	16	0	0	7	1	6
Digital pins	14	54	40	40	65	9	20
Power Consumption ¹	50mA	50mA	100mA	300mA	210mA	56mA	50mA
Dimensions	68.6 x 53.4 x 17 mm	101.6 x 53.34 x 17 mm	65 x 30 x 5 mm	85.60 x 56.5 x 17 mm	86.36 x 53.34 x 17 mm	25 x 38 x 5 mm	127 x 72 x 12 mm
Price ²	20	35	11	39	44	14	85

After the description in the previous sections, the advantages and disadvantages presented, the information on Table 2.3 and knowing that the system requires at least one aggregation, capable of connection to the Internet, and one sensor node, capable of retrieving sensor data, both analog and digital, with both nodes needing a various communication protocol adaptation, the chosen platforms to do further tests are the Raspberry Pi 3 and the Huzzah 8266, for the aggregation node, and the Arduino Uno for the sensor node. There is also a possibility to use the Raspberry Pi Zero W or the Arduino Mega, as a substitute for the previous stated, when space or more pins are required.

2.5 Related Work

2.5.1 IoT Gateway

IoT gateways [35] are dedicated hardware applications used to connect the user to the network, allowing the conversion of data between the short distance communication protocols to the traditional communication network. The gateway is

¹Average on idle

²Price according to official reseller, without VAT

supposed to support different types of sensor nodes, multiple communication protocols, both wireless or wired, and provide a set of unified information for the application or user, making these only responsible for data processing. The main challenge on creating an IoT gateway is the lack of standards, being that each sensor node can communicate with a different protocol that is not compatible with others. This makes the development of a general purpose gateway a complicated task, which explains why it is common to find gateways developed for specific applications. Nevertheless, all have the same key requirements: low-cost hardware, easy implementation and extensibility and an application layer support.

R. Gerstendorf in his article [36] examines ten platforms for creating smart homes, from ZigBee to the Apple HomeKit. All ten are market solutions available to consumers as a ready-to-use product in order to control smart devices through a gateway. Each one has a communication protocol, proprietary characteristics and a range of pros and cons.

In the literature it is possible to find several proposals [23, 37, 38] for IoT gateways implemented using low-cost hardware devices, such as Arduino and Raspberry Pi. Most of them use these devices to support the web server, which difficults its access from outside the network. Other solutions were found [39, 40], that use wireless communication protocols for specific applications and little to none IoT gateways were designed to use wired protocols. Only one of these [41] did it, where the authors used the gateway with the RS485 protocol to control end-devices from the Internet. This makes all of these solutions limited in flexibility and adaptation to other environments.

The literature presented a similar concept for a multi-communication protocol IoT gateway. In this [42], the authors present a heterogeneous IoT gateway capable of using the same board to communicate with multiple wireless protocols as well as support for a large amount of communication buses, in a modularity basis.

2.5.2 Visualization Platforms

The IFTTT [43] platform is also a service that allows users to create chains of simple conditional statements, connecting their devices to other services. For example with IFTTT it is possible to turn on a light bulb with a tweet or Facebook post, or automatically send an email every time a door or a window is open. This platform works with all major controllers and several major brand services. But it is impossible to keep track of the information and compare all the devices, because statements created are based in a single device, keeping out the option to connect every device with a single action.

Also available is Amazon AWS IoT [44], a cloud based platform capable of interfacing with end point devices, to monitor and control, providing the capability of creating rules mechanisms and additional features such as Machine Learning or storage. To interface with this solution, all devices must implement the AWS IoT Device SDK, in order to communicate with the platform through MQTT or HTTP. The main disadvantage of AWS IoT is that it is a paid service.

The most similar solution found was myDevices Cayenne [45], an IoT drag and drop project builder. With similar functionalities, like remote monitoring and control, use of rules, a web platform for data display, this project allows the user to connect to their devices using MQTT or LoRaWAN. The range of devices supported by the platform includes all Arduino and Raspberry Pi boards and a limited number of sensors, being this one of the disadvantages of the platform. The main difference from our project to Cayenne, is that the last aims to the hobbyist market and not to the custom and plug and play solution market.

Chapter 3

Communications Tests

One of the purposes of our research was to identify the best communication protocol for connecting all the system nodes, not only to guaranty the best data transfer but also an easy implementation with low cost and power efficiency. There are two main communication parts in the system, the server side communications, that connect the server to the network devices, and network side communications, that connects the devices inside the network. The following sections introduce the tests done to encounter the best solution for our system.

3.1 Server side communications

When connecting devices to servers, it is usual to use standard HTTP connections, with POST and GET requests, but, as we see in Chapter 2.3, the MQTT protocol is best suited for our system requirements, since it has a lower bandwidth consumption and latency, being ideal for IoT networks.

To ensure that the best communication protocols were used in the system, a set of tests was applied to the network nodes to prove the MQTT advantages facing standard HTTP. These tests focused on the exchange of strings of data between the aggregation node and the server, simulating a normal operation of the system. For these, three different tests were made in order to see the following:

(i) Communication Pattern; (ii) Bandwidth usage; (iii) Latency. To obtain and evaluate the results the Wireshark³ software was used.

For the first test, the aggregation node sends a typical system message, i.e. *"sensor:temp1,20"*, in order to see the transaction packets exchange needed to get the message to the server.

In the second one, a similar method is used, but this time 100 messages were sent with the purpose of getting the bandwidth used per second.

The last one uses the same test done for bandwidth, but evaluating the average RTT value.

Since MQTT has multiple levels of reliability, that imply different amounts of packets per transaction, all the tests have been done using QoS level 0 and 2 messages. Also, all tests were performed assuming that the MQTT connection was already established, including the subscription process, and with an SSL encrypted connection.

3.1.1 Test Scenario

The tests were performed using a Raspberry Pi 3, as the aggregation node, running a Python script that uses the requests library to send HTTP POST messages or the Eclipse Paho Python Client [46] library for publishing MQTT messages; and a Virtual Private Server (VPS) with a PHP script that receives the HTTP POST message and that also hosts the MQTT Broker and a Python script with the previous stated library to subscribe and receive the MQTT message. Figure 3.1 shows this test scenario.

3.1.2 Results

As displayed in Figures 3.2, 3.3 and 3.4, it is possible to notice that the HTTP POST requires 28 packets to transmit the message while MQTT only requires

³<https://www.wireshark.org/>

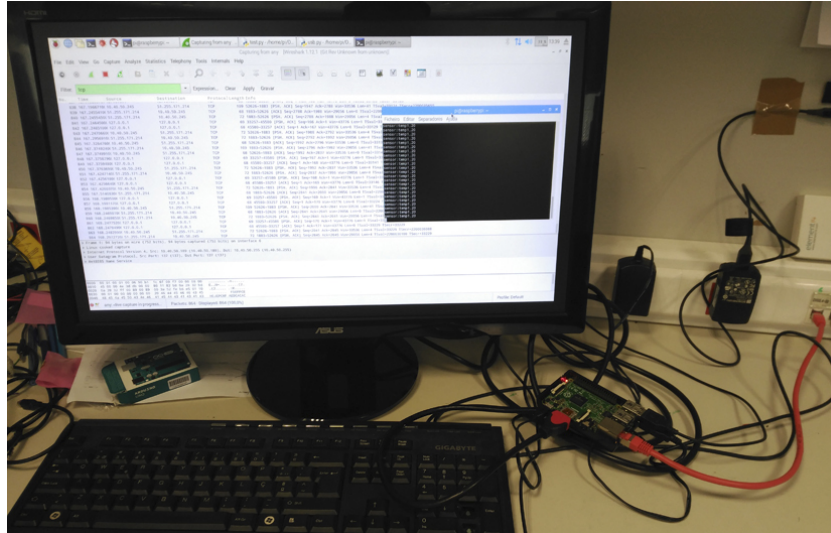


FIGURE 3.1: Server side communication test scenario

2 for QoS 0 or 3 for QoS 2. On the HTTP POST is possible to see that the SSL certificate requires several packets, light grey packets on Figure 3.2, and that the MQTT secure connection with SSL does not require additional ones. The only difference between MQTT messages is in the amount of packets required to improve reliability, with QoS 2 requiring an additional published message that piggybacks with the ACK from the server and then the ACK to that message from the aggregation node, due to a four-way handshake mechanism that is used to guarantee the message is delivered.

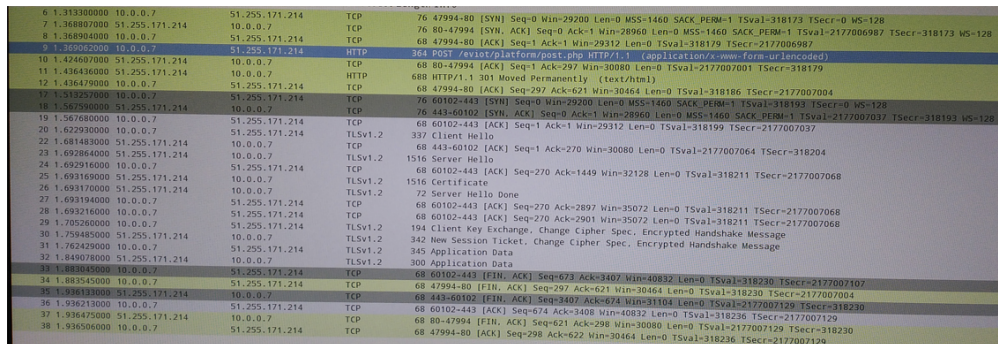


FIGURE 3.2: HTTP POST Wireshark packet capture

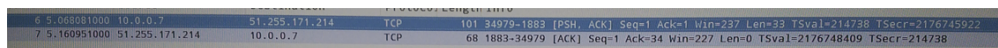


FIGURE 3.3: MQTT QoS 0 Wireshark packet capture

No.	Time	Source	Destination	Protocol	Length	Info
4	1.529975000	10.0.0.7	51.255.171.214	TCP	103	3530->1883 [PSH, ACK] Seq=1 Ack=1 Win=237 Len=35 TSval=202326 TSecr=2176714893
5	1.581019000	51.255.171.214	10.0.0.7	TCP	72	1883->3530 [PSH, ACK] Seq=1 Ack=36 Win=227 Len=4 TSval=2176717369 TSecr=202326
6	1.581153000	10.0.0.7	51.255.171.214	TCP	68	3530->1883 [ACK] Seq=36 Ack=5 Win=237 Len=0 TSval=202331 TSecr=2176717369

FIGURE 3.4: MQTT QoS 2 Wireshark packet capture

Figure 3.5 shows the bandwidth consumption for each of the 3 messages sent. With these results it is possible to prove that MQTT has a better bandwidth consumption with less 98% consumption facing the standard HTTP POST. Comparing both MQTT QoS and as concluded in the previous test, the QoS 2 transaction has more packets consequently wasting more bandwidth, with an increase of 40% in bandwidth usage facing QoS 0.

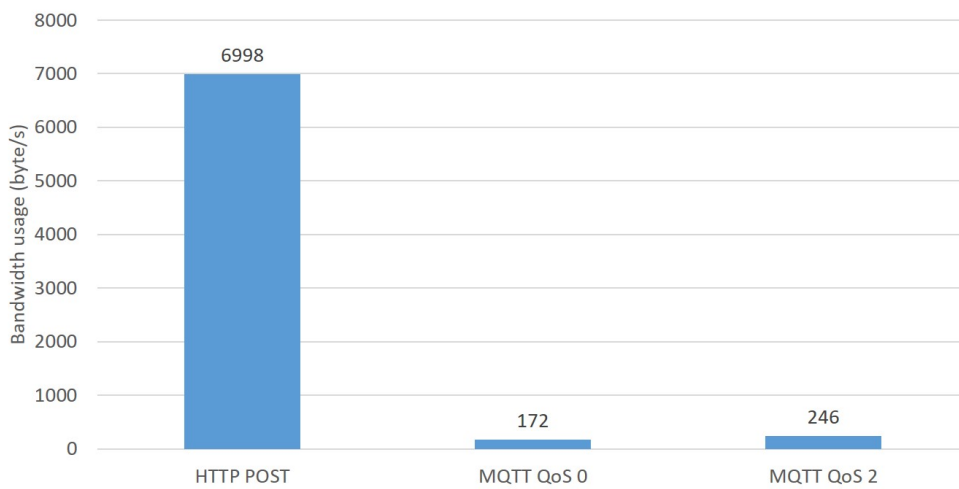


FIGURE 3.5: Bandwidth consumption comparison

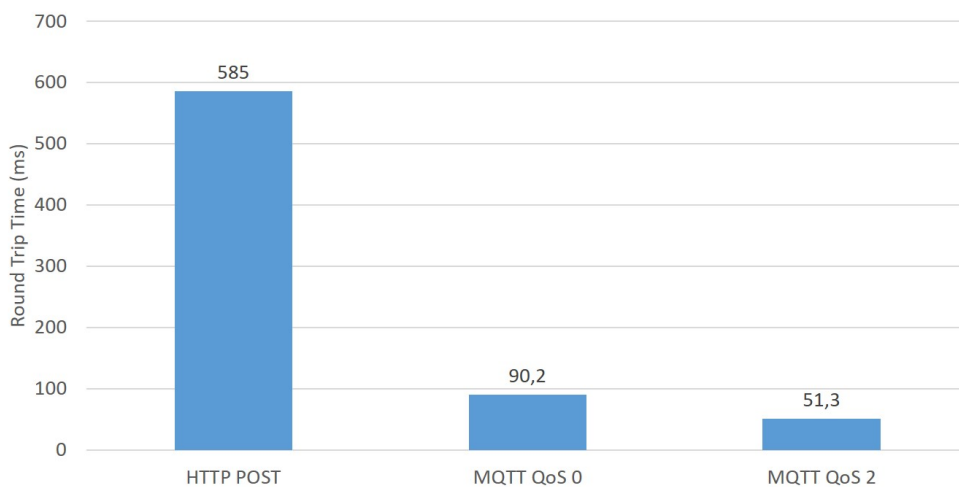


FIGURE 3.6: Transaction RTT comparison

Latency was evaluated with the RTT between the sent message and the ACK. The results are shown in Figure 3.6, where it is possible to see that MQTT has a

better latency with almost 500ms of difference facing the standard HTTP POST. Comparing both MQTT QoS, the QoS 2 has a lower latency, about 40 ms, since it implements a reliability method to ensure the message was delivered, meaning that the transaction is only completed when the ACK is sent to confirm the message. In QoS 0, the transaction is concluded without the need of an ACK to confirm it. Although the server confirms the QoS 0 message with an ACK, since is not necessary, the deployment of the ACK is done with a slight delay, rather than the QoS 2 ACK message needed is deployed immediately after the r of the message, thus justifying the difference.

3.2 Network side communications

While it is possible to use almost every communication protocol available, some environments require specific adaptation and as we see in Chapter 2.3, the best suited protocols for our requirements are the I²C and RS232, for wired protocols, and ZigBee and LoRa, for wireless protocols. Also in Chapter 2.4, we concluded that the Raspberry Pi 3, ESP8266 and Arduino Uno are the best boards for each node available in the network.

To ensure that the best communication protocols were used in the system, a set of tests was applied to the network nodes in order to choose the best one. These tests focus on the exchange of strings of data between the aggregation node, the master, and the sensor node, the slave. The main goal was to see if the communication protocol can work with these platforms, how well they perform when increasing the distance between nodes and also to compare the results with theoretical ones.

For these, three different tests were made in order to see the following values: (i) Throughput; (ii) Message Delay; (iii) Efficiency. Also the setup complexity for the system to work and master slave combination were evaluated.

For the first test, the master sends a continuous amount of data packets, with no interval between them, and the slave just receives them, registers the timestamp and increases the number of packets received. Then we see how many packets the slave can process in a second, in order to obtain the true system throughput.

In the second one, a similar method is used, but this time only 100 packets are sent with an interval of one second. The slave receives the packets and registers the time between packets. Then the average time is calculated, in order to achieve the delay, over one second, that the system needs to receive the packets.

In the last one, 1000 packets were sent to the slave and the slave only increased the number of received packets if the data was equal to the one sent. This way it is possible to see how many packets were lost or contain errors.

All the tests have been done with a distance of 1, 2, 5, 10, 20 and 50 meters between nodes and with the following board combination, as both master and slave: Raspberry Pi - Arduino, Arduino - Arduino, ESP8266 - Arduino, in order to evaluate every scenario possible.

3.2.1 Setup Complexity

For I²C and RS232, the Arduino, ESP8266 and Raspberry Pi platforms have native pins to use these protocols, so no additional hardware is needed. ZigBee and LoRa need the external radio interfaces in order to communicate. For ZigBee a pair of XBee S2 and Arduino Wireless Proto Shield, Figure 3.7, and XBee Explorer, Figure 3.8, were used to connect everything. For LoRa only a pair of Adafruit RFM69HCW, Figure 3.9, is required. The connections between boards and modules can be found in Appendix A.

In terms of software, for I²C in the Raspberry Pi the `smbus` library was used and in the Arduino and ESP8266 the `Wired` library. For a correct communication only the slave address needs to be set on both platforms as well as the callbacks on both ends. I²C does not allow the Raspberry Pi to be a Slave, so some modifications to the code were needed in order to set the Arduino as a Master. Also a minimum

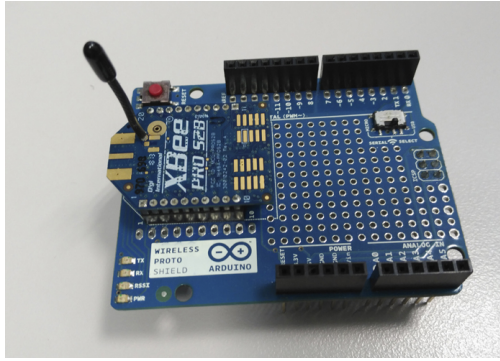


FIGURE 3.7: Arduino Wireless Protonshield with XBee S2



FIGURE 3.8: XBee Explorer

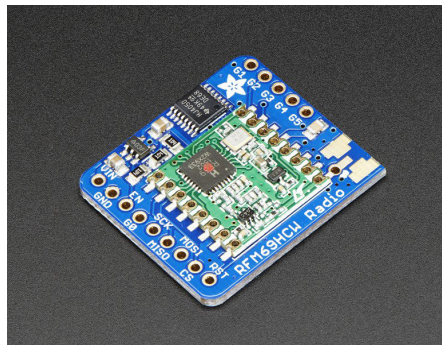


FIGURE 3.9: Adafruit RFM69HCW

of 1 ms delay was needed between packets in order to avoid overflow of packets when sending to the Arduino.

For RS232 the communication works as a simple Serial Communication with the RX and TX ports on both ends, with a modification on settings of the Raspberry Pi so that the OS does not use that port for console communication. Also the pySerial library was needed.

For ZigBee, the most complex configuration, it was necessary to pair the radios using XCTU⁴ software. After that, each radio was assigned with an address that the other end has to know to be able to send the packet. On the Raspberry Pi the XBee library was used and on the Arduino, a 10ms delay between packets was needed to ensure a correct transmission of packets.

For LoRa, the RadioHead library was used on all platforms. To configure it, only an address for each node had to be chosen, guaranteeing it was different for each node.

⁴<https://www.digi.com/products/xbee-rf-solutions/xctu-software/xctu>

3.2.2 Test Scenario

The wired communication tests were performed using Cat5 UTP (4 pairs) cable or standard Ethernet cable, with the required lengths, to connect master and slave. Figure 3.10 shows an example of the tests performed for Raspberry Pi and Arduino using RS232 with a 5 meters cable.

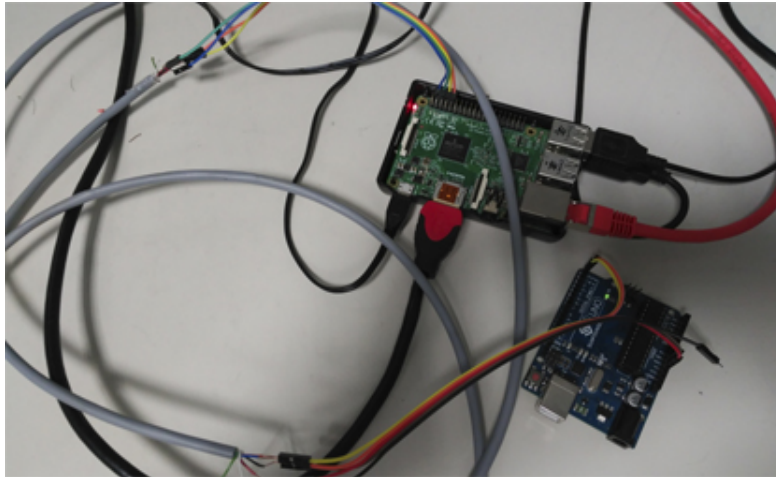


FIGURE 3.10: Wired test example

The wireless communication tests were performed on floor 0 of ISCTE-IUL Building 2, near the Instituto de Telecomunicações laboratories. An approximate model of the space used can be seen at Figure 3.11, with the master and each slave positions marked.

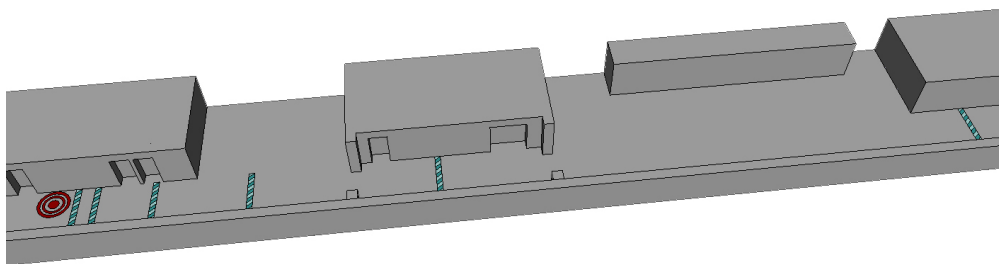


FIGURE 3.11: Test Scenario Model (Master: Red Circle; Slaves: Blue Lines)

This specific grounds were chosen due to availability of 50 meters in line of sight with some variations in space, with open spaces, doors and some people walking. Figure 3.12 shows the master and the 50 meters slave position, with a line of sight between them.



FIGURE 3.12: Test scenario grounds (Left: Master position; Right: Slave 50 meter position)

Figure 3.13 shows an example of the tests performed for the Arduino combination using ZigBee at 10 meters.



FIGURE 3.13: Wireless Test example (Left: Master position; Right: Slave 10 meter position)

3.2.3 Results

The first thing to notice is that some protocols were not able to connect in some platforms, I²C with ESP8266 is one of the cases and also LoRa with the Raspberry Pi. The ESP8266 and ZigBee connection can be done, but due to ZigBee interference with Wi-Fi, the results were not viable so it was discarded as a possibility.

One disadvantage detected was the fact that the Raspberry Pi can not be an I²C slave, having the Arduino to wait a second between packets and therefore affecting the results.

