Instituto Superior de Engenharia
Politécnico de Coimbra

DEPARTAMENTO DE ENGENHARIA ELETROTÉCNICA

Development of an IoT-based Smart Water Metering System

Trabalho de Projeto para a obtenção do grau de Mestre em Engenharia Eletrotécnica

Especialização em Automação e Comunicações em Sistemas Industriais

Autor

Eugenio Cimino

Orientadores

Professor Doutor Fernando José Pimentel Lopes

Professor Coordenador

Professor Doutor Victor Daniel Neto dos Santos

Professor Coordenador

INSTITUTO POLITÉCNICO DE COIMBRA

INSTITUTO SUPERIOR DE ENGENHARIA DE COIMBRA

Coimbra, Dezembro de 2024

ACKNOWLEDGEMENTS

My sincere thanks to my advisors, Professor Fernando José Pimentel Lopes and Professor Victor Daniel Neto dos Santos, for always guiding me wisely during my academic years and for their essential suggestions and constructive feedback that enhanced the quality of my work.

I wish to express my gratitude to my family for always supporting me in all my choices.

I am immensely grateful to my girlfriend Giada, for always being by my side and sharing all the successes of my life.

E por fim, obrigado Coimbra.

RESUMO

Um dos problemas mais relevantes que afectam a eficiência da distribuição de água são as perdas de água nos sistemas de abastecimento, que podem atingir cerca de 50% da água distribuída. Estas perdas podem ser causadas pela deterioração das infra-estruturas, por sistemas de faturação ineficazes, por medidas imprecisas ou por consumo não autorizado. As empresas de gestão da água preferem normalmente utilizar contadores de água totalmente mecânicos, uma vez que são de baixo custo e têm boa fiabilidade. No entanto, não proporcionam uma monitorização em tempo real nem automática do consumo de água, o que, certamente, não ajuda na luta contra as perdas. Conhecer em tempo real e em múltiplos pontos o valor da água consumida é útil para identificar perdas de água na conduta.

O objetivo deste projeto é desenvolver um dispositivo que não substitua o contador de água convencional já existente, mas que o transforme num AMR medidor. O dispositivo deve ser integrado numa aplicação IoT de monitorização.

O sistema desenvolvido utiliza a câmara ESP-EYE para adquirir imagens periódicas do contador de água convencional. As imagens são enviadas via Bluetooth para um Raspberry Pi, que é o núcleo de computação do sistema desenvolvido. A imagem é submetida a um pré-processamento, com o objetivo de selecionar apenas a parte que contém o valor lido pelo contador e de a tornar o mais nítida possível para o reconhecimento dos carateres . A imagem processada é o input do OCR Tesseract, que extrai dela o valor indicado pelo contador de água.

Este valor é armazenado numa base de dados local e enviado para o servidor Akenza, de forma que os dados também estejam disponíveis remotamente. O sistema desenvolvido deve ser instalado perto do contador de água. Se esse local estiver ao alcance de uma rede Wi-Fi, os dados serão enviados para o servidor Akenza através de MQTT sobre TCP/IP. Se, pelo contrário, o contador estiver longe de uma rede Wi-Fi, os dados serão enviados para o servidor TTN via LoRaWAN. Depois, por integração, chegarão ao servidor Akenza.

Palavras-Chave: OCR; IoT; AMR medidor; Contadores de água.

ABSTRACT

One of the most relevant problems that affect the water distribution efficiency are the water losses in supply systems that can reach up to about 50% of the distributed water. They can be caused by infrastructure deterioration, ineffective billing systems, inaccurate metering, or unauthorised consumption. Water management companies normally prefer to use fully mechanical water meters, since they are low-cost and have good reliability. However, they do not provide real-time nor automatic water consumption monitoring and this, certainly, does not help in the fight against losses. Knowing in real time and at multiple points the value of consumed water is useful for identifying water leaks on the pipeline.

The objective of this project is to develop a device that does not substitute the already present conventional water metering, instead it turns the conventional one into an AMR meter. The device should be integrated in a IoT monitoring application.

The developed system uses the ESP-EYE camera to periodically acquire images of the conventional water metering. The images are sent via Bluetooth to a Raspberry Pi, which is the computing core of the developed system. The image undergoes a pre-processing step, aimed at cropping out only the part containing the value read by the meter and at making it as clear as possible for character recognition. The processed image is the input of the OCR Tesseract, which extracts from it the value showed by the water meter.

This value is both stored in a local database and sent to the Akenza server, so that the data is also remotely available. The developed system must be installed near the water meter. If this location is in the range of a Wi-Fi network, then the data will be sent to Akenza using MQTT on TCP/IP. If, instead, the meter is far from a Wi-Fi network, then the data will be sent to TTN via LoRaWAN. Then, through integration, it will reach Akenza.

Keywords: OCR; IoT; AMR meter; Water meters.

TABLE OF CONTENTS

Acknowledgements i
Resumo iii
Abstract v
Table of contents vii
Table of figures x
Tables xvi
List of acronyms xvii
1 Introduction 1
1.1 Scope and motivation 1
1.2 Objectives 2
1.3 Document structure 3
2 State of the Art 5
2.1 Conventional Water Metering systems 5
2.1.1 Positive Displacement Flow Meters 6
2.1.2 Turbine Flow Meters 7
2.1.3 Electromagnetic Flow Meters 9
2.1.4 Ultrasonic Flow Meters 10
2.2 AMR Meter and AMR Smart Meter 12
2.2.1 Automatic Meter Reading (AMR) 12
2.2.2 AMR Smart meter 14
2.3 Meter-Bus (M-bus) 15
2.3.1 Principles of Operation 16
2.4 Overview on Artificial Intelligence (AI) 18
2.4.1 Artificial Intelligence 18
2.4.2 Machine Learning (ML) 19
2.4.3 Neural Network (NN) 20
2.4.4 Feedforward Neural Network (FNN) 23
2.4.5 Recurrent Neural Network (RNN) 26
2.4.6 Long Short-Term Memory (LSTM) 29
2.5 Optical Character Recognition (OCR) 31
2.5.1 History of OCR 31

2.6 Tesseract 32
2.6.1 Legacy Tesseract OCR engine 33
2.6.2 Neural net (LSTM) based OCR engine 33
2.7 Open-Source Computer Vision Library (OpenCV) 35
2.8 Internet of things (IoT) 35
3 Architecture of the Implemented System 37
3.1 ESPRESSIF ESP-EYE 39
3.1.1 Arduino IDE 40
3.2 Edge computing 40
3.2.1 Raspberry Pi 4 Model B 41
3.2.2 LILYGO® TTGO LoRa32 43
3.3 Bluetooth 44
3.4 Universal Asynchronous Receiver-Transmitter (UART) over Universal Serial Bus (USB) 46
3.4.1 USB Cable and Connectors 46
3.4.2 Universal Asynchronous Receiver-Transmitter (UART) 47
3.5 ISO OSI Model and TCP/IP Model 49
3.6 Wi-Fi 50
3.7 MQTT 51
3.8 LoRa 52
3.8.1 LoRa Message Format 55
3.9 LoRaWAN 56
3.9.1 LoRaWAN Message Format 57
3.9.2 LoRaWAN Architecture 58
3.10 The Things Network 61
3.11 Akenza 62
4 Implemented System 65
4.1 Initial configuration of the Raspberry 65
4.2 Getting started with TTN version 3 67
4.3 Cayenne LPP Formatter Type 75
4.3.1 The Cayenne LPP in TTN 76
4.4 Getting started with Akenza 77
4.4.1 Akenza Dashboard 83
4.5 Integration between TTN and Akenza 85

4.6 How to Install Tesseract and OpenCV on a Raspberry Pi 93
4.7 Configuring the Tesseract OCR for improved results 94
4.8 Tesseract training 95
4.8.1 Image collection 95
4.8.2 Pre-processing of the images 96
4.8.3 Creation of the boxes 99
4.8.4 Training 100
4.8.5 Further pre-processing of the images for the last digit 103
4.9 Post-processing 104
4.10 Description of the Software in the ESP-EYE 106
4.10.1 The *initBT()* function 107
4.10.2 The *btCallback()* function 108
4.10.3 The *writeSerialBT()* function 108
4.10.4 The *initCamera()* function 108
4.10.5 The *setCameraParam()* function 109
4.10.6 The *capture()* function 109
4.11 Description of the Software in the Raspberry 110
4.12 Description of the Software in the LoRa32 113
5 Experimental and Field Test Results 117
5.1 Field Test Results 117
5.2 Reading using the English language 121
5.3 Reading of a partial digit 122
5.4 Incorrect reading of the first digits, resolved by the algorithm 123
5.5 Inserting a white frame around the RoI of the last digit 124
5.6 Database 125
5.7 Analysis of the collected data 126
5.8 Water leakage detection 127
6 Conclusions and future work 131
Bibliography 133