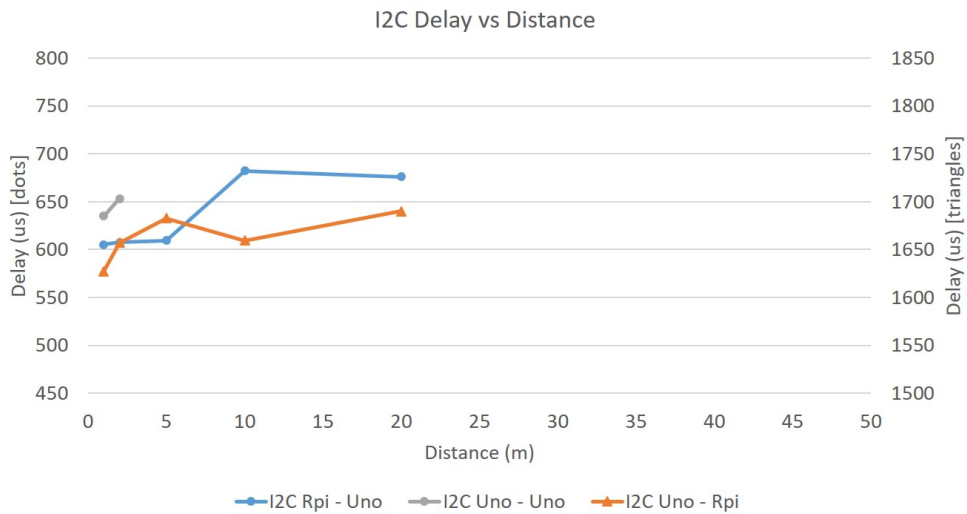


FIGURE 3.14: I²C Delay vs Distance results

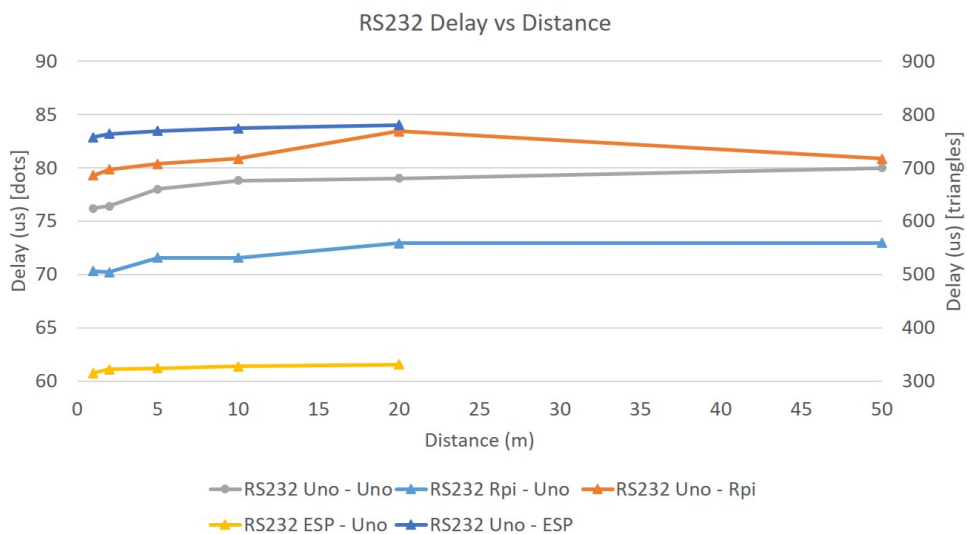


FIGURE 3.15: RS232 Delay vs Distance results

With these delay results, shown in Figures 3.14, 3.15, 3.16 and 3.17, it is possible to understand that the increase of distance does not affect much the delay value, with only a slight increase in all protocols. RS232 has the best results in all scenarios, with a delay ranging from around 75 to 800 us. LoRa has the higher delay with values ranging from 800 to 2500 us and also the biggest variation between scenarios. As of distance, the only one really affected is ZigBee with a

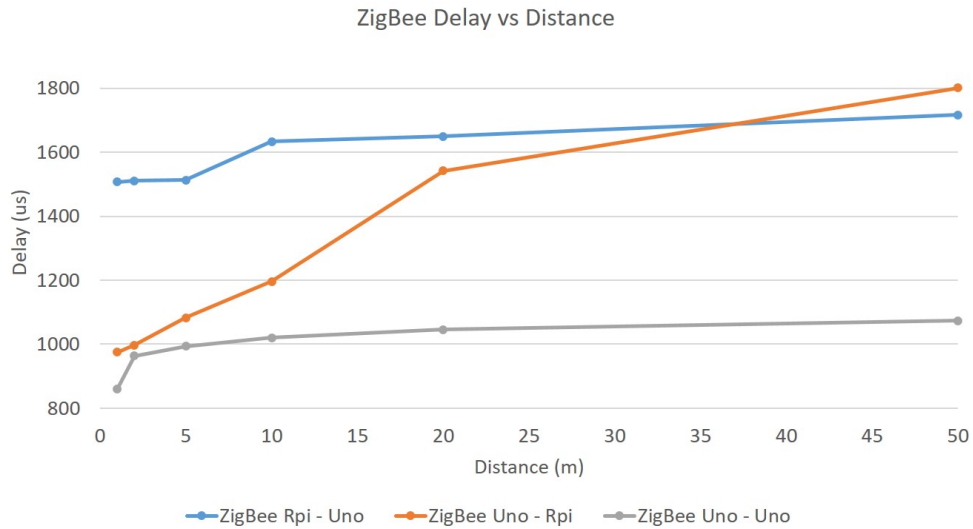


FIGURE 3.16: ZigBee Delay vs Distance results

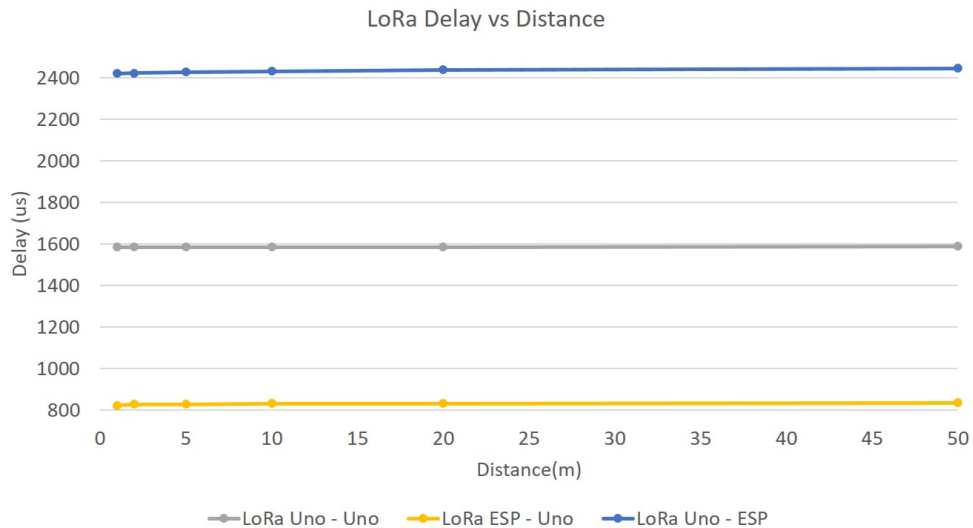


FIGURE 3.17: LoRa Delay vs Distance results

bigger increase in delay over 10 meters in all scenarios. Regarding the master slave combination, was when both platforms were the Arduino Uno that the best results could be achieved. When only one Arduino Uno was present, the delay decreases when the it is the slave.

Regarding throughput results, as shown in Figures 3.18, 3.19, 3.20 and 3.21, it is possible to understand that distance affects the results, with a different pattern in each scenario, and that in every protocol the maximum value reached is quite different from the theoretical expected values. In some cases there are ups and

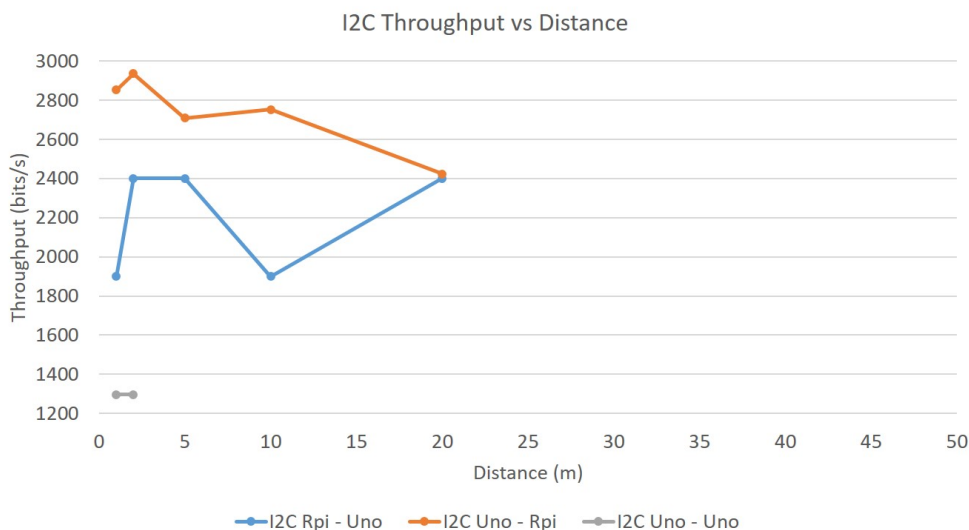


FIGURE 3.18: I²C Throughput vs Distance results

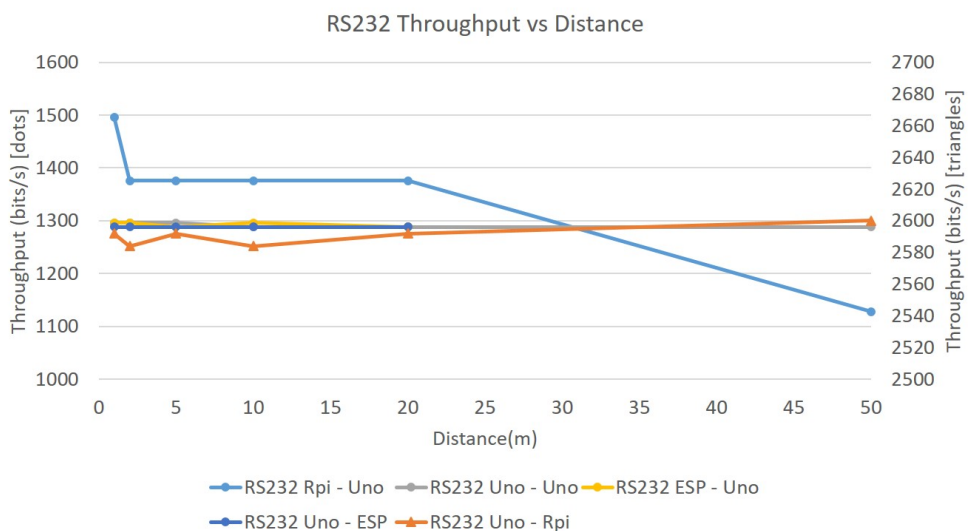


FIGURE 3.19: RS232 Throughput vs Distance results

downs with the increase of distance but always with slight variations, so that was not taken in consideration. I²C has the most inconsistent results with big variations in both distance and master slave combination, with a throughput raging from 1.3 to 2.9 kbits/s. On the other hand, RS232 has similar throughput results, raging from 1.1 to 2.6 kbits/s, but with a more consistent and linear results. LoRa and ZigBee got very similar results, ranging from 0.75 to 1.3 kbits/s, with LoRa having the most linear results, almost none variation with the increased distance. Regarding the master slave combination, the best results are achieved when only one Arduino Uno is present, with some variations as it is the master or the slave.

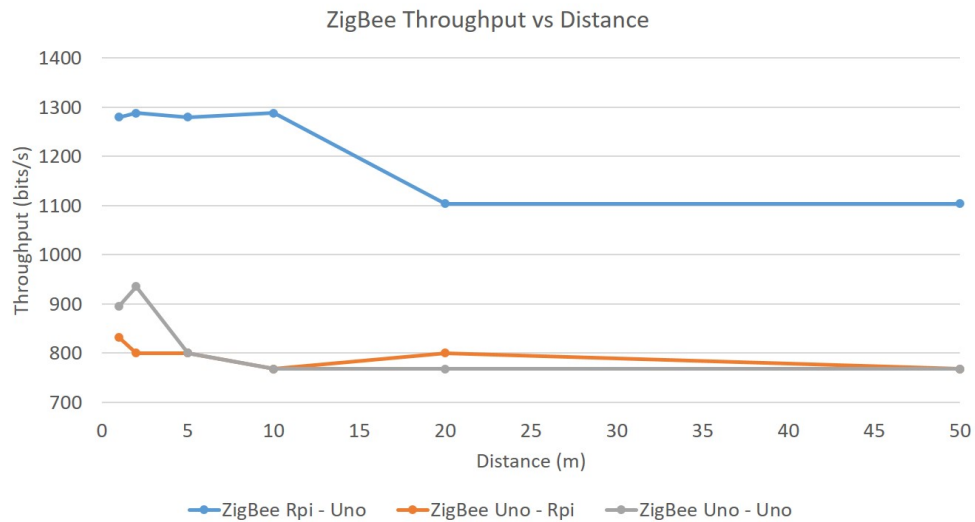


FIGURE 3.20: ZigBee Throughput vs Distance results

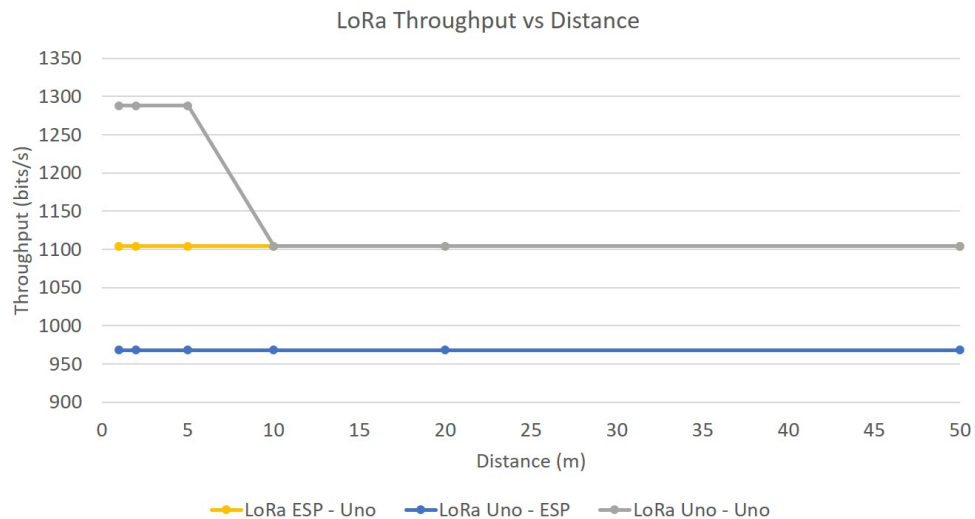


FIGURE 3.21: LoRa Throughput vs Distance results

As for the efficiency tests, I²C had the worst results as communications at 50 meters were not feasible and when both platforms were Arduino Uno, nothing over 2 meters worked. This could be improved with additional hardware such as I²C range extenders. RS232, when connecting ESP8266 and Arduino Uno was not able to connect over 20 meters. Every other scenario tested got a 100% efficiency rate, including LoRa and ZigBee for every distance.

3.3 Discussion

This chapter focused on comparing communication protocols, to choose the best suited for a set of scenarios using IoT low-cost devices. For the server side connections the results were evaluated based on communication patterns, bandwidth consumption and latency, while the network side communication were based on delay, throughput and efficiency.

Regarding server side communications, it was possible to identify the advantages of MQTT as a protocol designed for IoT, with low bandwidth consumption and latency when facing the HTTP protocol. All of this without compromising the security of the messages and with only a slight increase in consumption when guaranteeing higher reliability.

As for the network side communications, one thing that it is possible to highlight from the beginning is that wired protocols have better results than wireless ones, in both delay and throughput. As for distance, with the increase affecting each protocol differently, each scenario presents only slight variations, meaning that up to 50 meters there is no availability and reliability problems.

Concluding the communication research, RS232 is the better choice when a wired protocol is needed and LoRa is a more reliable choice for a wireless protocol mainly because of the low complexity and cost needed and the fact that does not interferes with WiFi.

Another focus was choosing the best devices for each node of the system, based on previous stated characteristics and the communication tests done. With all these results in mind, it is possible to say that the best suited combination includes an Arduino Uno as an Sensor Node and the ESP8266 as the Aggregation Node. This last decision takes place not only because it is a cheaper solution, not putting in risk the reliability of the system, capable of sustaining the node specifications but also due to the Raspberry Pi 3 extra features that are not needed for the system and which implies a bigger power consumption and some interference between the system console and the RS232 lines.

Chapter 4

System Architecture

The main goal of this project was to develop an IoT system that can adapt to any object or environment, based on automation, efficiency and versatility, capable of turning them into smart ones. With this system it is possible to have full secure remote control and monitor off the environment through a web platform, allowing the visualization of real time data and the control of actuators.

As said in Chapter 2.5, there are already some solutions available, for gateways and visualization platforms, but none of them was capable to fulfill our requirements so all the modules needed to assemble the system were developed from scratch.

The system relies on a group of devices connected over a network that is controlled by an application using a given communication protocol. Figure 4.1 presents a high level system architecture and, as shown, the system is composed by hardware, software and the implemented communication features. The proposed system was conceived using hardware capable of supporting multiple communication protocols and flexible enough so as to run adaptable software, as described in the following sections.

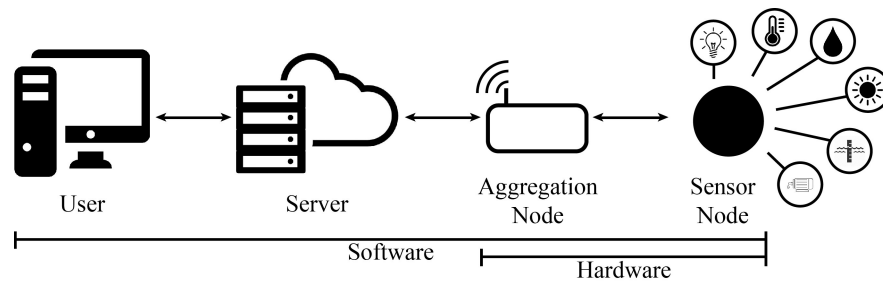


FIGURE 4.1: System Architecture

4.1 Software

As shown in Figure 4.1 there are several components which include software, essential to the proposed architecture. As stated before, the goal was to build a system that can be adapted to any environment and specification. With that in mind all the software was designed to be used with little or none modification and easily modifiable by any person.

The system software is divided in two key features, the visualization platforms and the support scripts that run in all the nodes. Both features are mentioned in detail in the sections below.

4.1.1 Visualization Platforms

The visualization platforms consist on a display for the information gathered from the sensor network. These provide the user not only with the possibility to monitor, in real time, what is happening in the environment but also offering a bridge to the sensor network, using the MQTT protocol and providing real time control.

The visualization platform is available in both web and mobile, the last one being an adaptation from the web platform for small screens. The platform was built from the ground using HTML, Javascript, JQuery, CSS, AJAX and PHP, for both control and communication. It is hosted on a VPS, containing a private MySQL database for storage, with the relational design presented in Figure 4.2.

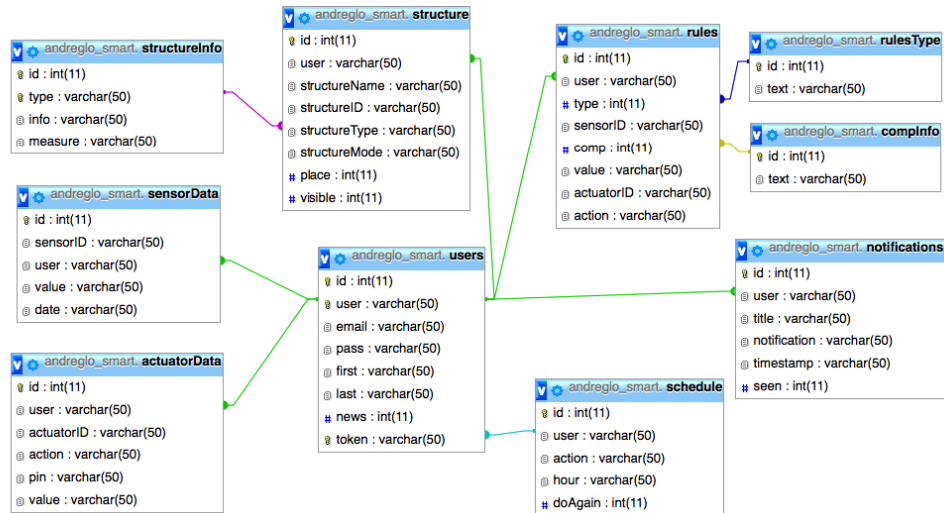


FIGURE 4.2: Database relational design

The platform main page, as can be seen in Figure 4.3, offers the user a graphic way to see data as well as a set of features in order to control the network. All the features offered by the platform are presented to the user in two main sections that dynamically adapt to what the user is doing.



FIGURE 4.3: Screenshot of the web platform

4.1.1.1 Monitoring and Control

On the top section, Figure 4.4, the user can see all the sensors and actuators present in the network as well as the real time information about them. For the sensors the name, type, last value and timestamp are provided, for actuators only the name and state (ON/OFF).

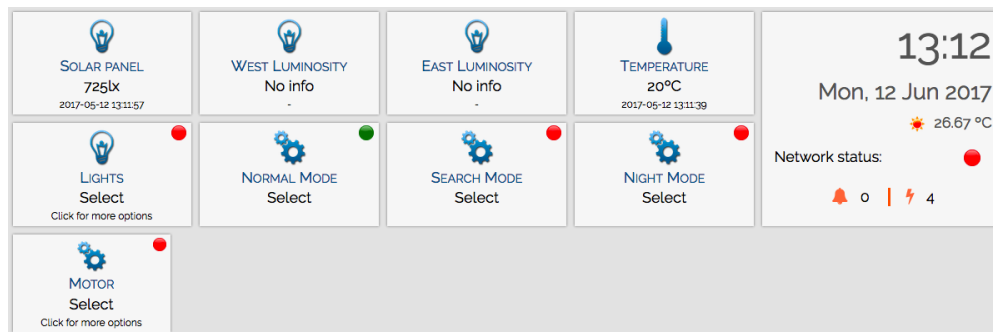


FIGURE 4.4: Top section with sensor and actuator information

For both types, an action is deployed when the corresponding cell is pressed. At an actuator cell, when "Select" is pressed the platform tells the network to change that actuator state, thus changing the state on the platform if the action is successful. If the cell is clicked, no matter whether it is a sensor or actuator, it opens a custom information panel at the bottom section of the platform.

For each actuator selected, Figure 4.5, the possibility to change the actuator state is available with a press of the "Select" button. Also allows the user to schedule actions for that actuator in two different ways. One sets the time the actuator remains in a state from the current time, the other allows a date and time schedule, both start and end, with the possibility to set the repetition of the schedule every day, week or month. All the schedules can be seen and canceled.

For each sensor selected an interactive graph, Figure 4.6, where the user can choose the period of time to display and see the timestamp for each value, with all the data retrieved from the network, is shown as well as information of name, type, current, maximum and minimum value retrieved.