TABLE OF FIGURES

Figure 1 – Essential Architecture. 2
Figure 2 – Essential Architecture for multiple water meters. 3
Figure 3 – Positive Displacement flow meter [3]. 6
Figure 4 – Turbine flow meter [4]. 7
Figure 5 – Single and Multi-Jet flow meters [5]. 8
Figure 6 – Electromagnetic flow meter [6] 9
Figure 7 – Ultrasonic flow meter [7]. 10
Figure 8 – Ultrasonic flow meter, V mounting method [7]. 11
Figure 9 – Conventional Meters Vs. Smart Meters [13]. 15
Figure 10 – The M-Bus layers in the OSI-Model [15]. 16
Figure 11 – Block diagram showing principle of the M-Bus System [13]. 16
Figure 12 – M-Bus communication [16]. 17
Figure 13 – Artificial Intelligence hierarchy. 18
Figure 14 – Three phases of AI in the history [17]. 19
Figure 15 – Machine Learning algorithms. 20
Figure 16 – Structure of a Real Neuron [21]. 21
Figure 17 – Artificial Neuron. Adapted from [20]. 21
Figure 18 – Step, sigmoid and hyperbolic functions. Adapted from [20]. 22
Figure 19 – Classification by a neuron with a step logic transfer function. Adapted from [22]. 22
Figure 20 – Example of FNN. Adapted from [23]. 23
Figure 21 – Input space division by two parallel neurons. Adapted from [24]. 24
Figure 22 – Input space division by two hidden layers made up of two neurons each. Adapted from [24]. 24
Figure 23 – Input space division by two hidden layers made up of two neurons each and a single neuron output layer. Adapted from [24]. 25
Figure 24 – FNN with 784 inputs, 15 neurons in hidden layer and 10 neurons in output layer [25]. 26
Figure 25 – RNN Example. Adapted from [26] 27
Figure 26 – RNN evolution in time. Adapted from [26] 28

Figure 27 – Vanishing gradient problem [26]. 28
Figure 28 – LSTM Architecture [28]. 29
Figure 29 – Detailed LSTM Architecture [29]. 30
Figure 30 – Legacy Tesseract OCR engine [33] 33
Figure 31 – How Tesseract uses LSTMs [35]. 34
Figure 32 – SoftMax function [36]. 34
Figure 33 – Architecture of the implemented system. 37
Figure 34 – Three-layered IoT Architecture. 37
Figure 35 – Key components, interfaces and functions of the ESP-EYE [42]. 39
Figure 36 – Arduino IDE 2.0. Adapted from [43]. 40
Figure 37 – Raspberry Pi 4 Model B [44]. 41
Figure 38 – Raspberry Pins. Adapted from [46]. 42
Figure 39 – Two versions of TTGO LILYGO. Adapted from [48] 43
Figure 40 – Rear and frontal view of the LoRa32 T3 model. 43
Figure 41 – *ls /dev/tty** while the LoRa32 was not connected to the Raspberry. 44
Figure 42 – *ls /dev/tty** while the LoRa32 was connected to the Raspberry. 44
Figure 43 – Bluetooth protocol stack [51] 45
Figure 44 – USB connectors [53]. 46
Figure 45 – USB 2.0 cable [54]. 47
Figure 46 – UART frame format. Adapted from [56]. 48
Figure 47 – OSI and TCP/IP Models. Adapted from [58]. 49
Figure 48 – Wi-Fi Evolution Timeline. Adapted from [60]. 51
Figure 49 – MQTT on TCP/IP [62]. 51
Figure 50 – MQTT communication [63] 52
Figure 51 – Linear up-chirp and linear down-chirp [67]. 53
Figure 52 – Distance range versus data rate of various wireless technologies [68]. 53
Figure 53 – Example of LoRa signal with SF=2. 54
Figure 54 – LoRa Physical layer Message (Frequency versus Time). Adapted from [73]. 56
Figure 55 – LoRaWAN model [74]. 56
Figure 56 – LoRaWAN Star-of-Stars Topology. 57
Figure 57 – LoRaWAN Frame [77] 57