The last cell to the right, Figure 4.7, is the only default one, which include present time and date, current temperature for the region where the network is,

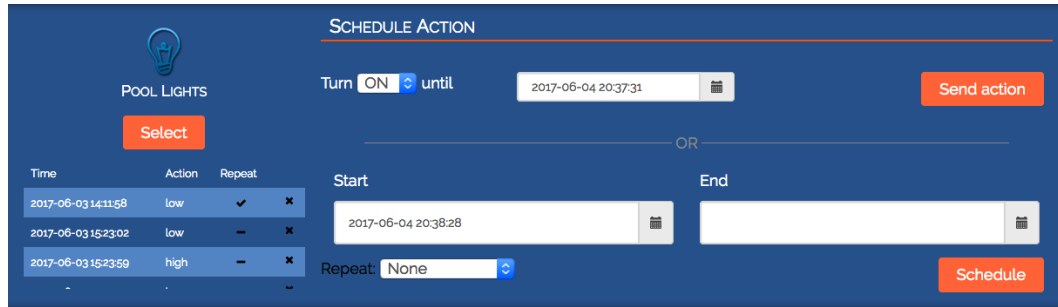


FIGURE 4.5: Bottom section with actuator schedule

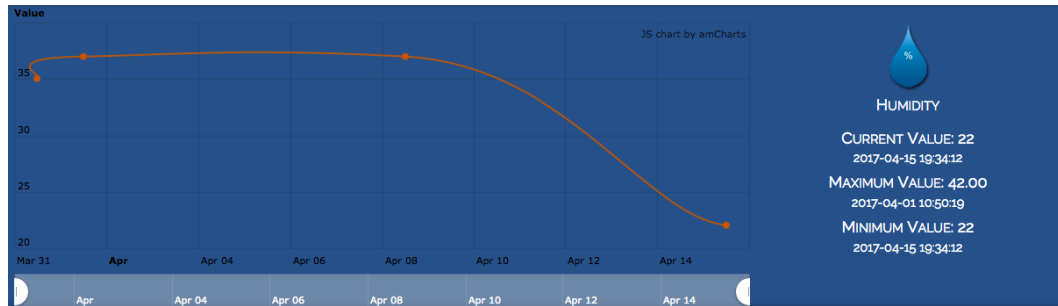


FIGURE 4.6: Bottom section with sensor information

obtained from Open Weather Map API [47], the network state and information about how many unseen notification and rules are set by the user.

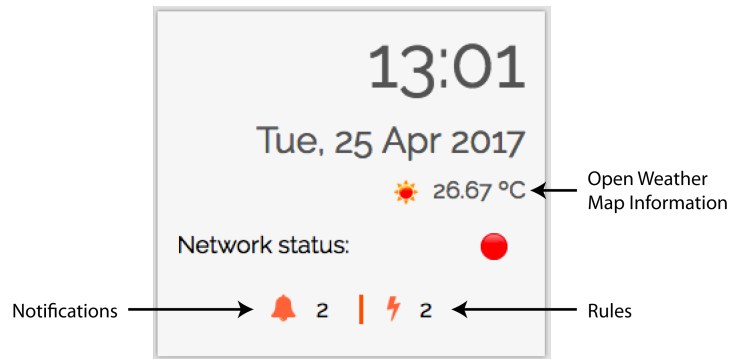


FIGURE 4.7: General Information Cell

4.1.1.2 Rules

When the rules icon is clicked, the bottom section, Figure 4.8, displays a table that shows all the rules set by the user, with the possibility to remove or edit them, and a way to add new rules. To give the user the chance to stop a rule without erasing it, an ON/OFF switch was added to set the rule active or inactive.

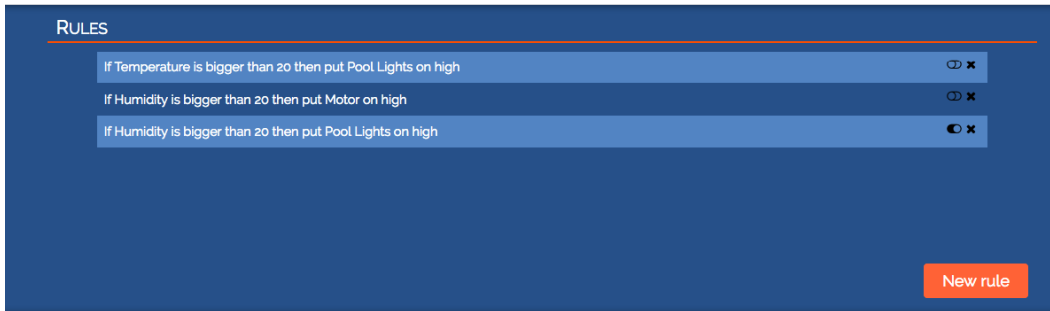


FIGURE 4.8: Bottom section with rules information

In order to create new rules a Bootstrap Modal is used, where the user can choose which sensors will trigger the actions. One by one, the user need to select the sensor, the comparison type and the value. The actions can be of two types:

- i) Actuator changes: user selects, one by one, the actuators and the value to be set;
- ii) Notification: user select either email or platform notification and the title.

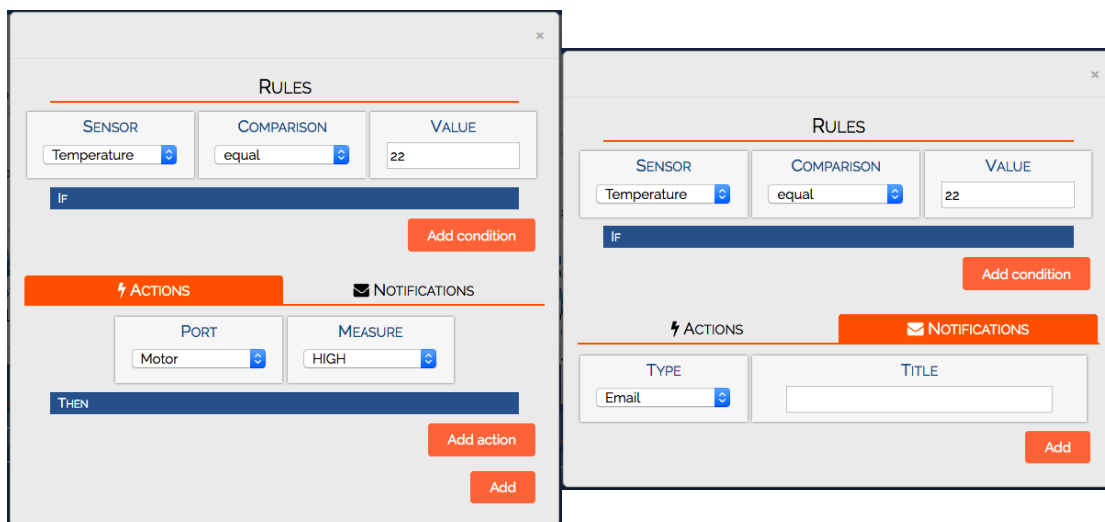


FIGURE 4.9: Rules Bootstrap Modal

4.1.1.3 Notifications

When the notification icon is clicked the user is able to see all the notifications, Figure 4.10.

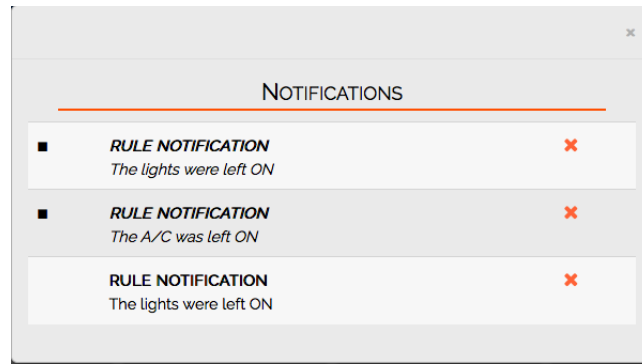


FIGURE 4.10: Notification Bootstrap Modal

4.1.1.4 Personal Area

The web platform also has a personal area, where the user can change his personal information and control what structures are visible, add, edit and remove structures.

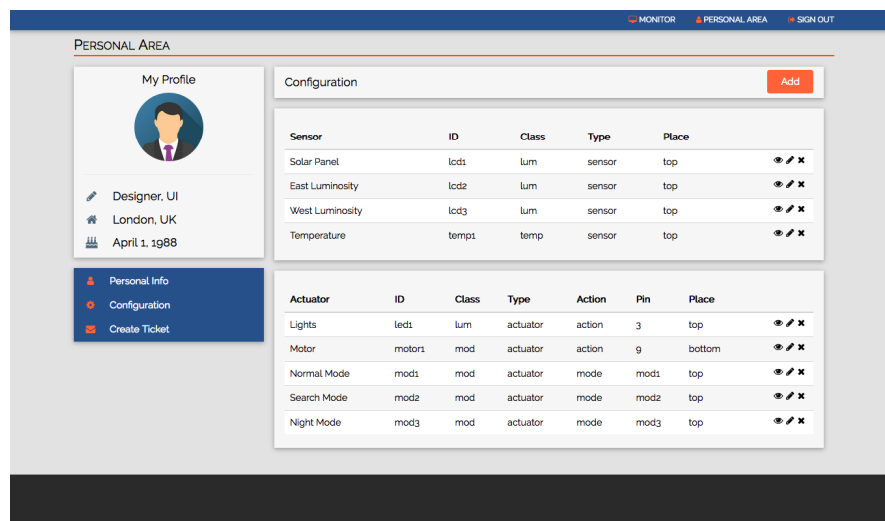


FIGURE 4.11: Screenshot of the personal area

4.1.2 Support Scripts

The support scripts are present in every node of the system and are responsible mainly for gathering, transferring, storing and analyzing data. All scripts were designed using Artificial Intelligence methods in order to provide a generic code adaptable to every situation.

4.1.2.1 Visualization Platform

In the platform a JavaScript script, using the Eclipse Paho JavaScript Client [46], is responsible for supporting the MQTT communication between the user and his platform. It starts by connecting the user platform to the MQTT broker, using the client ID, and subscribing to the unique user topic that will allow him to send and receive information. It was designed to do a specific task pending the message received, as shown in Figure 4.12, and also send to the network MQTT messages with actions and rules as the user create them.

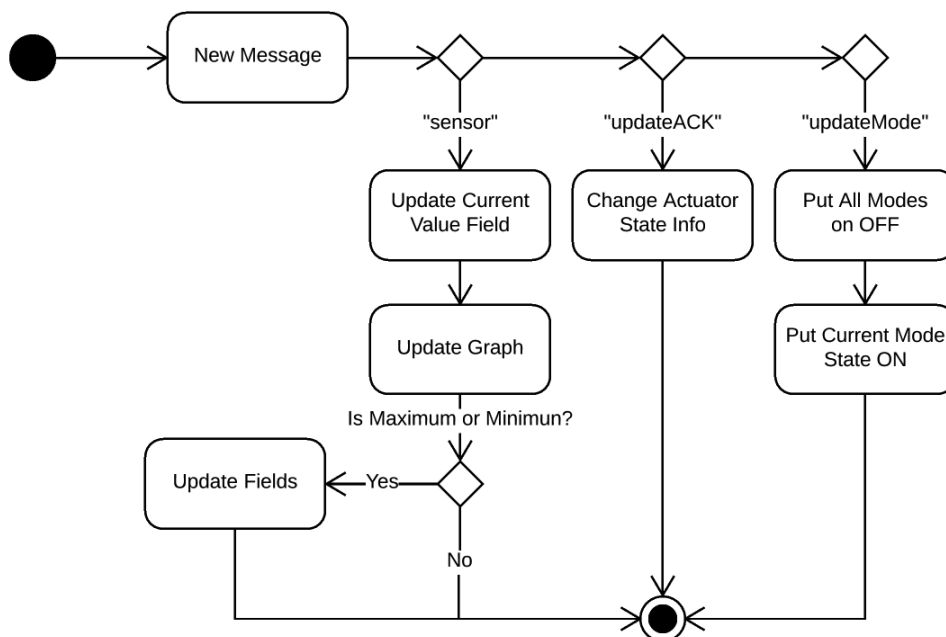


FIGURE 4.12: Visualization Platform Tasks pending MQTT Message Received

To guarantee that the system can offer an efficient workflow, a rule conflict algorithm was created to make sure that contradictory rules were not created or that user can not change the actuator state, without an warning, if a rule or schedule are affecting it. This prevent the user from creating multiple rules that are active by the same sensor combination but have inverse performance or that the user can override a rule without erasing it.

4.1.2.2 Server

In the server, a Python script is used in order to listen to all user network inputs that send sensor data or notifications, using the Eclipse Paho Python Client [46]. These are done here because the client Javascript script is only running when the platform is active, so the data is accumulated in a queue to be delivered when the user re-opens the platform. With this script all data is processed in real time without the need for the client to be active. In the case of notifications, this script is responsible to send the email or platform notification to the user, with the custom message chosen by the user.

Another task performed by this script is the management of scheduled actions. As seen in Figure 4.13, when it is time to perform a scheduled action, the script sends it to the correspondent network and removes it from the schedule. If the action is a repetitive one, the script is responsible to schedule the next interaction. This course of action was chosen to guarantee a more efficient management of the database, putting scheduled actions there only when needed instead of scheduling multiple iteration at once.

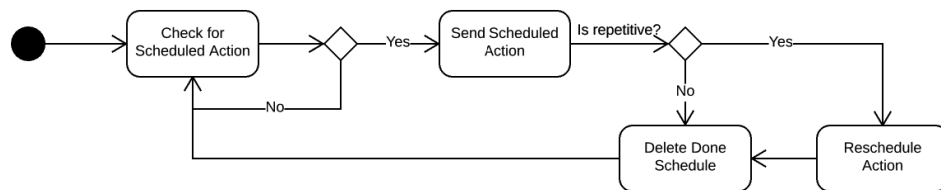


FIGURE 4.13: Scheduled Actions Management

Also in the server it was installed the MQTT broker, to which the web platform and the clients scripts will connect, using the Mosquitto MQTT broker [48].

4.1.2.3 Aggregation Node

In the aggregation node a MicroPython [49] script is responsible for receiving the data from the sensor node and sending them to the server using the MQTT protocol, with the umqtt library for MicroPython [50]. Also this script, using

Artificial Intelligence methods as shown in Fig. 4.14, is capable of comparing the value received from the active rules set by the user and send the appropriate action back to the sensor network when the rule is valid. In order to create a more efficient algorithm and network functionality, the script analyzes the action to implement in the network and only performs them if the current state of the actuator is different from the new one.

Being this a real time system, sensors can be read multiple times per minute, so when a notification rule is set it can be active that many times. To ensure that a SPAM like notification is not being sent, notifications created by rules are only sent in intervals of a minute, or any other time defined by the user, so even if the rule is active by the sensors data the action is not performed.

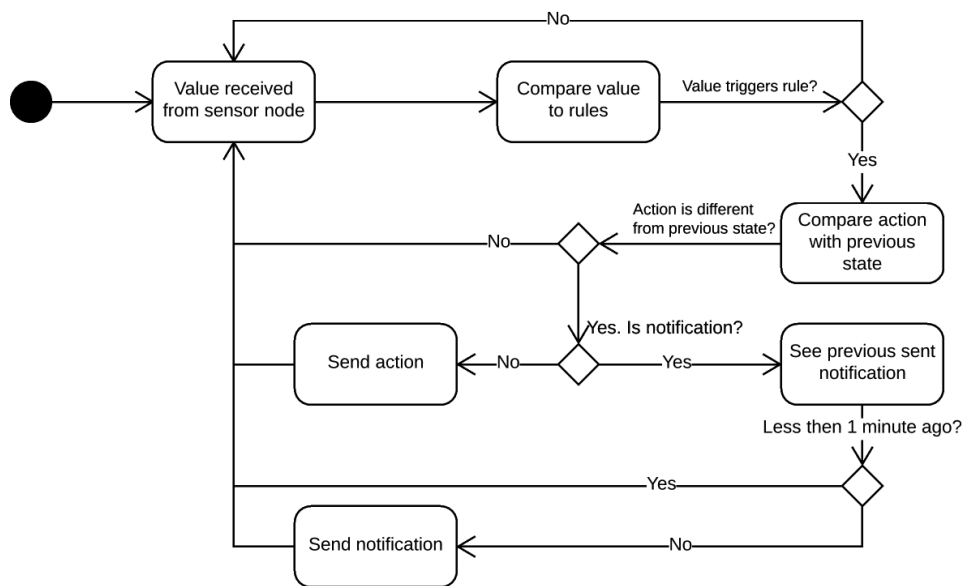


FIGURE 4.14: Value and rules process on aggregation node

To ensure that all data reaches the user, when there is no Internet connection the data is stored and send back to the user upon connectivity is achieved.

The script is also responsible for receiving commands and rules sent from the user through the web platform, routing the action to the sensor node and storing the rules.

4.1.2.4 Sensor Node

The sensor node uses a custom C++ library, built from scratch, that is responsible for transforming generic inputs sent from the platform through the aggregation node into actions on the network, such as turning on and off actuators or switching between functionality modes, responding with an update ACK to guaranty the action took place. It is also in this script that the sensors are read and the information sent to the aggregation node that will forward them to the server.

All these scripts are combined together in order to create the perfect system to build, monitor and control smart environments. The way they combine can be seen in Fig. 4.15 and 4.16, where it is possible to see the process needed to initiate the system, with the authentication of both network and user, including the initial message exchange for the online platform configuration as well as the data and action exchange process in order to send information from the Sensor Network to the user and actions and rules from the user to the Sensor Network.

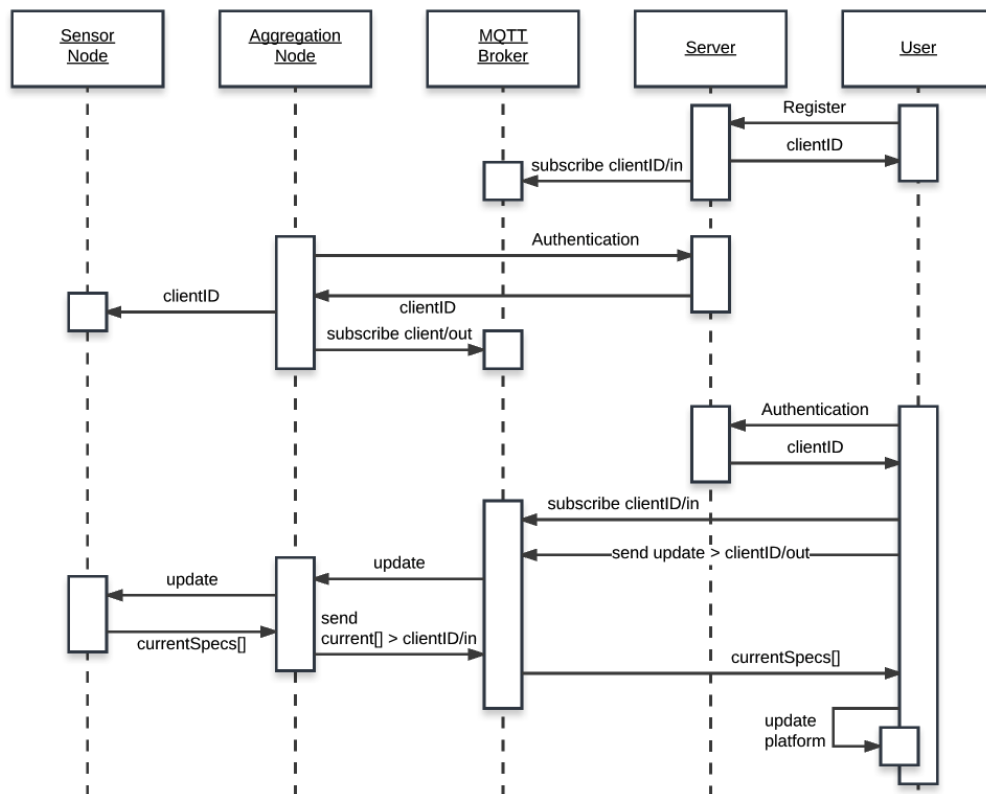


FIGURE 4.15: Network and User authentication process and initial process

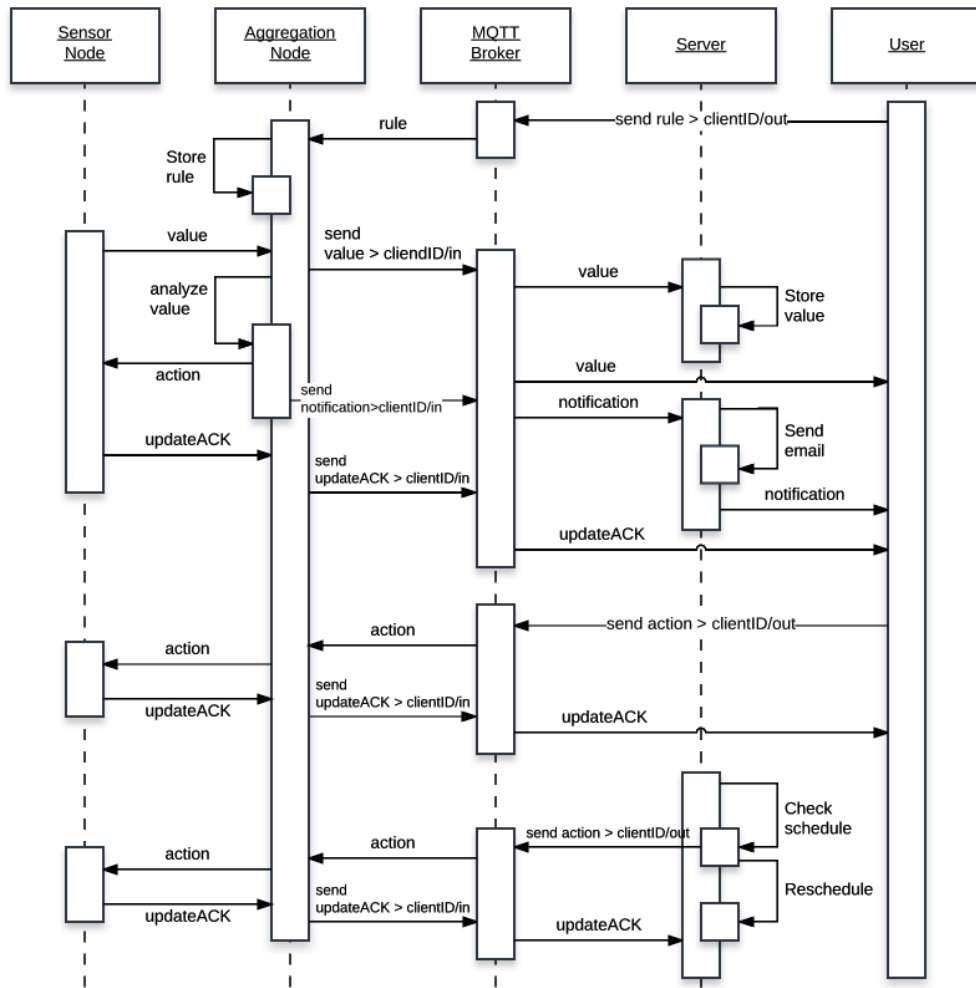


FIGURE 4.16: Rules, Data and Actions exchange

4.1.3 Network Security

Since the system will deal with data exchange, it was important to guarantee that all the data retrieved reaches the destination without being changed, peeked or replaced. To achieve this, all software features include some sort of authentication or encryption.

In the visualization platform a login step, with an unique username and password, is required to access the monitor and control features. Also the connection to the server is done using an encrypted HTTPS connection with a SSL certificate that implements an asymmetric Public Key Infrastructure.

In the communications between the server and the aggregation node, each user

has its own subscription topic using his token, composed by 10 alphanumeric symbols created when the user registers in the system, guaranteeing that only he can listen and publish to that topic thus ensuring the secure exchange of data. Besides this, the MQTT protocol can also implements encrypted communication, adding a second layer of protection to the communication.

The communications inside the sensor network are only encrypted when LoRa is used, since the data exchange is done using an 16-bit private key to encrypt and decrypt the messages. When wired communications are used there is no encryption added, but when needed a base 64 AES symmetric key encryption can be used.

4.2 Hardware

As displayed in Figure 4.1, the hardware for our implementation is composed by an aggregation node or gateway and, at least, one sensor node. The aggregation node is not responsible for reading sensors, it just provides a gateway between the user and the Sensor Network and also performs some data analyses. The sensor node is responsible for gathering information from sensors, perform user actions and use communication mechanisms to send data to the aggregation node.

The choice to have two separate nodes in the network relies on the need to have a single point of connection with the server and the possibility to have multiple sensor nodes communicating. Therefore, in the implemented system, the aggregation node is put on a place where a Wi-Fi connection is available and within communication range from the multiple sensor nodes, that are placed in the actuation site.

The communication protocol, the sensors and actuators are selected according to the environment and user specification, and are addressed in a section ahead.

4.2.1 Sensor Node

The sensor node is the simplest node in the network, requiring only analog and digital I/O pins capable of interfacing with sensors and actuators and a communication mechanism to exchange data and actions. With these in mind and considering the decisions made in the previous chapters and the software developed, an Arduino Uno board is all what is needed to check the requirements.

With size and price being one of the concerns about the system, it is possible to change the Arduino Uno for a Breaduino, Figure 4.17, an Arduino Uno replica built from scratch using common electronics components, including the ATmega328p. This solution has all the features that the ATmega328p provides with the advantage of rearranging the board to our specifications and a lower cost, being the only disadvantage the need to use FTDI to program the chip. But with additional hardware the Arduino Uno capabilities can be matched, such as 3.3V voltage regulator, 9-12V power input or USB programming.

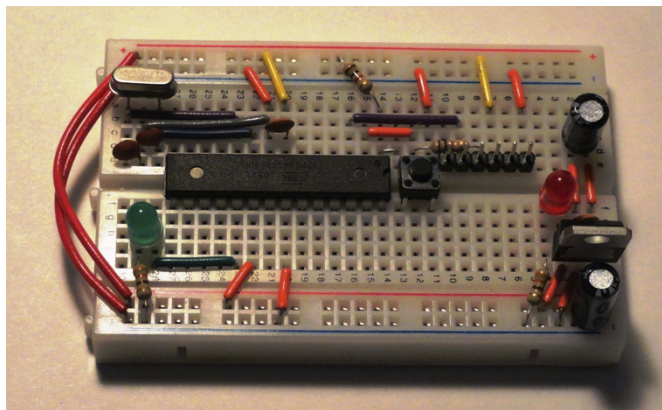


FIGURE 4.17: Breaduino implementation

In terms of communication protocols, the sensor node has to be able to use the RS232 and/or LoRa, that can easily be achieved as shown in Chapter 3.

Since the node was designed to be a part of the actuation zone, mostly in outdoor environments, the input power can be provided from solar power, using a small solar panel and battery capable of sustaining the entire node. When the node is in an indoor environment, the required 5V can be sent using the the Arduino USB port.

A PCB projection for the sensor node built using the Breaduino solution with both communication mechanisms required, can be found in Appendix B.

4.2.2 IoT Gateway

To maintain the normal requirements for an IoT gateway and in order to be able to use multiple communication protocols, adapting them to user requirements, the created solution is a gateway based on common requirements and capable of using every sensor and communication protocol, both wired and wireless, that are available.

From the start we decided that, if possible, we would build our own IoT gateway using standard microcontroller components. To achieve this, bearing in mind the decisions made in the previous chapters and the software developed, we started by understanding the needs for this board. First it will use the Adafruit HUZZAH 8266 as the core processor, enabling the necessary Wi-Fi connection. Second it will need to be able to use RS232 and/or LoRa, so an external UART interface is needed and/or an external antenna for the LoRa chip. Lastly, the gateway must have a way to store data when Wi-Fi connection is not available and also store the configuration files that the user uses to adapt the gateway to the environment and its personal space. For this an microSD card is used, so an SD card reader is needed. Also a way to provide external power to the board is required, with a 5V supply coming from an micro-USB port. Additionally an RTC can be added, if real time clock is needed.

Beginning with the build of a prototype using off the shelf boards. As seen in Figure 4.18, the HUZZAH board was mounted on a breadboard connected to the SD card reader, each powered by USB from a laptop.

Connecting the gateway to the previously stated sensor node using RS232 and LoRa, Figures 4.19 and 4.20, it was possible to test the full system, register and correct some errors found in both software and hardware.

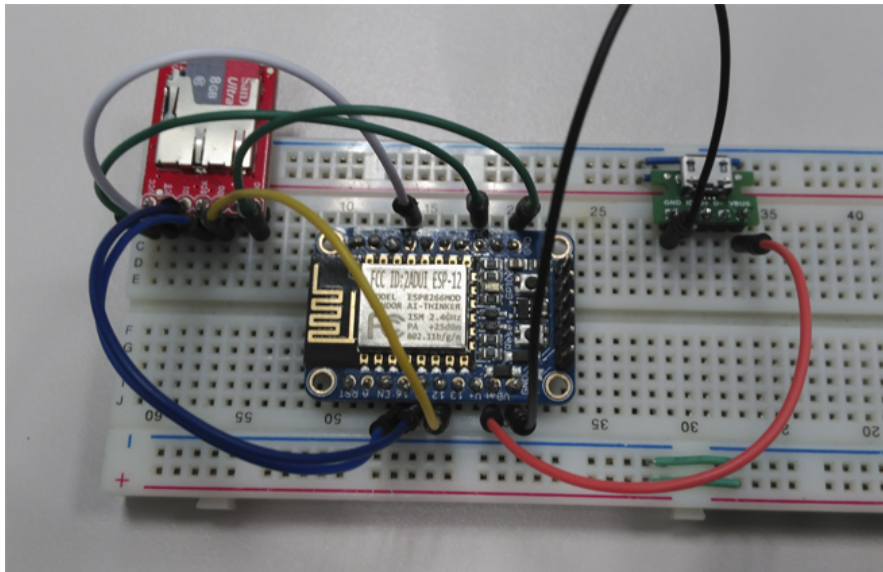


FIGURE 4.18: IoT Gateway first prototype

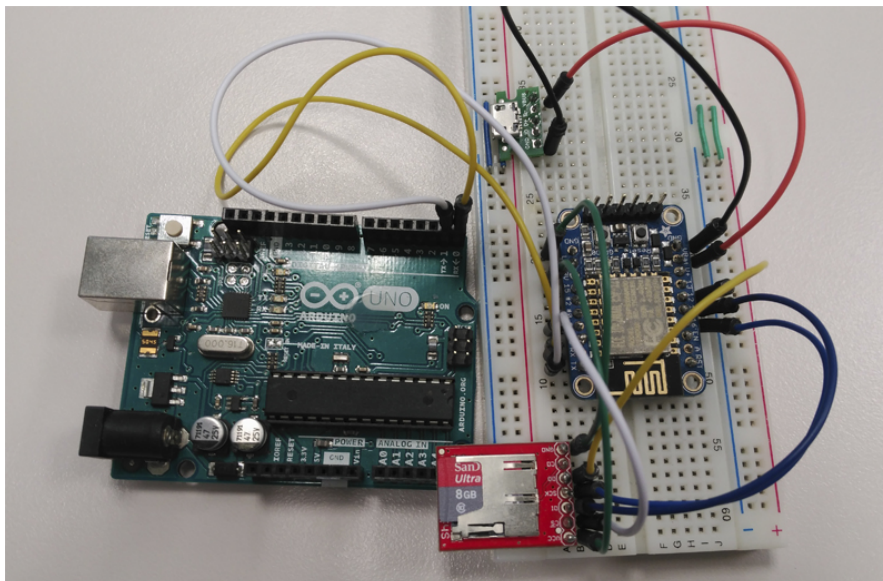


FIGURE 4.19: RS232 network connection

As we advance, after the test done with the first prototype and due to software and communication necessities, such as the MicroPython Software that does not have a library for external LoRa radios, it was necessary to add a second processor to the gateway. With size and price being one of the concerns about the gateway we decided to implement the Breaduino solution into the gateway, since the Arduino Uno processor is capable of performing the needed tasks. This solution allows a reduction in more than half in price and the ability to rearrange the board layout to fit our necessities. The system continues to be powered by the 5V micro-USB

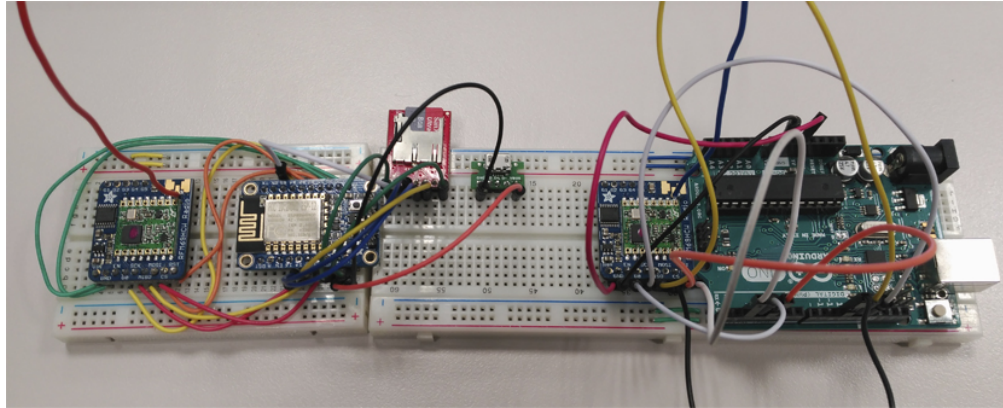


FIGURE 4.20: LoRa network connection

connector, since both processors have a 5V input voltage.

With this changes in mind, a new breadboard prototype was built, Figure 4.21, and tested as the first prototype.

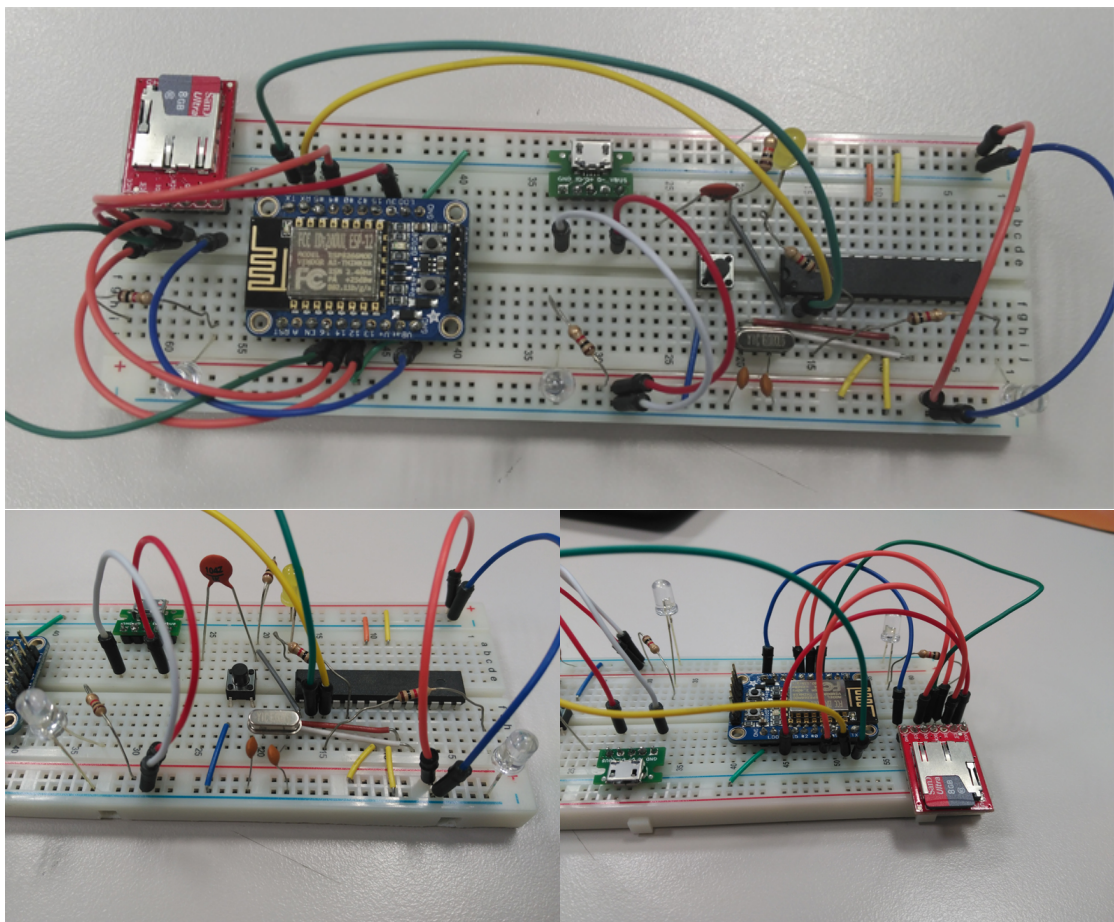


FIGURE 4.21: IoT Gateway second prototype

The need to have two processors on the gateway can be seen as a disadvantage, but taking advantage of these needs, we were able to develop a more powerful with more features gateway. With this, it is possible to transform the gateway into a multi-purpose board, that can act as simply the Aggregation Node or as a single network node combining the Aggregation and Sensor Node in the same board. Also using the ATmega328p it is possible to convert the gateway in an Arduino friendly board, capable of using its libraries, communication protocols, thus making the gateway a more efficient and adaptable board. This means LoRa communication can be a native gateway feature even when using MicroPython with the HUZDAH and also, with the help of headers with the Arduino Uno layout, the gateway is capable of using all Arduino Shields, increasing even more the features that can be added to the board, like ZigBee, GSM, Motor control, among many others.

After testing the new prototype and with all the specifications concluded, a new prototype was built on a PCB as seen in Figure 4.22. This prototype includes the two processors connected over RS232, with jumpers that can be taken when is necessary to program the board or if the Sensor Node is connected with RS232, not needing the second processor; with a prototyping area that can be used to implement LoRa communication or sensors and actuator; and the necessary power and SD card modules.

Also a miniaturized version of the board, without the prototyping area and the Arduino Uno layout, was designed and can be seen in Appendix B.

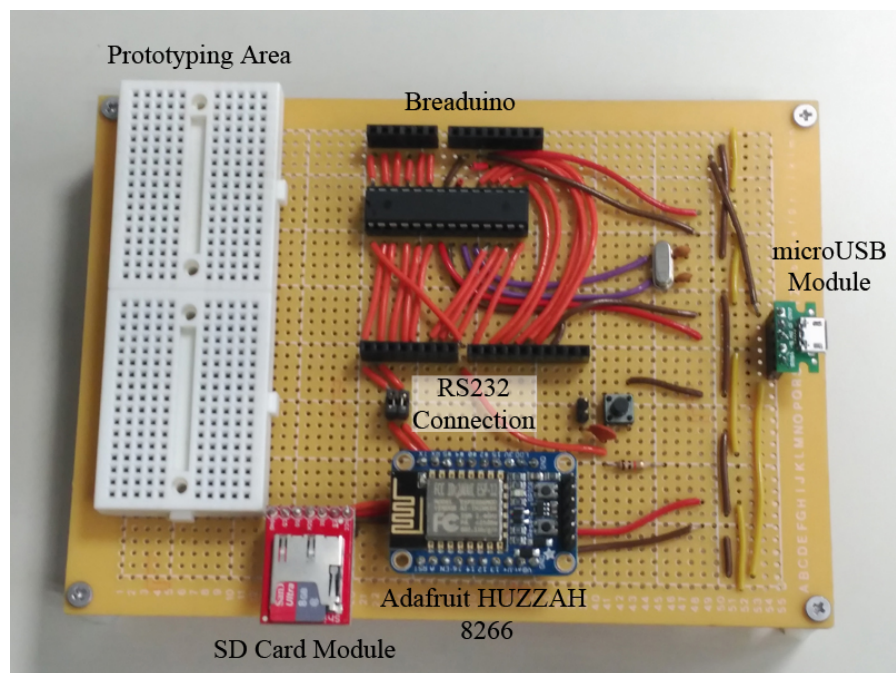


FIGURE 4.22: IoT Gateway PCB implementation

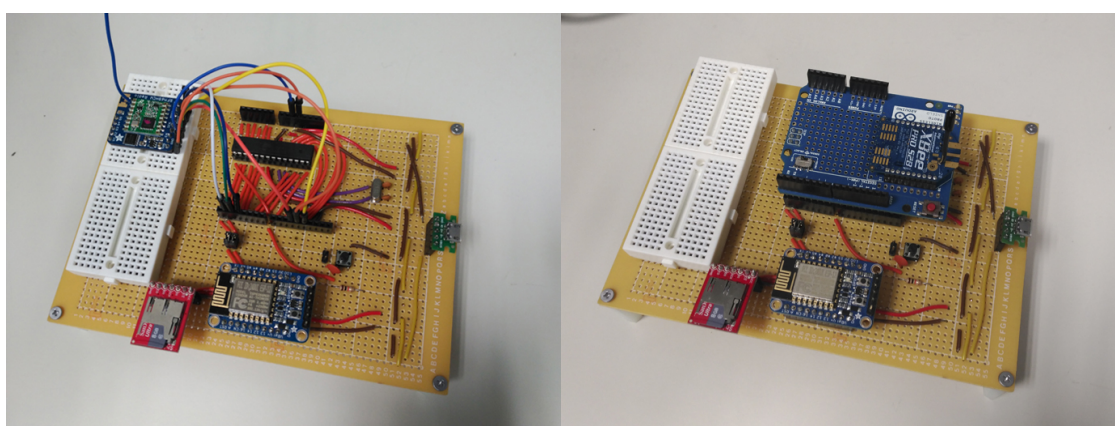


FIGURE 4.23: IoT Gateway with Wireless Communication (Left: LoRa; Right: ZigBee)

Chapter 5

Application Scenarios

Nowadays the user wants to be able to control everything from a mobile or Internet connected device and this leads to the need to create autonomous environments that facilitate the user day-by-day experience.

As said before, this project can be applied to any object or environment in order to improve efficiency and user experience. The project allows an impact in any environment, anywhere in the world, where there is the need to automate repetitive manual tasks and also a control over budgets, eg. in energy or water, from a circular economy perspective. It can provide a technological approach to situations that always exist, creating modern services and processes that lead to an excellent performance and reduce economic resources to the final user, in both materials and labor.

The developed system can be implemented in several situations, with multiple different purposes, being the main focus the following:

- Home and Business Use:
 - Room/Space monitoring;
 - Gardens;
 - Swimming Pools and Water features;

- Agriculture:
 - Harvest monitoring;
 - Irrigation control;
- Renewable Energies:
 - Monitoring and control;

5.1 Implemented Environments

In order to prove the practical functionality of the developed system it was implemented in several different environments. Each environment tested shows at least one of the developed functionalities. Also for each environment the visualization platform was adapted to accommodate the used sensors, actuators and rules needed. The platform setup configuration process can be seen in a tutorial video at <https://www.youtube.com/watch?v=qqbaITyVKsM>.

5.1.1 Solar Panel Control

The first implemented environment aims to improve the efficiency of a solar panel. To achieve maximum capability on harvest energy with a solar panel, it must have full sun exposure at all time. Most solar panel projects are designed with fixed positions, so the panel does not move according to sun position, making its efficiency lower at certain times of the day.

To improve this environment, a sensor node was included in the solar panel as well as a rotor to spin the panel as needed. The goal was to monitor the energy created by the solar panel and move it according to the sun position. Since a Solar Panel was not available at the time, a LDR sensor was used, attached to Pan&Tilt controlled by servos. LDR [51] is a Light Dependent Resistor used to check luminosity. Composed with two cadmium sulphide photo-conductive cells,

that can imitate the spectral responses of an human eye or a photovoltaic cell, in which resistance falls with increasing light intensity, providing 100Ω for 20000 LUX, corresponding to an entire blue sky, 400Ω for 1000 LUX, corresponding to an overcast day, and $9k\Omega$ for 10 LUX, corresponding to no sun light.

Since the sensor node and the aggregation node are separated, the communication between them was achieved using LoRa. Figure 5.1 shows the layout of the implemented solution.

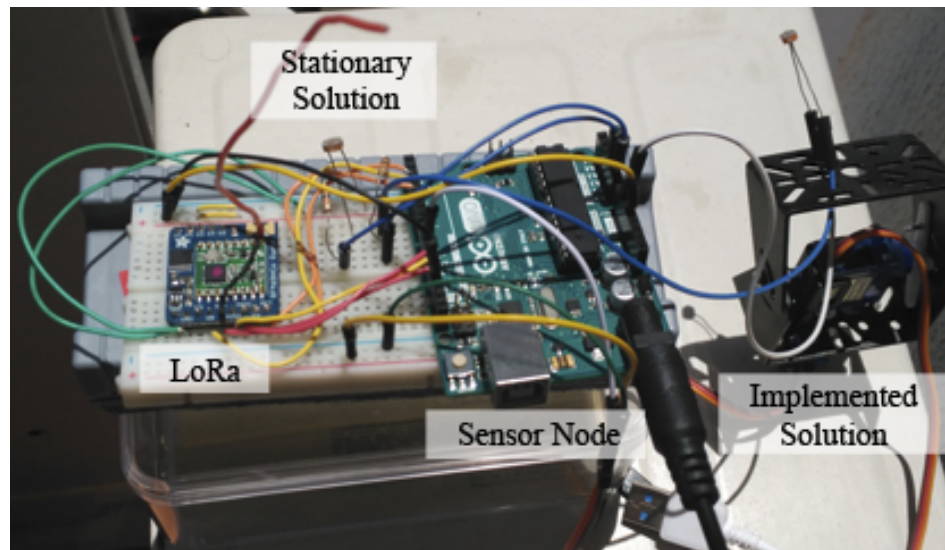


FIGURE 5.1: Solar node implementation

To automate and improve the efficiency of the implementation, a set of function modes were programmed.

1. Normal Mode: Check the energy created by the sun every 10 minutes;
2. Search Mode: Move the solar panel from left to right searching for the best position;
3. Night Mode: Fixed position, checking energy created every hour;

To test the implemented environment, a second LDR was added to act as a stationary solar panel and to compare the values between both scenarios.

As said before, the LDR has a resistance level according to the amount of the exposed light. With that in mind, a threshold of 350Ω was defined as the bottom

value to get a good energy harvest from the solar panel, being that values higher than that will indicate an inefficient situation.

To the LDR that uses the developed system to improve its workflow, 3 rules based on the developed modes and threshold defined were applied:

1. If LDR resistance is bigger than 350Ω then switch to Search Mode;
2. If LDR resistance is less than 350Ω then switch to Normal Mode;
3. If LDR resistance bigger then 600Ω then switch to Night Mode;

5.1.1.1 Results

The system was tested throughout 7 hours and the results can be seen in Figure 5.2, with 88 values retrieved.

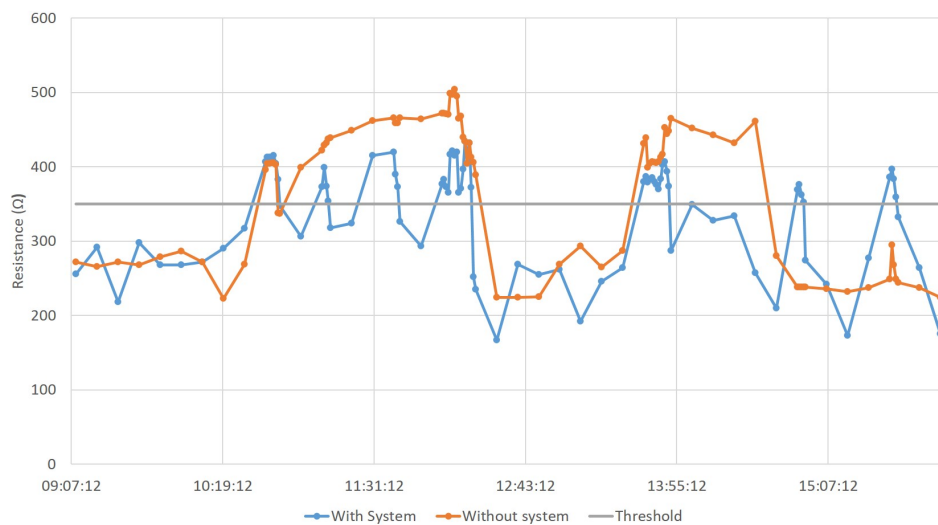


FIGURE 5.2: Solar Panel results

The first thing to notice is that in the 7 hours it was expected to retrieve 42 values in Normal Mode, but only 37 were obtained. This shows that on the remaining 50 minutes the system works on Search Mode, so in this time the solar panel was not getting the correct amount of solar exposure. Of these, the LDR with the implemented system was under 350Ω on all 37 values while the stationary LDR only 34 where under 350Ω . Taking in consideration the values obtained, it

is possible to calculate the average resistance capture by both sensors, with the stationary LDR averaging 368Ω and the implemented system LDR with 340Ω . Although the difference is not large, the implemented system was able to get an average below the threshold unlike the stationary system. Another thing that was possible to conclude is that, when comparing each value from both systems, the implemented one was able to get a lower resistance value in 80% of cases.

A laboratory demonstration video of this implementation was made showing the automation of the system, with real time data visualization and value based switch of functionality modes according to rules, can be seen at <https://www.youtube.com/watch?v=aW-1m99Bz8g>.

5.1.2 Class Room Control

The second implemented environment aims to improve the energy consumption of a class room, or any other room. One of the most common scenarios after a class is to left the room lights or the air conditioner turned on, wasting unnecessary energy.

To improve this, the IoT Gateway working as a single node was included in the room, with a motion sensor, to see if anyone is in the room, a LDR, to check if lights are on and a temperature sensor, as seen in Figure 5.3. The objective was to monitor the sensors attached and send a notification to the teacher or security guard, when the room is empty and the lights were left turned on. To make sure that the node uses the energy provided by the environment, it was connected to a room computer that is always turned on and can provide the necessary energy.

The implementation was done in one of the laboratories of Instituto de Telecomunicações at ISCTE-IUL, with the motion sensor view demonstrated in Figure 5.4.

To automate and improve the efficiency of the implementation, a set of function modes were programmed.

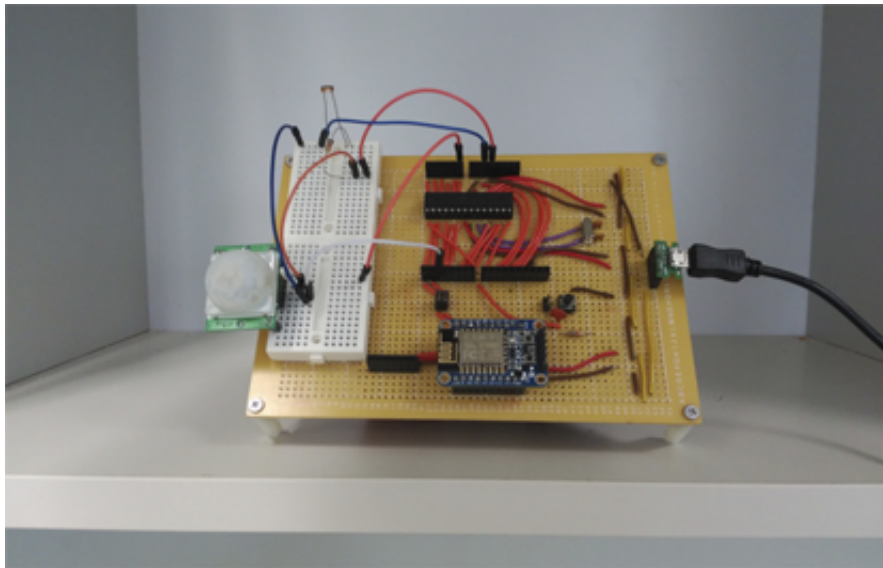


FIGURE 5.3: Room Control scenario



FIGURE 5.4: Motion sensor view

1. Normal Mode: Check the sensors every 10 minutes;
2. Off Mode: Turn the system off, to use in situations like tests where there is little movement or sound although people are in the room;

One thing to consider is that the movement sensor has to work every second to guarantee the correct functionality, giving 0 is there is no movement from the

previous iteration and 1 otherwise, and the LDR gives back a resistance of around 100Ω for an artificial light. With these conditions, 2 rules were defined to improve the workflow of the environment:

1. If Movement is 0 and LDR resistance bigger then 300Ω then send an Email;
2. If Movement is 0 and Temperature less than 20°C then send an Email;

5.1.2.1 Results

The system was tested throughout 7 hours and the results can be seen in Figure 5.5, with 41 values retrieved.

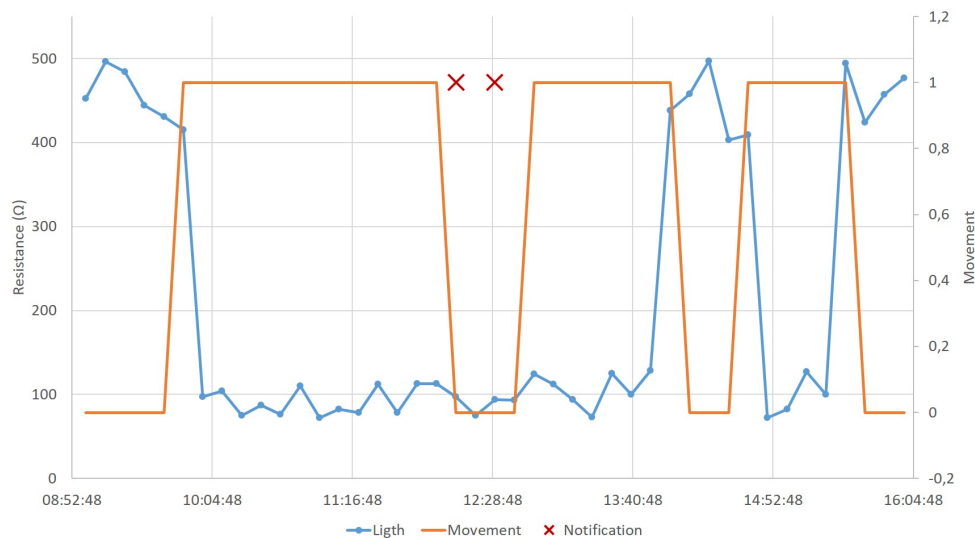


FIGURE 5.5: Room Control results

With these results it is possible to understand the use of the room throughout the day, with 4 moments of emptiness. It is also possible to see that when movement is first detected a change in lights occurs.

Bearing in mind the purpose of the system, it is possible to see that on the 4 periods of time when no movement was detected, only one of the lights was left on. At that period, of about 30 minutes, two notifications were sent with an interval of 20 minutes or two sensor iterations. As specified before, the notifications are not sent in every iteration to avoid a SPAM like functionality and allow the user

to correct or change something in the environment. In this case, it was possible to see that the notification was ignored, since the lights were not turned off. On the other periods, as expected, no notifications were sent since the data retrieved did not activate the rule.

5.1.3 Vertical Garden

The third implemented environment consisted on replacing the repetitive task of watering the plants of a vertical garden. To maintain the vertical garden it is necessary to water the plants on a daily basis, something that requires someone to do it every day. Replacing this task with the addition of the IoT Gateway acting as a single node it is possible to use the schedule functionality to water the plants at a certain hour, for a certain amount of time, every day.

Figure 5.6 shows the implementation, where a water pump was added to get water from a reservoir to the plants.



FIGURE 5.6: Vertical Garden implementation

To automate and improve the efficiency of the implementation, a schedule action was defined to turn the sprinklers ON for 10 seconds every day at 10:00 and 18:00.

To improve even more the functionality, with the addition of a water level and soil humidity sensors, 2 rules can be defined to improve the workflow of the environment:

1. If Humidity is less than 30% then turn the sprinkler ON;
2. If Water Level is less than 20cm then send an Email;

5.1.4 ISCTE Satellite Station

To prove that the system can really adapt to any situation, an already automated environment was chosen to do the final implementation.

Recently the ISCTE Satellite Station was updated to allow monitor and control over a web platform in order to control the antenna rotor and radio [52]. The current implementation of the system, despite being fully functional, has some efficiency problems trading data between the Raspberry Pi and the server due to the use of standard POST and GET from HTTP, using them also to recursively asking the server for new actions from the user, making it a poor real time system.

As seen in Figure 5.7, it is possible to notice that a similar architecture can be found in terms of software and hardware.

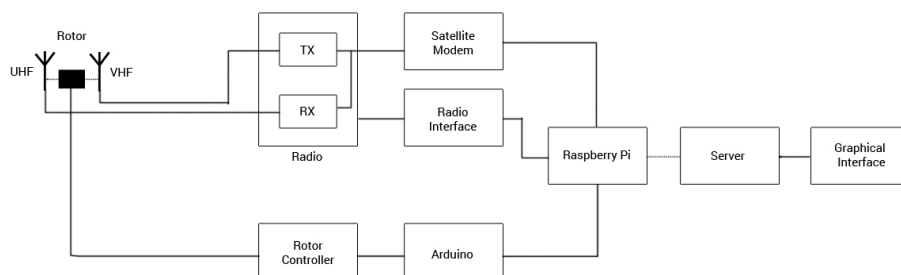


FIGURE 5.7: ISCTE Satellite Station Architecture [52]

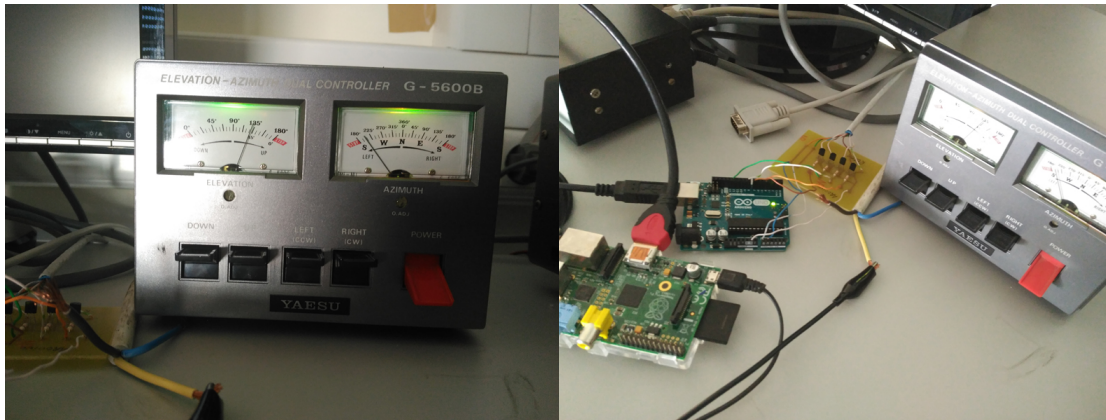


FIGURE 5.8: ISCTE Satellite Station Hardware

Figure 5.8 show the implemented environment.

To improve the station system functionality, without making major changes, some key features from our project were added:

- i) The visualization platform was kept, with minor changes in order to implement our JavaScript script to implement MQTT communication;
- ii) Since the system needs a Raspberry Pi to run the *Predict* script, it was turned into the Aggregation Node for the system. Converting the MicroPython script, from the ESP8266, into a Python script that includes the functionalities from *actionScript* and *dataScript*, the previously created scripts, it is possible to improve the data exchange using MQTT instead of HTTP;
- iii) On the system Arduino, our custom library was added including the previous specifications.

With these modifications, the system was able to have a proper real time functionality, with major saves in both bandwidth and energy.

5.1.4.1 Results

Regarding energy, as seen in Figure 5.9, after the application of our system the CPU usage has a big improvement, using less 75% of the previous system load in

both idle, where the system is no longer asking the server if there is new actions every 5 seconds with a HTTP GET, and performing actions, where there is no longer necessity to request the action and to confirm it using HTTP POST. When calculating the antenna positions to track the satellite, the improvement was only in getting the satellite to track, not asking if there is a new satellite to track every 5 seconds, leaving the rest of the process exactly as is, with the use of *Predict*, that requires more computational power, thus having a lower improvement. Since the Raspberry Pi carries a lower computational load, the system will have a much better energy performance.

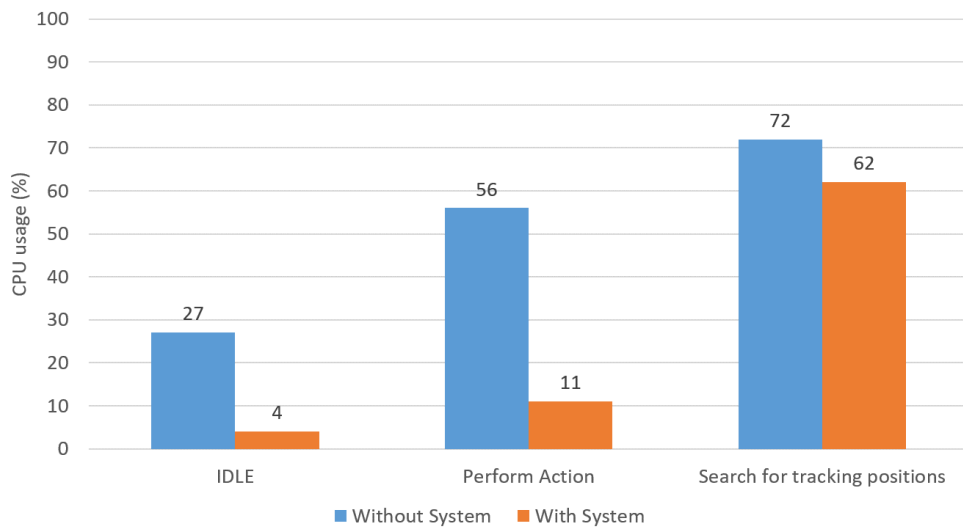


FIGURE 5.9: CPU Usage comparison between systems

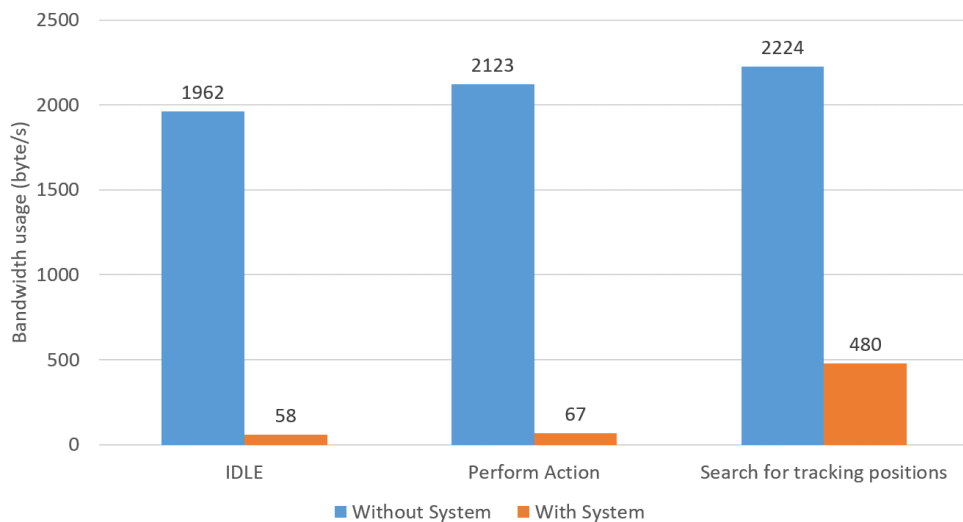


FIGURE 5.10: Bandwidth Usage comparison between systems

As for bandwidth, Figure 5.10 shows that after the application of our system the consumption had a big improvement on every scenario, using less 97% bandwidth in both idle and performing actions and less 80% when calculating the antenna positions. Once again, replacing the recursive HTTP POST and GET to the server with MQTT implies a huge improvement on the system.

A video of this implementation can be seen at https://www.youtube.com/watch?v=avqQRqvX_Ek.

5.2 Other Environments

Besides the implemented cases, to prove the viability of the project more environments were planned, although they have not been tested.

The first one was for a swimming pool, with an application capable of monitoring the water temperature, level and circulation, the environmental temperature, relative humidity, air pollution and luminosity and remotely controlling the water pumps and pool lights. Also keep track of machine room environment, for example sending an notification to the user when the water level in the room exceeds a certain level. To do this some rules and function modes are already specified, for example a mode where only the sensors are active, a maintenance mode that is active when dirt or low water level is detected, or a hibernation mode, for nights and winter. With this, a better control of water and power consumption could be achieved as well as a more efficient user experience.

The second one was for agriculture harvest areas, being capable of monitoring the environmental temperature and humidity as well as the soil humidity. This combined with a rain sensor can achieve a better irrigation and addition of substances timing, improving not only the plants health but also the water consumption. With the system it was easy to monitor and control multiple species and automatically adapting the action according to geographic zones and conditions, improving not only the farmer job but also the water and material consumption.

5.3 Discussion

In this chapter we focused on demonstrating the practical functionalities of the developed system using real case scenarios. The system was implemented in four different environments, each chosen to show at least one of the features designed to improve automation, efficiency and versatility.

The solar panel implementation shows that a rule based operation working with real time data gathering allows the improvement of the sun harvest operation, with better results obtained in 80% of the time when facing a stationary panel. Although the average sun exposure results are only slightly different, when the system was applied it was kept under the defined threshold something that does not happen with the stationary solution. To improve this result, the rule to exit the Search Mode could be changed to a value lower than the threshold, meaning that it only stop when a value better then the necessary minimum is achieved.

On the room control, even with the users not acting upon the notifications, it is possible to see that a warning situation when the lights were left on in an empty room could help to reduce the power consumption. To improve this environment, a notification could also been sent to the security guard or building management or, using the automation part of the system, include the building lights in the system and use a rule to turn the lights off if the room is empty.

The schedule based operation replacing repetitive manual tasks has been proven to be advantageous with the vertical garden scenario.

The previous results for MQTT over HTTP were once again proven, this time in a real case scenario, with an improvement of 75% for CPU usage and 97% for bandwidth consumption when implementing our system over the old Satellite Station system, without losing efficiency and functionalities.

It was possible to demonstrate the versatility with the system being applied in every environment, even if there was already an automate solution, with slight modifications done between cases, only at programming the sensors as well as

the hardware implementation of sensors and actuators. As for the Visualization Platform, with the easy configuration process, it was adapted to each scenario.

Chapter 6

Conclusions

In this dissertation we demonstrated how to design and develop an IoT system based on sensor networks and low-cost devices that allow full remote control over objects or environments turning them into smart ones. The system, that was developed to achieve automation, efficiency and versatility, was divided in three specific modules that together create a system able to fulfill any requirement: (i) Software, with an online visualization platform and nodes adaptive software; (ii) Hardware, with the the network nodes architecture design; and (iii) Communications, that connect the previous ones.

Since the network depends on communications and devices, we have evaluated the best solution available through a set of specific tests to get the right ones for our specifications, including flexibility, availability, energy efficiency and implementation simplicity. The RS232 showed to be a better choice when wired protocols are needed while LoRa was the best for wireless protocols. Also MQTT was proved to be a better solution when facing HTTP POST. As for devices the Arduino Uno and ESP8266 proved to be the best ones for the network nodes.

The designed system architecture relies on a visualization platform as a way to monitor and control the sensor network and although there were already many solutions available, none fitted our requirements or had all the features desired so a new one was built from scratch, capable of adapting to any specification and

using Artificial Intelligence algorithms to improve the user experience. The same methodology was applied to all the software developed for the network nodes.

The sensor network depends on the capability of gathering and transporting information and for that a set of boards designed specifically for an adaptive system were built. It was also possible to demonstrate that a multi-purpose board, capable of interfacing with almost every sensor, actuator and communication protocol, can be built using low-cost IoT devices.

The potential of the developed system was tested on multiple environments, with the results showing that the system can indeed achieve automation, efficiency and versatility. The automation, with a schedule watering functionality in a vertical garden shows that the system can replace repetitive manual tasks. The improve of efficiency can be seen in the Solar Panel implementation where, when comparing a case with and without the system, it is possible to achieve a better solar exposure on 80% of the time. Versatility was proven with the variety of different implemented environments with even the possibility to implement it in previously automated systems, as shown with the ISCTE Satellite Station.

There are big challenges when implementing IoT projects and the proposed solution is another one to help improve and overcome the challenges. With a low-cost solution, capable of adapting to every application with little or none changes, using any sensor and various communications protocols it is possible to reduce the price of a typical solution and surpass the lack of standards for different applications within the same system, thus proving to be a solution capable of creating and sustaining smart environments with real time monitoring and control.

6.1 Future Work

Despite a complete system being described with all presented functionalities being implemented and tested some features can be added to improve even more the overall performance, in both software and hardware.

The implementation of more Arduino functionalities to the boards, such as USB programming or OTA updates, and production of printed PCB and cases, are some things that can be done on the hardware side. For Software, the development of a mobile application, the possibility to connect our system to other cloud based platforms such as Amazon AWS IoT or Cayenne through an API, or programming modes in the platform, are functionalities that can be added. Also more implementations can be tested to prove even more the practical functionality.

Appendices

Appendix A

Communications Tests Assembly

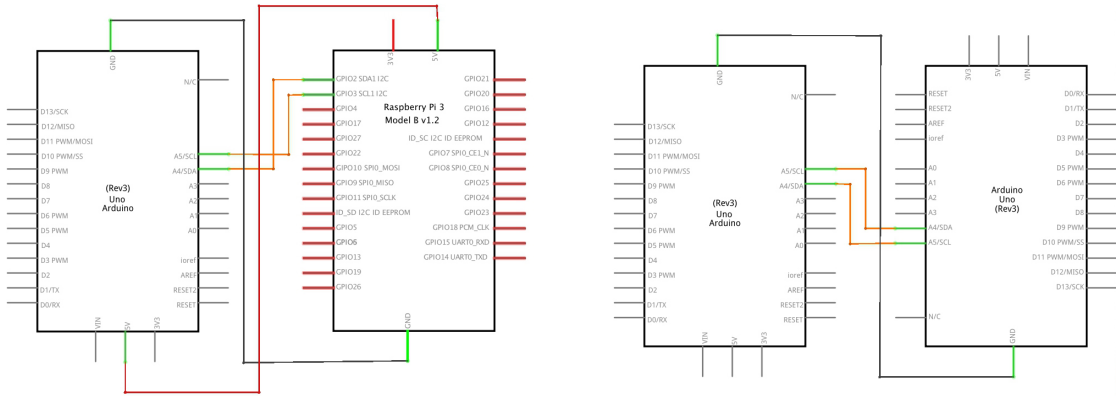


FIGURE A.1: I2C connections

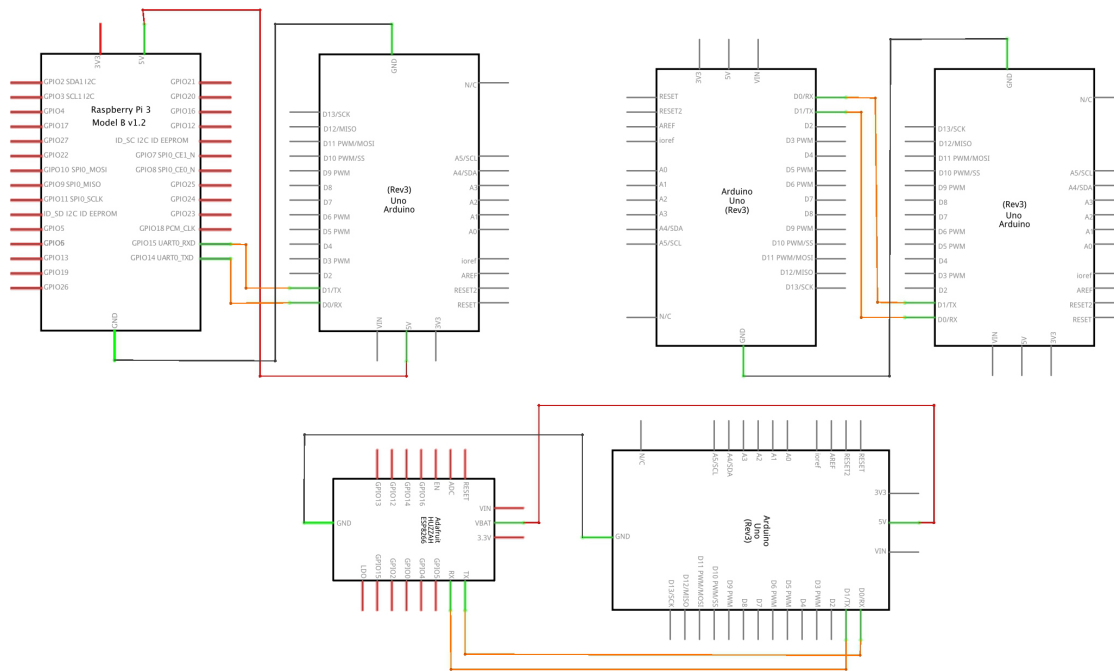


FIGURE A.2: RS232 connections

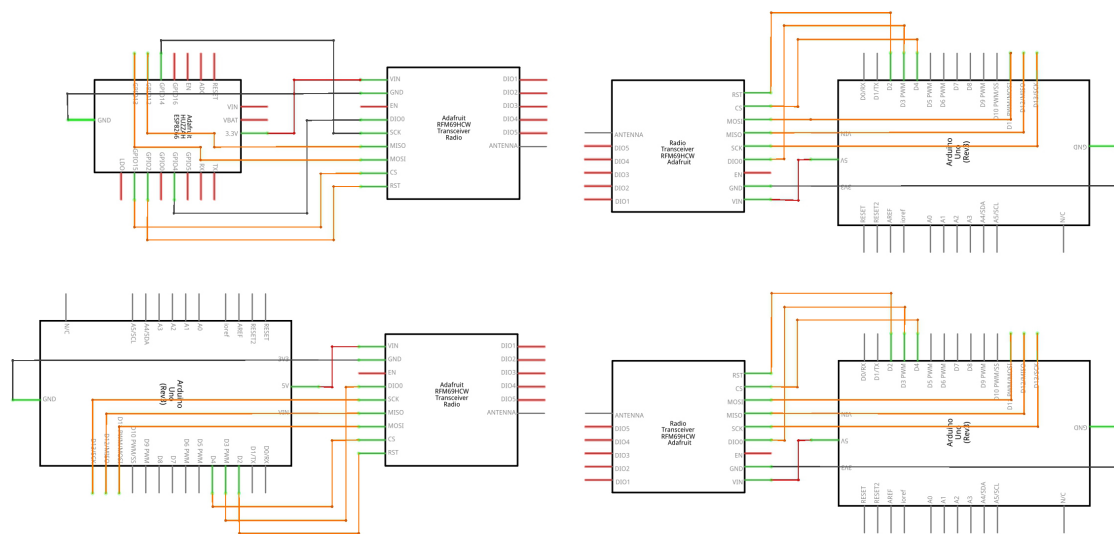


FIGURE A.3: LoRa connections

Appendix B

Nodes Assembly

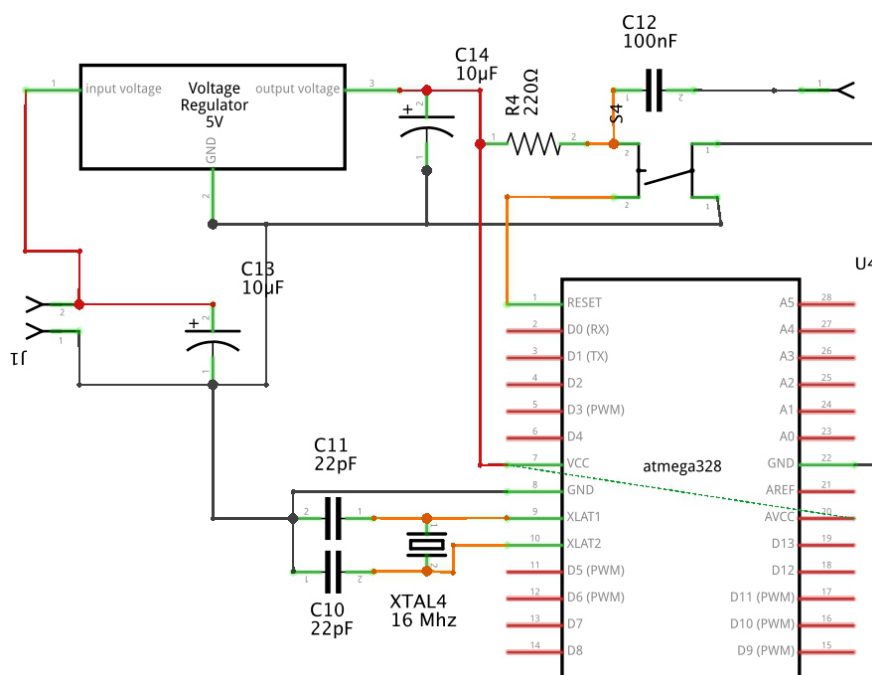


FIGURE B.1: Sensor Node Connections

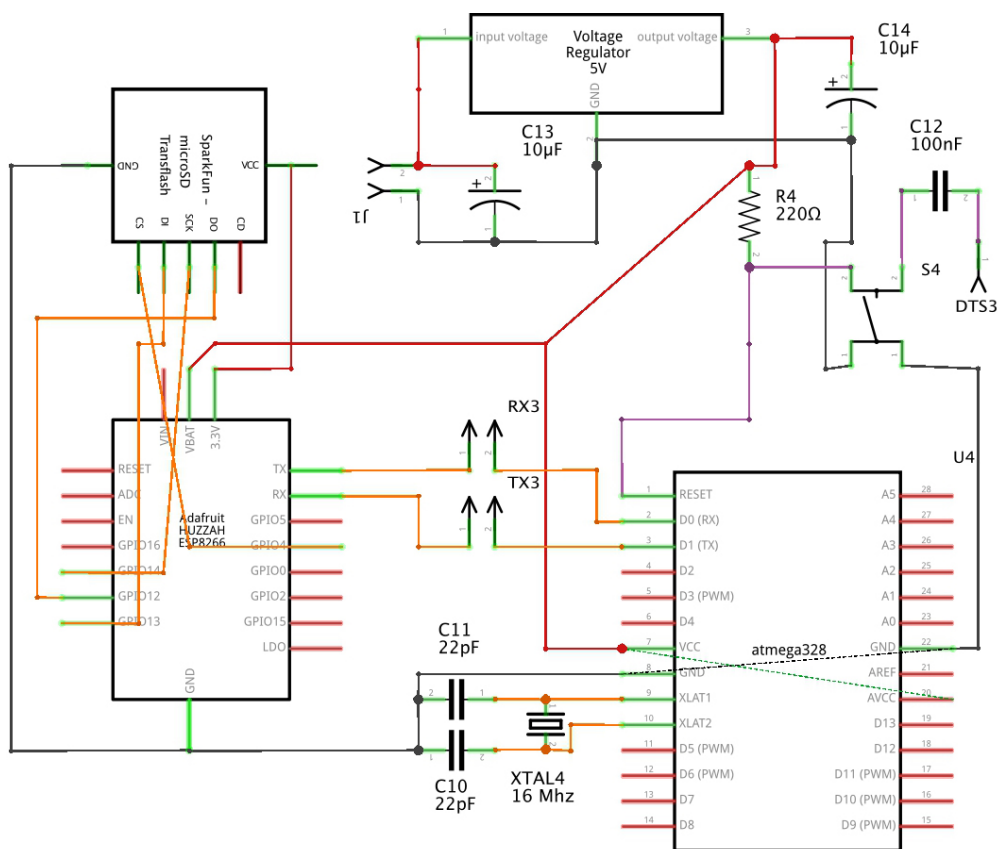


FIGURE B.2: IoT Gateway Connections

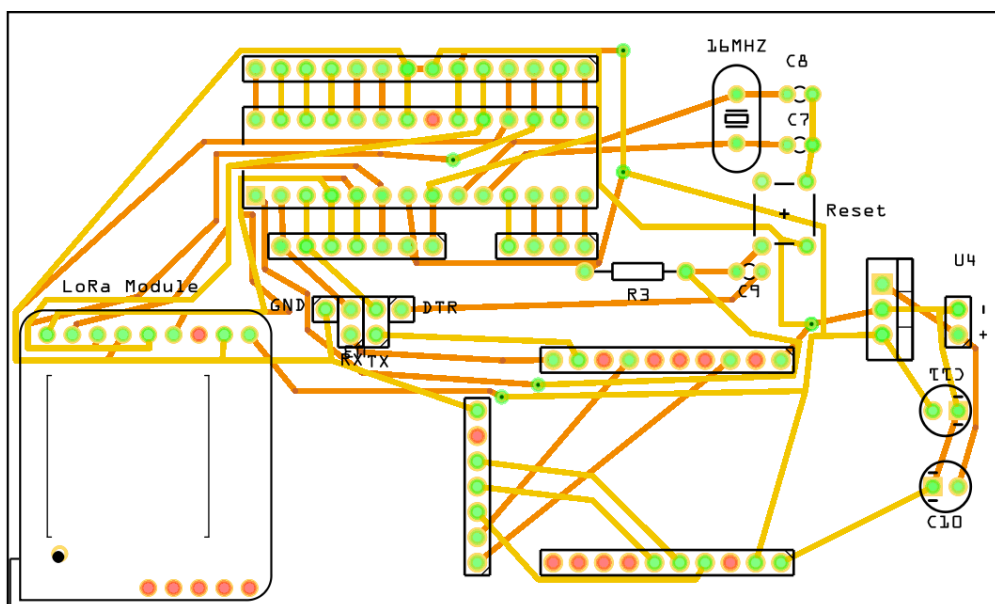


FIGURE B.3: IoT Gateway PCB Projection

Appendix C

Scientific Articles Published



Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 109C (2017) 568–575

Procedia
Computer Science

www.elsevier.com/locate/procedia

The 8th International Conference on Ambient Systems, Networks and Technologies
(ANT 2017)

Design and implementation of an IoT gateway to create smart environments

André Glória^{a,*}, Francisco Cercas^{a,b}, Nuno Souto^{a,b}

^aISCTE-IUL, Av. das Forças Armadas, Lisbon, Portugal

^bInstituto de Telecomunicações, Av. Rovisco Pais, 1, Lisbon, Portugal

Abstract

The paper presents a proposal of a practical implementation for an IoT gateway dedicated to real-time monitoring and remote control of a swimming pool. Based on a Raspberry Pi, the gateway allows bidirectional communication and data exchange between the user and the sensor network implemented on the environment using an Arduino.

1877-0509 © 2017 The Authors. Published by Elsevier B.V.

Peer-review under responsibility of the Conference Program Chairs.

Keywords: Internet of Things; Sensor Networks; Smart Environments

1. Introduction

Nowadays the urge to connect everything to the Internet is growing, not just to send information to servers for processing and storage but also to provide full control of physical devices over the web.

While humans will continue to connect their devices to the Web in greater numbers, by 2020 more than 200 billion smart devices are expected to be connected to the Internet¹, making Machine-to-Machine (M2M) communications up to 45% of the whole Internet traffic^{1,2,3}.

Examples such as Smart Homes, where users can control their thermostats or lights with a smartphone, are the basis for Internet of Things (IoT). IoT was designed to play a great role improving our quality of life and its applications are present in many of our day to day experience such as transportation, health care and industrial automation.

IoT has the ability to transform a simple physical device into a smart one, using the embedded technology and computational power. Using the sensors and actuators available to guarantee the features of the device, it is possible to share that information between devices and put them to work together to improve the user experience. This will contribute to a bigger explosion coming from things connected to the Web that were not connected before, did not exist, or now use their connection as a core feature.

* Corresponding author.

E-mail address: afxga@iscte-iul.pt

The Internet is one of the most important developments of man kind and IoT will represent the next evolution of the Internet^{2,3}. With the capability of gathering, analyzing and distributing the data, IoT consists in the connection between the Internet and a range of devices and sensors.

IoT, as shown in Fig. 1, can be divided into six elements⁴ that help us understand its real meaning and functionality, i.e. *identification, sensing, communication, computation, services* and *semantics*.

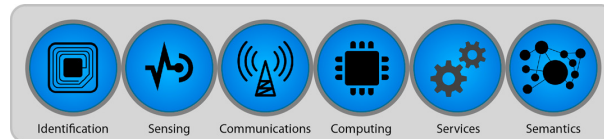


Fig. 1. IoT elements

IoT projects have the ability to do more than just connect the device to the Internet, they can be a big part of improving the efficiency or even adding new features such as Artificial Intelligence, transforming every common objects into connected one.

In this paper the authors describe the designing/developing of a system that can be applied to any object or environment with little or no modifications and easily used by any person. The system will provide full remote and secure control and monitoring of sensor networks, via an online platform, that can be applied to any non-smart object or environment allowing them to be connected to the Internet and to the user. With the possibility to add a set of rules and Artificial Intelligence it is possible to improve the efficiency leading to potential gains, such as energy or water savings. Although the main goal is to create a low cost system that is flexible in the sense that it can be easily adapted to any specification, in this paper the practical functionality will be tested and evaluated in a swimming pool, with an application capable of monitoring the water temperature and level, the environmental temperature, relative humidity, air pollution and luminosity and remotely control the water pumps and pool lights.

2. Related Work

IoT gateways⁵ are dedicated hardware applications used to connect the user to the network, allowing the conversion of data between the short distance communication protocols to the traditional communication network. The gateway is supposed to support different types of sensor nodes, multiple communication protocols, both wireless or wired, and provide a set of unified information for the application or user, making these only responsible for data processing.

The main challenge on creating an IoT gateway is the lack of standards, being that each sensor node can communicate with a different protocol that is not compatible for others. This makes the development of a general purpose gateway a complicated task, which explains why it is common to find gateways developed for specific applications. Nevertheless, all have the same key requirements: low-cost hardware, easy implementation and extensibility and an application layer support.

R. Gerstendorf in his article⁶ examines ten platforms for creating smart homes, from ZigBee to the Apple HomeKit. All ten are market solutions available to consumers as a ready-to-use product in order to control smart devices through a gateway. Each one has a communication protocol, proprietary characteristics and a range of pros and cons.

In the literature it is possible to find several proposals^{7,8,9} for IoT gateways implemented using low-cost hardware devices, such as Arduino and Raspberry Pi. Most of them use these devices to support the web server, which difficults its access from outside the network. Other solutions were found^{10,11}, that use wireless communication protocols for specific applications and little to none IoT gateways were designed to use wired protocols. Only one of these¹² did it, where the authors used the gateway with the RS485 protocol to control end-devices from the Internet. This makes all of these solutions limited in flexibility and adaptation to other environments.

The literature presented a similar concept for a multi-communication protocol IoT gateway. In this¹³, the authors present a heterogeneous IoT gateway capable of using the same board to communicate with multiple wireless protocols as well as support for a large amount of communication buses, in a modularity basis.

3. System Architecture

To maintain the normal requirements for IoT gateways and in order to be able to use multiple communication protocols, adapting them to user requirements, the authors of this paper created a gateway based on common requirements and capable of using every sensor and communication protocol, both wired and wireless, that is available. The system relies on a group of devices connected over a network that is controlled by an application using a given communication protocol. Fig. 2 presents a high level system architecture and, as shown, the system is composed by hardware, software and the implemented communication features. The proposed system was conceived using hardware capable of supporting multiple communication protocols and flexible enough so as to run adaptable software, as described in the following sections.

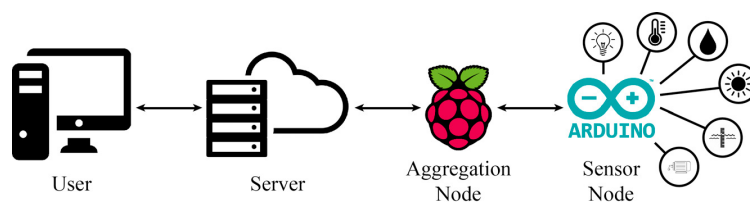


Fig. 2. System Architecture

3.1. Hardware

As displayed in Fig. 2 and 3, the hardware components adopted for our implementation consist in a Raspberry Pi 3¹⁴, a low-cost credit card sized computer with enough processing power and memory, that support programmable I/O ports and the use of standard peripherals, as an aggregation node or gateway; an Arduino Uno¹⁵, an open-source electronics single board based on a simple input/output capable of interfacing different peripherals, sensor and wireless communication devices, as a node sensor that collects information from temperature, humidity, luminosity and water level. Also a set of relays are used to control numerous actuators such as lights and motors.

The aggregation node are not responsible for reading sensors, they just provide a gateway between the user and the Sensor Network and also perform some data analyses. The sensor node, the lowest level of a Sensor Network, is responsible for gather information from sensors, perform user actions and use communication mechanisms to send data to the aggregation node.

3.2. Software

As presented in Figure 2 there are several pieces of software that are essential to the proposed architecture. They are divided into two sections that are mentioned below.

3.2.1. Web Platform

The web platform consists on a display for the information gathered from the sensor network. As can be seen in Fig. 4 in the platform the user can choose which sensors to display, as well as the current values and past data retrieved, turn on and off all the actuators and switch between functionality modes. The web platform also provides the user a bridge to the sensor network using the MQTT protocol, with the Eclipse Paho JavaScript Client¹⁶. It is hosted on a private server, containing a private MySQL database for storage and was built using HTML, PHP, Javascript, CSS and AJAX for both control and communication.

3.2.2. Python Scripts

Present in the aggregation node, this scripts are responsible for receiving the data from the sensor node and send them to the server using the MQTT protocol. They are also responsible for receiving commands sent from the user through the web platform and pass them to the sensor node. These scripts use the Eclipse Paho Python Client¹⁶. They

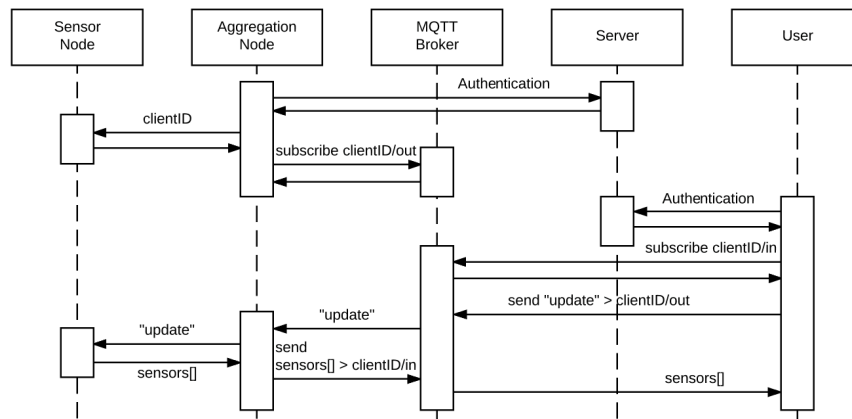


Fig. 5. Network and User authentication process and initial process

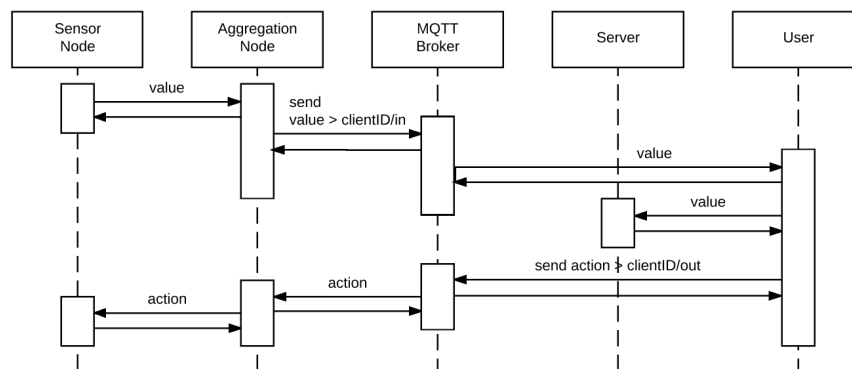


Fig. 6. Data and Actions exchange

protocol that can also connect a JavaScript Client to a Python Client, so the communications are done using MQTT, a protocol design for IoT projects. In the second, there is a range of options with both wired or wireless capabilities. In here we search for a low-cost, low-power, high range, multi-node communication protocol with low hardware complexity. The protocols that adapt to the specification range from both environments, like I2C, RS485, ZigBee or LoRaWAN but to ensure a bigger reliability and an easy connection the communications are done using Serial Communication in the form of an USB cable.

3.3.1. Universal Serial Bus

USB was developed in order to facilitate the connection of peripheral devices to a computer. The propose of USB was to give the ability to devices to be plug and play with a single interface and automatic configuration. Consists in a communication bus used for connecting and power devices, such as micro-controllers from other interfaces. USB operates at 5 volts using 4 different lines: Power, Ground and a twisted pair of data lines using NRZI encoding^{18,19}. Providing a fast Master/Slave interface capable of supporting up to 127 devices with up to 6 hubs, USB uses an enumeration process where each slave is assigned with a unique address and give the master information about which speed will be used and what type of data will be transferring.¹⁹.

3.3.2. Message Queuing Telemetry Transport

MQTT is a messaging protocol that aims at connecting embedded devices and networks with applications and middleware^{4,20,21}. Built on top of the TCP protocol, uses a publish/subscribe pattern, with a routing mechanism (one-to-one, one-to-many, many-to-many), to provide flexibility and simplicity transition making MQTT an optimal connection protocol for the IoT and M2M being suitable for small, cheap, low power and low memory devices with low bandwidth networks. Compared to HTTP, MQTT is designed to have a lower protocol overhead.

MQTT consist of three components: subscriber, publisher and broker. A device registers as a subscriber for specific topics of interest in order to get the information publish to that topic. The publisher acts as a generator of data for some topic, transmitting that information to the subscriber through the broker.

4. Results

The developed gateway can be seen in Fig. 7, 8, 9 and 10 with every hardware feature and communication protocol used.

In Fig. 7 we see the whole setup, with the online platform on the laptop, showing the data retrieved from the sensor network. It is possible to see that the data is shown at the respective place, that the user has the possibility to turn the lights on and off and change between modes (Normal Mode turned on, green on the platform and first light on the sensor node).

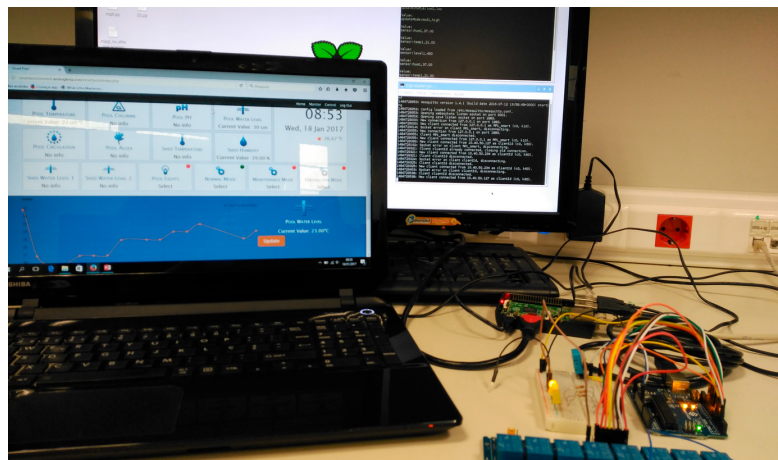
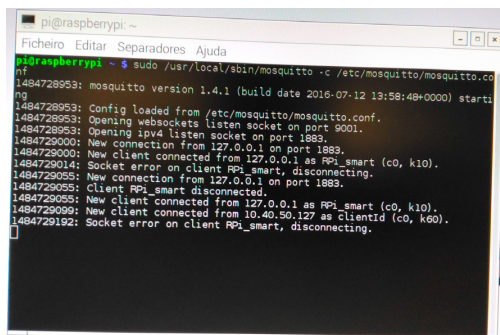


Fig. 7. Functional prototype setup

In Fig. 8 it is possible to see the MQTT broker log showing the client and the network connecting and disconnecting. Without these results the communication over MQTT is not possible, since both parts need to be connected to the broker in order to receive the messages from the topics they subscribed previously. Fig. 9 shows the script that controls the aggregation node with the response to the *update* message, data transmission and actions received. When the aggregation node receives the *update* message it responds with all the sensors and actuators present in the network and after the successful connection it receives the user chosen mode and light command. Also it is possible to see that the data retrieved from the sensor node over USB are successfully received in the aggregation node and then sent to the platform.

Fig. 10 demonstrates the hardware setup, with the nodes connected with USB, the sensors connected to the sensor node and the actuator working with user specifications. As said before the lights (red led) are off and the normal mode lights (first yellow led) are on, both selected by the user on the platform. The sensor values retrieved are accordingly with the expected ones, with the temperature at 22°C and humidity at 39% in a room at 25°C and low humidity. With these results we conclude that the sensor node is capable of obtaining the values, send them to the aggregation node and also perform the action that the user send using the online platform.

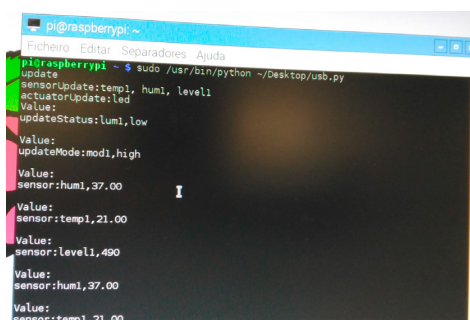


```

pi@raspberrypi:~$ sudo /usr/local/sbin/mosquitto -c /etc/mosquitto/mosquitto.conf
1484728953: mosquitto version 1.4.1 (build date 2016-07-12 13:58:49+0000) starting
1484728953: Config loaded from /etc/mosquitto/mosquitto.conf.
1484728953: Opening websockets listen socket on port 9001.
1484729000: Opening ipv4 listen socket on port 1883.
1484729000: New connection from 127.0.0.1 on port 1883.
1484729000: New client connected from 127.0.0.1 as RPi_smart (c0, k10).
1484729014: Socket error on client RPi_smart, disconnecting.
1484729055: New connection from 127.0.0.1 on port 1883.
1484729055: client RPi_smart disconnected.
1484729055: New client connected from 127.0.0.1 as RPi_smart (c0, k10).
1484729099: New client connected from 10.40.50.127 as clientid (c0, k60).
1484729192: Socket error on client RPi_smart, disconnecting.

```

Fig. 8. MQTT Broker script



```

pi@raspberrypi:~$ sudo /usr/bin/python ~/Desktop/us0.py
update
sensor:update:temp1,humi,level1
actuator:update:led
Value:
updateStatus:lumi,low
Value:
updateMode:modi,high
Value:
sensor:humi,37.00
Value:
sensor:temp1,21.00
Value:
sensor:level1,490
Value:
sensor:humi,37.00
Value:
sensor:temp1,21.00

```

Fig. 9. Aggregation node script

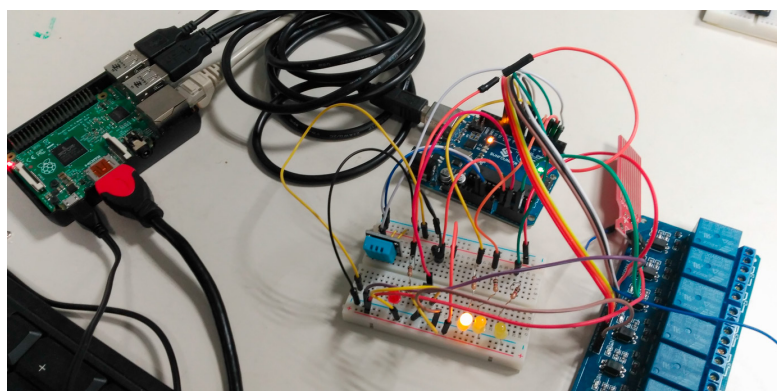


Fig. 10. Functional prototype hardware setup

5. Conclusions

In this paper, an IoT gateway for creating a smart swimming pool was designed, consisting in a Raspberry Pi acting as an aggregation node, an Arduino with a set of sensors as a sensor node and a web platform for monitor and control the network. The developed system so far, accomplishes the proposal features and purpose, giving the user the ability to control the environment remotely. Nevertheless more tests to the system in multiple situations can be done in order to prove the efficiency of the developed architecture.

In the future, the authors of this publication are planning to compare multiple communications protocols for each point of communication, or even offer a solution supporting multiple communication schemes. Furthermore, additional functionalities will be considered for the system, such as the possibility to create sets of rules directly from the platform, and add Artificial Intelligence or Machine Learning. At last the focus will be towards a more secure system, with the implementation of encrypted communications and a SSL protocol and certificate on web server side.

Acknowledgments

The authors would like to thank Professor Octavian Postolache for providing the sensors. This work was partially supported by the FCT - Fundação para a Ciência e Tecnologia and Instituto de Telecomunicações under project UID/EEA/50008/2013.

References

1. J. Gantz, D. Reinsel, THE DIGITAL UNIVERSE IN 2020: Big Data, Bigger Digital Shadows, and Biggest Growth in the Far East, IDC iView: IDC Anal. Future 2007 (2012) 1–16.
2. D. Evans, The Internet of Things How the Next Evolution of the Internet Is Changing Everything, CISCO, San Jose, CA, USA, White Paper.
3. S. Taylor, The Next Generation of the Internet Revolutionizing the Way We Work, Live, Play, and Learn, CISCO, San Francisco, CA, USA, CISCO Point of View.
4. A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, M. Ayyash, Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications, IEEE Communications Surveys and Tutorials doi:10.1109/COMST.2015.2444095.
5. C. L. Zhong, Z. Zhu, R. G. Huang, Study on the IOT architecture and gateway technology, in: Proceedings - 14th International Symposium on Distributed Computing and Applications for Business, Engineering and Science, DCABES 2015, 2016. doi:10.1109/DCABES.2015.56.
6. R. Gerstendorf, Ten platforms examined (9 2016).
7. A. Grygoruk, J. Legierski, IoT gateway – implementation proposal based on Arduino board, Federated Conference on Computer Science and Information Systems (FedCSIS) doi:10.15439/2016F283.
8. S. M. Kim, H. S. Choi, W. S. Rhee, IoT home gateway for auto-configuration and management of MQTT devices, in: 2015 IEEE Conference on Wireless Sensors, ICWiSE 2015, 2015. doi:10.1109/ICWiSE.2015.7380346.
9. R. Balakrishnan, IoT based Monitoring and Control System for Home Automation, Global Conference on Communication Technologies (GCCT).
10. T. Zachariah, N. Klugman, B. Campbell, J. Adkins, N. Jackson, P. Dutta, The Internet of Things Has a Gateway Problem- doi:10.1145/2699343.2699344.
URL <http://dx.doi.org/10.1145/2699343.2699344>.
11. S. Guoqiang, C. Yanming, Z. Chao, Z. Yanxu, Design and implementation of a smart IoT gateway, in: Proceedings - 2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing, GreenCom-iThings-CPSCOM 2013, 2013. doi:10.1109/GreenCom-iThings-CPSCOM.2013.130.
12. M. Dong, X. Zeng, S. Bi, S. Guo, Design and implementation of the multi-channel RS485 IOT gateway, in: Proceedings - 2012 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems, CYBER 2012, 2012. doi:10.1109/CYBER.2012.6392581.
13. E. Gioia, P. Passaro, M. Petracca, AMBER: an advanced gateway solution to support heterogeneous IoT technologies, 24th International Conference on Software, Telecommunications and Computer Networks (SoftCOM).
14. C. Bell, Beginning Sensor Networks with Arduino and Raspberry Pi, 1st Edition, Apress, 2013.
15. M. Banzi, Getting started with Arduino, 2nd Edition, Make:Books, 2011.
16. Eclipse Foundation Inc., Paho - Open Source messaging for M2M (2016).
URL <https://eclipse.org/paho/>
17. Eclipse Foundation Inc., Mosquitto - An Open Source MQTT broker (2016).
URL <https://mosquitto.org/>
18. M. Sharma, N. Agarwal, S. Reddy, Design and Development of Daughter Board for USB-UART Communication between Raspberry Pi and PC, International Conference on Computing, Communication & Automation.
19. Computer-solutions.co.uk, USB - a brief tutorial for embedded engineers (2015).
URL http://www.computer-solutions.co.uk/info/Embedded_tutorials/usb_tutorial.htm
20. D. Locke, MQ Telemetry Transport (MQTT) v3.1 protocol specification (2010).
URL <https://www.ibm.com/developerworks/webservices/library/ws-mqtt/index.html>
21. N. De Caro, W. Colitti, K. Steenhaut, G. Mangino, G. Reali, Comparison of two lightweight protocols for smartphone-based sensing, in: IEEE SCVT 2013 - Proceedings of 20th IEEE Symposium on Communications and Vehicular Technology in the BeNeLux, 2013. doi:10.1109/SCVT.2013.6735994.

Comparison of Communication Protocols for Low Cost Internet of Things Devices

André Glória^(1,2), Francisco Cercas^(1,2), Nuno Souto^(1,2)

⁽¹⁾ ISCTE-IUL, Av. das Forças Armadas, Lisbon, Portugal

⁽²⁾ Instituto de Telecomunicações, Av. Rovisco Pais, 1, Lisbon, Portugal

Abstract—Internet of Things (IoT) uses the connection between devices to improve their efficiency and user experience, being the communication one of the main elements for a proper IoT network. This paper presents a review of the most common wired and wireless communication protocols, discusses their characteristics, advantages and disadvantages as well as a comparison study to choose the best bidirectional sensor network composed by low power devices such as Arduino, ESP-12 and Raspberry Pi.

Index Terms—Internet of Things, Sensor Networks, I²C, RS232, ZigBee, LoRa, Arduino, ESP-12, Raspberry Pi

I. INTRODUCTION

In order to provide full control of physical devices over the web the urge to connect everything to the Internet is growing, with an expected 200 billion smart devices being connected to the Internet by 2020 [1]. With the capability of gathering, analyzing and distributing the data, Internet of Things consists in the connection between the Internet and a range of devices and sensors and was designed to play a great role improving our quality of live.

IoT relies on group of devices connected over a network controlled by an application using a communication protocol. These devices can be anything capable of retrieving and storing data, but the most usual are microcontrollers or System on Chip (SoC) devices. SoC [2] is a replacement for a computer, based on an electronic chip or integrated circuit containing the whole computing system. With this approach the cost of producing a computing system has drop, making these devices much cheaper for the end consumer. The most common examples of this platforms are the Arduino [3], an open-source electronics single board based on a simple input/output capable of interfacing different peripherals, sensor and wireless communication devices, the Raspberry Pi [4], a low-cost credit card sized computer with enough processing power and memory, that support programmable I/O ports and the use of standard peripherals, and most recently the ESP-12 [5], a WiFi module based on the ESP8266 core processor, providing the ability to interface with sensors and actuators through I/O pins and offering a complete and self-contained WiFi (IEEE802.11 b/g/n) networking solution as a low-power, low-cost and minimal space device.

For an IoT system capable of creating smart environments [6], the need to adapt to any specification requires that multiple options for devices and communication protocols be available. Fig. 1 shows the system architecture and it is possible to

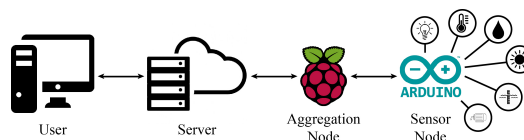


Fig. 1. System Architecture [6]

see that two types of devices are essential, an aggregation node that provide a gateway between the user and the Sensor Network and also perform some data analyses, and a sensor node responsible for gather information from sensors and perform user actions. The previously presented devices are capable of performing the task needed for this nodes, with the Raspberry Pi and the ESP-12 being more suitable for the aggregation node, as they have more computational power, memory and WiFi connectivity, whereas the Arduino is the best choice for the sensor node, as it has more I/O pin, with analog inputs available, and lower power consumption. Regardless of the platforms chosen, these two nodes require a bidirectional communication mechanisms in order to exchange the data retrieved from sensors to the aggregation node and also get the user actions from the aggregation node to the sensor node to be applied to the actuators. Other thing to have in mind when choosing the right communication protocol is the need to have a multi-node capability, due to the fact that multiple sensor nodes can be included in the system, all communicating with the same aggregation node. Also, and bearing in mind the low-cost and power saving capabilities of the system, is important that the protocol have these features as well.

In this paper the authors present a detailed review of the main communication protocols available, as well as a multiple scenario analysis to comprehend if distance and different board combination affects delay, data rate and efficiency for the tested protocols in order to choose the best communication protocol for each scenario.

II. COMMUNICATIONS PROTOCOLS

Communication is one of the main elements of IoT, as the need to trade data between devices is crucial to the efficiency of an IoT project. Besides the word Internet being part of IoT, not all the situations rely on Internet related protocols. With

the increased use of Sensor Networks and applications in the most diverse environments, the need of different protocols, both wired and wireless, is growing. Wired protocols are still used to connect devices, since they are more reliable, secure and can transfer data at higher rates. The most common wired technologies are Serial Communication or Universal Asynchronous Receiver and Transmitter (UART), mainly in the form of USB or RS232, RS485, SPI, I²C and PLC. Also CAN bus can be used for networks that need a bigger reliability. Wireless communication protocols, offer all what is needed to work with the advantage of a wireless low space environment. The most common wireless technologies are WiFi, ZigBee, Bluetooth, with new technologies like 6LoWPAN, LoRaWAN and even LTE-A being increasingly used.

Present in Table I are some key characteristics of the main communications protocols.

Without the possibility to test every protocol and bearing in mind that the goal is to use a low-cost, low-power, high range, multi-node communication protocol that also has a low complexity hardware configuration, the chosen protocols to do further tests are I²C, RS232, ZigBee and LoRa. A more precise description, as well as advantages and disadvantages, of each one can be found in the following subsections.

A. Inter-Integrated Circuit

The Inter-Integrated Circuit [7] was developed in 1982 with the purpose of reducing the number of wiring printed in the PCB and avoid additional logic, using just two wires for connecting the peripherals to the microcontroller, called Serial Data (SDA) and Serial Clock (SCL), making this protocol ideal for communication between integrated circuits and slow communication with on-board peripherals. With this two lines it is possible to connect multiple masters and slaves, making I²C a multi-master protocol. This two signal lines make communications between devices possible using a protocol that defines a unique 7-bit slave address, used to connect to the bus, 8 bytes containing data and a few bytes for communication control. Besides this two lines, physically, the bus also needs a ground connection. Both active lines are bi-directional making the device that starts the data transfer on the bus a master, being the other slaves.

I²C has a range of data rates [7] in which we can choose to transmit. *Standart mode*, at 100 kb/s, *fast mode*, at 400 kb/s,

and *high speed mode*, at 3.4 Mb/s, are the main data rates for I²C. But some variants of the protocol add *low speed mode*, at 10 kb/s, and *fast mode +*, at 1 Mb/s, as valid speeds.

Advantages of I²C [8]:

- Easy to connect;
- Widely supported;
- Automatically configured;
- Low power consumption;

Disadvantages of I²C [8]:

- Does not support long distance communication;
- Does not support High Speed connections;
- Number of nodes is limited by the address space on the bus;

B. Universal Asynchronous Receiver and Transmitter

The UART is a programmable integrated circuit capable of interfacing serial devices [9]. This interface provide operations such as converting the bytes to a single serial bit stream for outbound transmission, and vice versa for inbound transmission, adding a start and stop bits to signal the beginning and end of a data word, and a parity bit for error detection [9], [10]. To a correct stream of data there is no need for a clock signal since UART is asynchronous but both ends of the line must operate with the same baud rate [11].

There are four different standarts for UARTs: RS232, RS423, RS422 and RS485. The RS232 has a signe-ended line configuration with a Simplex or Full Duplex operation, being able to achieve up to 15 meters at 20 kbits/s.

Advantages of UART:

- Widely supported;
- No clock signal needed;
- Robust to errors;

Disadvantages of UART:

- Limited size of 9 bits;

C. IEEE 802.15.4 (ZigBee)

Introduced in 2002, ZigBee uses the IEEE 802.15.4 protocol as a base. Created for low-rate wireless private areas networks (LR-WPAN) it is one of the most used communication protocols for IoT due to is low consumption, low data rate, low cost and high message throughput. It can also provide high reliability, security, with both encryption and authentication

TABLE I
MAJOR COMMUNICATION PROTOCOLS CHARACTERISTICS

Feature	SPI	I ² C	RS232	RS485	WiFi	Bluetooth	ZigBee	LoRa
Based Data Rate [Mbps]	20	0.1	0.02	10	11	1	0.25	0.11
Frequency [GHz]	-	-	-	-	2,4	2,4	2,4	0.433
Version	-	Standard	-	-	802.11b	4.0	-	-
Range [m]	100	10	15	60	1-100	10-100	10-100	2000
Nodes/Masters	3	1024	256	256	32	7	65540	-
Power Consumption [mA]	-	-	-	-	100-350	1-35	1-10	1-10
Complexity	Medium	Low	Low	Low	High	Medium	Medium	Low
Security	-	-	-	-	WPA/WPA2	128 bit	128 bit	128 bit

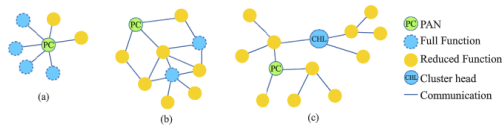


Fig. 2. IEEE 802.15.4 topologies. (a) Star, (b) Peer-to-Peer, (c) Cluster-tree

services, works with different platforms and can handle up to 65000 nodes. The IEEE 802.15.4 works in three frequencies, with a DSSS method, capable of transferring data at 250 kbps at 2.4 GHz, the most usual [8], [12]. To ensure that collisions are avoided, the IEEE 802.15.4 MAC sublayer uses the CSMA/CA protocol. The MAC sublayer is also responsible for flow control via acknowledged frame delivery, frame validations as well as maintaining network synchronization, controlling the association, administering device security and scheming the guaranteed time slot mechanism [13].

This standard can support two types of network nodes, Full Function Device (FFD), responsible for creating, controlling and maintain the network, and Reduce Function Device (RFD), simple nodes with low resources being just able to communicate with the coordinator in a star topology [12], [14]. Fig. 2 shows the topologies for the IEEE 802.15.4 protocol.

Advantages of ZigBee [8]:

- Low power consumption, allows devices to operate from batteries;
- One coordinator can control a numerous amount of slaves;
- Self-organizing network capabilities;
- Highly secure, with 128-bit AES encryption;

Disadvantages of ZigBee [8]:

- Setting up the network requires additional devices, which can increase costs;
- Low data transmission, only 127 bytes per message;
- Is incompatible with other network protocols and lacks Internet Protocol support;

1) *LoRa*: A bidirectional communication protocol that uses the LoRa physical layer in order to provide low power long-range communications. To achieve this, LoRa is based on Chirp Spread Spectrum (CSS) modulation that have the same low power characteristics as FSK modulation (present in great part of the other wireless communication protocols) but with a significant increase in the communication range. With a single base station it is possible to cover up to hundreds of square kilometers. To guarantee that all communications are completed, the LoRa MAC layer is responsible for join and accept the end-point and the gateway, schedule the receiving slots for end-points and confirm the reception of the received packets [15]. LoRa uses a star topology in order to maintain the low power long communication viable, reducing complexity and increasing network capacity and lifetime as opposed by a mesh topology. This is possible by using a higher link budget, lower

license-free frequency such as 433 or 900 MHz and software adaptive power output with transceiver modules [16].

Advantages of LoRa [16]:

- Low power consumption, allows devices to operate from batteries;
- Long communication range;
- Highly secure, with 128-bit AES encryption;

Disadvantages of LoRa:

- Low data transmission, only 55 bytes per message;

III. TESTS SCENARIOS

To ensure that the best communication protocols were used in the system, a set of tests was applied to the network nodes in order to choose the best one.

These tests focus on the exchange of strings of data between the aggregation node, the master, and the sensor node, the slave. The main goal was to see if the communication protocol can work with these platforms, how well they perform when increasing the distance between nodes and also to compare the results with theoretical ones.

For these, three different tests were made in order to see the following values:

- Throughput;
- Message Delay;
- Efficiency;

Also the complexity of setting the system to work and master slave combination were evaluated.

For the first test, the master sends a continuous amount of data packets, with no interval between them, and the slave just receives them, registers the timestamp and increases the number of packets received. Then we see how many packets the slave can process in a second, in order to obtain the true system throughput.

In the second one, a similar method is used, but this time only 100 packets are sent with an interval of one second. The slave receives the packets and registers the time between packets. Then the average time is calculated, in order to achieve the delay, over the one second, that the system needs to receive the packets.

In the last one, 1000 packets were sent to the slave and the slave only increase the number of received packets if the data is equal to the one sent. This way it is possible to see how many packets were lost or have errors.

All the tests have been done with a distance of 1, 2, 5, 10, 20 and 50 meters between nodes and with the following board combination, as both master and slave: Raspberry Pi - Arduino, Arduino - Arduino, ESP12 - Arduino, in order to evaluate every scenario possible.

A. Complexity

For I²C and RS232, Arduino, ESP12 and Raspberry Pi platforms have native pins to use this protocols, so no additional hardware is needed. ZigBee and LoRa need the external radio interfaces in order to communicate. For ZigBee a pair of XBee S2, an Arduino Wireless Proto Shield and XBee Explorer were

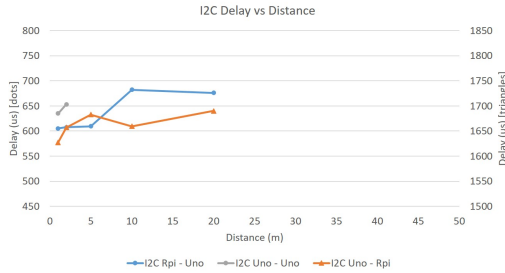


Fig. 3. I²C Delay vs Distance results

used to connect everything. For LoRa only a pair of Adafruit RFM69HCW is needed.

In terms of software, for I²C in the Raspberry Pi the smbus library was used and in the Arduino and ESP12 the Wire library. For a correct communication only the slave address needs to be set on both platforms as well as the callbacks on both ends. I²C does not allow the Raspberry Pi to be a Slave, so some modification to the code was needed in order to set the Arduino as a Master. Also a minimum of 1 ms was needed between packets in order to avoid overflow of packets when sending to the Arduino.

For RS232 the communication works as a simple Serial Communication with the RX and TX ports on both ends, with a simple modification on settings of the Raspberry Pi in order to tell the OS not to use that port for console communication. Also the pySerial library was needed.

For ZigBee, the most complex configuration, it was necessary to pair the radios using XCTU software. After that each radio was assigned with an address that the other end needed to know in order to send the packet. On the Raspberry Pi the XBee library was used and on the Arduino and ESP12 a 10ms between packets was needed to ensure a correct transmission of packets.

For LoRa, the RadioHead library was used on all platforms. To configure it only an address for each node had to be chosen, guaranteeing it was different for each node.

IV. RESULTS

The first thing to notice is that some protocols were not able to connect in some platforms, I²C with ESP-12 is one of the cases and also LoRa with the Raspberry Pi. The ESP-12 and ZigBee connection can be done, but due to ZigBee interference with WiFi, the results were not viable thus it was discarded as a possibility. One disadvantage detected was the fact that the Raspberry Pi can not be a I²C slave, having the Arduino to wait a second between packets and therefore affecting the results.

With these delay results, Fig. 3, 4, 5 and 6, it is possible to understand that the increase of distance does not affect much the delay value, with only a slight increase in all protocols. RS232 has the best results in all scenarios, with a delay

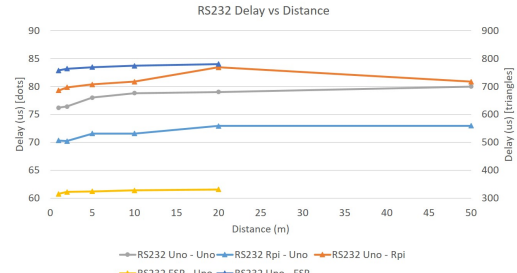


Fig. 4. RS232 Delay vs Distance results

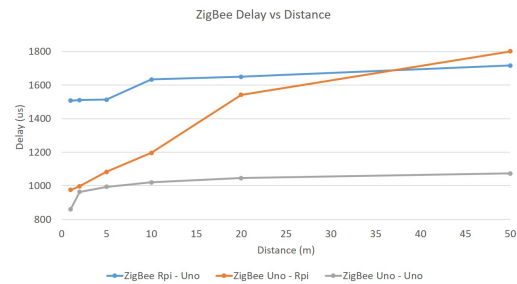


Fig. 5. ZigBee Delay vs Distance results

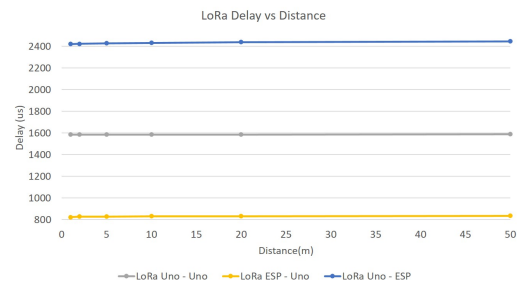


Fig. 6. LoRa Delay vs Distance results

ranging from around 75 to 800 us. LoRa has the higher delay with values ranging from 800 to 2500 us and also the biggest variation between scenarios. As of distance, the only one really affected is ZigBee with a bigger increase in delay over 10 meters in all scenarios. Regarding the master slave combination, is when both platforms are the Arduino Uno that the best results are achieved and, with only one Arduino Uno present, the delay decreases when the Uno is the slave.

Regarding throughput results, Fig. 7, 8, 9 and 10, it is possible to understand that distance affects the results, with a different pattern in each scenario, and that in every protocol the maximum value reached is widely different from the

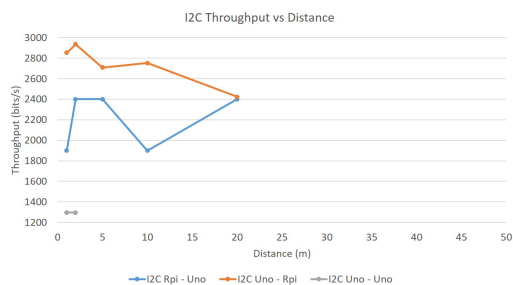


Fig. 7. I²C Throughput vs Distance results

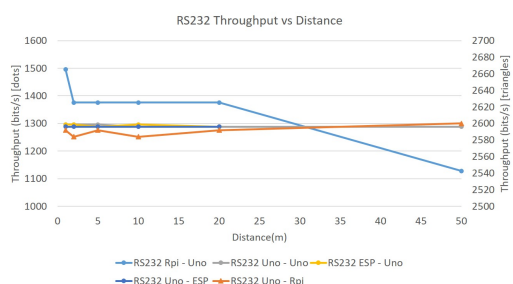


Fig. 8. RS232 Throughput vs Distance results

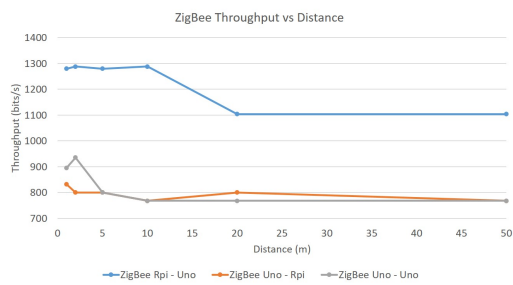


Fig. 9. ZigBee Throughput vs Distance results

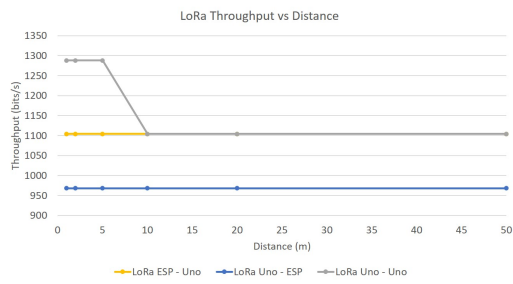


Fig. 10. LoRa Throughput vs Distance results

theoretical values. In some cases there is up and downs with the increase of distance but always with slight variations, so that was not taken in consideration. I²C has the most inconsistent results with big variations in both distance and master slave combination, with a data rate ranging from 1.3 to 2.9 kbits/s. On the other hand, RS232 has similar data rate results, ranging from 1.1 to 2.6 kbits/s, but with a more consistent and linear results. LoRa and ZigBee got very similar results, ranging from 0.75 to 1.3 kbits/s, with LoRa having the most linear results, almost none variation with the increased distance. Regarding the master slave combination, the best results are achieved when only one Arduino Uno is present, with some variations as it is the master or the slave.

As for the efficiency tests, I²C had the worst results as communications at 50 meters were not able to connect and when both platforms were Arduino Uno, nothing over 2 meters worked. This could be improved with additional hardware such as I²C range extenders. RS232, when connecting ESP-12 and Arduino Uno was not able to connect over 20 meters. Every other scenario tested got a 100% efficiency rate, including LoRa and ZigBee for every distance.

V. CONCLUSIONS

In this paper the authors focused on comparing communication protocols, to choose the best for a set of scenarios using low-cost devices, based on delay, data rate and efficiency.

Concluding the communication research, RS232 is the better choice when a wired protocol is needed and LoRa is a more reliable choice for a wireless protocol mainly because of the low complexity and cost needed and the fact that does not interferes with WiFi. Another focus was choosing the best devices for each node of the system, based on previous stated characteristics and the communication tests done. With all these results in mind, it is possible to say that the best suited combination includes an Arduino Uno as an Sensor Node and the ESP12 as the Aggregation Node. This last decision takes place not only because it is a cheaper solution, not putting in risk the reliability of the system, capable of sustaining the node specifications but also due to the Raspberry Pi 3 extra features, such as Internet browser, file systems and others, are not needed for the system, it has a bigger power consumption and the RS232 lines are also used in the system console, meaning that some interference can exist.

ACKNOWLEDGMENT

This work was partially supported by the FCT - Fundação para a Ciência e Tecnologia and Instituto de Telecomunicações under project UID/EEA/50008/2013.

REFERENCES

- [1] J. Gantz and D. Reinsel, "THE DIGITAL UNIVERSE IN 2020: Big Data, Bigger Digital Shadows, and Biggest Growth in the Far East," *IDC iView: IDC Anal. Future*, vol. 2007, pp. 1–16, 2012.

- [2] A. Ghosh, "Intelligent Appliances Controller Using Raspberry Pi Through Android Application and Browser," *IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, 2016.
- [3] M. Banzai, *Getting started with Arduino*, 2nd ed., O'Reilly, Ed. Make:Books, 2011.
- [4] C. Bell, *Beginning Sensor Networks with Arduino and Raspberry Pi*, 1st ed., Apress, Ed. Apress, 2013.
- [5] AI Thinker, "ESP-12E WiFi Module Datasheet," 2015. [Online]. Available: <https://mintbox.in/media/esp-12e.pdf>
- [6] A. Glória, F. Cercas, and N. Souto, "Design and implementation of an IoT gateway to create smart environments," *The 8th International Conference on Ambient Systems, Networks and Technologies (ANT 2017)*, 2017.
- [7] F. Leens, "An introduction to I2C and SPI protocols," *IEEE Instrumentation and Measurement Magazine*, vol. 12, no. 1, pp. 8–13, 2009.
- [8] A. Hafeez, N. H. Kandil, B. Al-Omar, T. Landolst, and A. R. Al-Ali, "Smart home area networks protocols within the smart grid context," *Journal of Communications*, vol. 9, no. 9, pp. 665–671, 2014.
- [9] U. Nanda and S. K. Pattnaik, "Universal Asynchronous Receiver and Transmitter (UART)," *3rd International Conference on Advanced Computing and Communication Systems (ICACCS)*, pp. 22–23, 2016.
- [10] L. Frenzel, "What's The Difference Between The RS-232 And RS-485 Serial Interfaces?" [Online]. Available: <http://ocwitic.epsem.upc.edu/assignatures/se/recursos/rs232-vs-rs485>
- [11] M. Sharma, N. Agarwal, and S. Reddy, "Design and Development of Daughter Board for USB-UART Communication between Raspberry Pi and PC," *International Conference on Computing, Communication & Automation*, 2015.
- [12] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Communications Surveys and Tutorials*, 2015.
- [13] N. S. Bhat, "Design and Implementation of IEEE 802.15.4 Mac Protocol on FPGA," *Innovative Conference on Embedded Systems, Mobile Communication and Computing (ICEMC2)*, pp. 1–5, 2011.
- [14] J.-S. Lee, Y.-W. Su, and C.-C. Shen, "A Comparative Study of Wireless Protocols: Bluetooth, UWB, ZigBee, and Wi-Fi," *IECON 2007 - 33rd Annual Conference of the IEEE Industrial Electronics Society*, 2007.
- [15] LoRa™ Alliance, "LoRaWAN™ Specification," 2015. [Online]. Available: <https://www.lora-alliance.org/portals/0/specs/LoRaWANSpecification1R0.pdf>
- [16] LoRa® Alliance Technical Marketing Workgroup, "LoRaWAN™ What is it? A technical overview of LoRa ® and LoRaWAN™," *LoRa® Alliance, San Ramon, CA, White Paper*, 2015.

Bibliography

- [1] J. Gantz and D. Reinsel, “THE DIGITAL UNIVERSE IN 2020: Big Data, Bigger Digital Shadows, and Biggest Growth in the Far East,” *IDC iView: IDC Anal. Future*, vol. 2007, pp. 1–16, 2012.
- [2] D. Evans, “The Internet of Things How the Next Evolution of the Internet Is Changing Everything,” *CISCO, San Jose, CA, USA, White Paper*, 2011.
- [3] S. Taylor, “The Next Generation of the Internet Revolutionizing the Way We Work, Live, Play, and Learn,” *CISCO, San Francisco, CA, USA, CISCO Point of View*, 2013.
- [4] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, “Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications,” *IEEE Communications Surveys and Tutorials*, 2015.
- [5] L. Atzori, A. Iera, G. Morabito, and A. Diee, “The Internet of Things : A survey,” *Computer Networksxxx*, 2010.
- [6] C. Bell, *Beginning Sensor Networks with Arduino and Raspberry Pi*, 1st ed., Apress, Ed. Apress, 2013.
- [7] F. Leens, “An introduction to I2C and SPI protocols,” *IEEE Instrumentation and Measurement Magazine*, vol. 12, no. 1, pp. 8–13, 2009.
- [8] A. Hafeez, N. H. Kandil, B. Al-Omar, T. Landolsi, and A. R. Al-Ali, “Smart home area networks protocols within the smart grid context,” *Journal of Communications*, vol. 9, no. 9, pp. 665–671, 2014.

- [9] U. Nanda and S. K. Pattnaik, “Universal Asynchronous Receiver and Transmitter (UART),” *3rd International Conference on Advanced Computing and Communication Systems (ICACCS)*, pp. 22–23, 2016.
- [10] L. Frenzel, “What’s The Difference Between The RS-232 And RS-485 Serial Interfaces?” [Online] Available: <http://ocwitic.epsem.upc.edu/assignatures/se/recurso/rs232-vs-rs485>, (visited 09/12/2016).
- [11] M. Sharma, N. Agarwal, and S. Reddy, “Design and Development of Daughter Board for USB-UART Communication between Raspberry Pi and PC,” *International Conference on Computing, Communication & Automation*, 2015.
- [12] S. Kim, E. Y. Kwon, M. Kim, J. H. Cheon, S. H. Ju, Y. H. Lim, and M. S. Choi, “A secure smart-metering protocol over power-line communication,” *IEEE Transactions on Power Delivery*, vol. 26, no. 4, pp. 2370–2379, 2011.
- [13] E. Driscoll Jr., “The history of X10,” [Online] Available: http://home.planet.nl/~lhendrix/x10_history.htm, (visited 20/06/2017).
- [14] E. Ferro and F. Potortì, “Bluetooth and Wi-Fi wireless protocols: A survey and a comparison,” *IEEE Wireless Communications*, vol. 12, no. 1, pp. 12–16, 2005.
- [15] J.-S. Lee, Y.-W. Su, and C.-C. Shen, “A Comparative Study of Wireless Protocols: Bluetooth, UWB, ZigBee, and Wi-Fi,” *IECON 2007 - 33rd Annual Conference of the IEEE Industrial Electronics Society*, 2007.
- [16] LoRa® Alliance Technical Marketing Workgroup, “LoRaWAN™ What is it? A technical overview of LoRa ® and LoRaWAN™,” *LoRa® Alliance, San Ramon, CA, White Paper*, 2015.
- [17] D. Locke, “MQ Telemetry Transport (MQTT) v3.1 protocol specification,” 2010, [Online] Available: <https://www.ibm.com/developerworks/webservices/library/ws-mqtt/index.html>, (visited 01/11/2016).
- [18] N. De Caro, W. Colitti, K. Steenhaut, G. Mangino, and G. Reali, “Comparison of two lightweight protocols for smartphone-based sensing,” in *IEEE*

- SCVT 2013 - Proceedings of 20th IEEE Symposium on Communications and Vehicular Technology in the BeNeLux*, 2013.
- [19] D. Thangavel, X. Ma, A. Valera, H.-X. Tan, C. Keng, and Y. Tan, "Performance Evaluation of MQTT and CoAP via a Common Middleware," in *IEEE ISSNIP 2014 - 2014 IEEE 9th International Conference on Intelligent Sensors, Sensor Networks and Information Processing, Conference Proceedings*, 2014.
- [20] A. Ghosh, "Intelligent Appliances Controller Using Raspberry Pi Through Android Application and Browser," *IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, 2016.
- [21] A. Nayyar and V. Puri, "A review of Arduino board's, Lilypad's and Arduino shields," *3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, 2016.
- [22] Arduino, "Arduino - Introduction," 2016, [Online] Available: <https://www.arduino.cc/en/Guide/Introduction>, (visited 28/11/2016).
- [23] A. Grygoruk and J. Legierski, "IoT gateway – implementation proposal based on Arduino board," *Federated Conference on Computer Science and Information Systems (FedCSIS)*, 2016.
- [24] M. Banzi, *Getting started with Arduino*, 2nd ed., O'Reilly, Ed. Make:Books, 2011.
- [25] ESP8266 Community Wiki, "Getting Started with ESP8266," 2017, [Online] Available: <http://www.esp8266.com/wiki/doku.php?id=getting-started-with-the-esp8266>, (visited 24/05/2017).
- [26] AI Thinker, "ESP-12E WiFi Module Datasheet," 2015, [Online] Available: <https://mintbox.in/media/esp-12e.pdf>, (visited 04/05/2017).
- [27] BeagleBoard.org, "BeagleBoard.org - about," 2016, [Online] Available: <http://beagleboard.org/about>, (visited 24/05/2017).

- [28] N. Balani, “What Is the Intel® Edison Module?” 2016, [Online] Available: <https://software.intel.com/en-us/articles/what-is-the-intel-edison-module>, (visited 24/05/2017).
- [29] Arduino, “Arduino Uno,” 2016, [Online] Available: <https://www.arduino.cc/en/Main/ArduinoBoardUno>, (visited 28/05/2017).
- [30] —, “Arduino Mega,” 2016, [Online] Available: <https://www.arduino.cc/en/Main/ArduinoBoardMega>, (visited 28/05/2017).
- [31] Raspberry Pi Foundation, “Raspberry Pi Zero W,” 2017, [Online] Available: <https://www.raspberrypi.org/products/pi-zero-w/>, (visited 28/05/2017).
- [32] —, “Raspberry Pi 3 Model B,” 2016, [Online] Available: <https://www.raspberrypi.org/products/raspberry-pi-3-model-b>, (visited 29/11/2016).
- [33] L. Ada and Adafruit ®, “Overview | Adafruit HUZZAH ESP8266 breakout,” 2016, [Online] Available: <https://learn.adafruit.com/adafruit-huzzah-esp8266-breakout/overview>, (visited 28/05/2017).
- [34] Intel Corporation., “Product Brief Intel® Edison,” 2015, [Online] Available: http://download.intel.com/support/edison/sb/edison_pb_331179002.pdf, (visited 28/05/2017).
- [35] C. L. Zhong, Z. Zhu, and R. G. Huang, “Study on the IOT architecture and gateway technology,” in *Proceedings - 14th International Symposium on Distributed Computing and Applications for Business, Engineering and Science, DCABES 2015*, 2016.
- [36] R. Gerstendorf, “Ten platforms examined,” 2016, in *Elektor Magazine*, pp. 10-20, 9 2016.
- [37] S. M. Kim, H. S. Choi, and W. S. Rhee, “IoT home gateway for auto-configuration and management of MQTT devices,” in *2015 IEEE Conference on Wireless Sensors, ICWiSE 2015*, 2015.

- [38] R. Balakrishnan, “IoT based Monitoring and Control System for Home Automation,” *Global Conference on Communication Technologies (GCCT)*, 2015.
- [39] T. Zachariah, N. Klugman, B. Campbell, and et al., “The Internet of Things Has a Gateway Problem,” in *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications (HotMobile '15)*, 2015.
- [40] S. Guoqiang, C. Yanming, Z. Chao, and Z. Yanxu, “Design and implementation of a smart IoT gateway,” in *Proceedings - 2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing, GreenCom-iThings-CPSCom 2013*, 2013.
- [41] M. Dong, X. Zeng, S. Bi, and S. Guo, “Design and implementation of the multi-channel RS485 IOT gateway,” in *Proceedings - 2012 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems, CYBER 2012*, 2012.
- [42] E. Gioia, P. Passaro, and M. Petracca, “AMBER: an advanced gateway solution to support heterogeneous IoT technologies,” *24th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, 2016.
- [43] IFTTT, “Learn how IFTTT works,” [Online] Available: <https://ifttt.com/>, (visited 10/12/2016).
- [44] Amazon Web Services, “AWS IoT Platform,” [Online] Available: <https://aws.amazon.com/pt/iot-platform/>, (visited 27/06/2017).
- [45] myDevices, “Cayenne IoT Project Builder,” [Online] Available: <https://mydevices.com/>, (visited 10/12/2016).
- [46] Eclipse Foundation Inc., “Paho - Open Source messaging for M2M,” 2016, [Online] Available: <https://eclipse.org/paho/>, (visited 22/07/2016).

- [47] OpenWeatherMap Inc., “Weather API - OpenWeatherMap,” 2016, [Online] Available: <http://openweathermap.org/api>, (visited 27/11/2016).
- [48] Eclipse Foundation Inc., “Mosquitto - An Open Source MQTT broker,” 2016, [Online] Available: <https://mosquitto.org/>, (visited 15/07/2016).
- [49] D. George, “MicroPython,” 2017, [Online] Available: <https://micropython.org/>, (visited 01/05/2017).
- [50] MicroPython, “umqtt library,” 2017, [Online] Available: <https://github.com/micropython/micropython-lib/tree/master/umqtt.simple>, (visited 01/05/2017).
- [51] Sunrom Technologies, “LDR Data Sheet,” 2008, [Online] Available: http://igem.org/wiki/images/1/1a/File-T--Technion_Israel-Hardwarespecsldr.pdf, (visited 26/12/2016).
- [52] A. Glória, A. Marques, and D. Fernandes, “ISCTE Satellite Station,” 2016, [Online] Available: <https://goo.gl/EmREYc>, (visited 06/06/2017).