Figure 58 – LoRaWAN Architecture. Adapted from [78]. 58
Figure 59 – LoRaWAN Gateways in Coimbra. 59
Figure 60 – Over-the-Air Activation (OTAA). Adapted from [80]. 60
Figure 61 – The Things Stack. Adapted from [81]. 61
Figure 62 – Choosing the Raspberry Pi OS on the Raspberry Pi Imager. 65
Figure 63 – a) PuTTY Configuration. b) Raspberry Command Prompt seen from PuTTY. 66
Figure 64 – The VNC Viewer Software. 67
Figure 65 – a) Arduino IDE, File → Preferences. b) Arduino IDE, Boards Manager. 68
Figure 66 – Arduino IDE, Select board → Select Other Board and Port. 68
Figure 67 – Arduino IDE, Library Manager. 69
Figure 68 – File *lmic_project_config.h*. 69
Figure 69 – TTN, Applications 70
Figure 70 – TTN, Create application. 70
Figure 71 – TTN, Application overview. 71
Figure 72 – TTN, Register end device (part 1). 72
Figure 73 – TTN, Register end device (part 2). 72
Figure 74 – TTN, Register end device (part 3). 73
Figure 75 – TTN, LoRa32 Device overview 73
Figure 76 – Arduino IDE, inserting AppEUI, DevEUI and AppKey into the code. 74
Figure 77 – Arduino IDE, writing the correct pins of the specific TTGO LoRa32. 74
Figure 78 – TTN, setting the Cayenne LPP Formatter type 77
Figure 79 – Akenza, welcome page. 78
Figure 80 – a) Akenza, creating an Organization. b) Akenza, creating a Workspace. 78
Figure 81 – a) Akenza, getting started. b) Akenza, Data Flows. 79
Figure 82 – Akenza, creating a Data Flow. 79
Figure 83 – Akenza, creating the MQTT Data Flow. 80
Figure 84 – Akenza, saving the MQTT Data Flow 80
Figure 85 – Akenza, creating the Raspberry Device (part 1). 81

Figure 86 – Akenza, creating the Raspberry Device (part 2). 81
Figure 87 – Akenza, generating the Raspberry Device ID. 82
Figure 88 – Akenza, the Raspberry Device created. 82
Figure 89 – Inserting the MQTT details into the Raspberry code. 83
Figure 90 – Akenza, Dashboard homepage. 83
Figure 91 – Akenza, creating the Dashboard (General information). 84
Figure 92 – Akenza, creating the Dashboard (Scope selection). 84
Figure 93 – Akenza, creating the Dashboard (Settings). 85
Figure 94 – TTN, Application ID. 85
Figure 95 – TTN, creating an API key (part 1). 86
Figure 96 – TTN, creating an API key (part 2). 86
Figure 97 – Akenza, creating the Integration with TTN. 87
Figure 98 – Akenza, inserting the details from TTN. 87
Figure 99 – Akenza, naming the Integration with TTN. 88
Figure 100 – Akenza, the created Integration. 88
Figure 101 – TTN, the resulting Integration. 89
Figure 102 – Akenza, Assets. 89
Figure 103 – Akenza, list of Data Flows. 90
Figure 104 – Akenza, creating the LoRaWAN Data Flow. 90
Figure 105 – Akenza, choosing the Device Connector. 91
Figure 106 – Akenza, choosing the Device Type. 91
Figure 107 – Akenza, choosing the Output Connector. 92
Figure 108 – Akenza, updated list of Devices in Assets. 92
Figure 109 – Akenza, assigning the LoRaWAN Data Flow to the LoRa32 device. 93
Figure 110 – OCR Engine modes. 94
Figure 111 – Page segmentation modes. 94
Figure 112 – Prototype placed on the water meter. 95
Figure 113 – Camera fixed inside the prototype. 96
Figure 114 – Digit 1 of an example image used for the training. 96
Figure 115 – Digit 1 in grayscale. 97
Figure 116 – 5X5 kernel used for blurring. 97
Figure 117 – digit 1 after blur. 97

Figure 118 – Digit 1 after the inverted binarization. 98
Figure 119 – Digit 1 after closing. 98
Figure 120 – Digit 1 after the resize. 99
Figure 121 – jTessBoxEditor. 100
Figure 122 – Folders necessary to do the Training. 101
Figure 123 – Rotation of the last digit from the value 0 to the value 1. 103
Figure 124 – Contour of the highest shape in the image. 104
Figure 125 – Last digit after cutting. 104
Figure 126 – Algorithm used for correctly read from digit 1 to digit 6. 105
Figure 127 – ESP-EYE Software: *setup()* function. 106
Figure 128 – ESP-EYE Software: *loop()* function. 107
Figure 129 – ESP-EYE Software: *initBT()* function. 107
Figure 130 – Flowchart of the Raspberry Software. 111
Figure 131 – LoRa32 Software: *setup()* function. 113
Figure 132 – LoRa32 Software: *loop()* function. 114
Figure 133 – LoRa32 Software: *do_send()* function. 115
Figure 134 – Example of an image acquired by the device. 117
Figure 135 – Water Measurements MQTT. 118
Figure 136 – Water Measurements LoRaWAN. 119
Figure 137 – Joining event on the LoRa32. 120
Figure 138 – Water consumption for the last hour. 120
Figure 139 – First value sent to Akenza. 120
Figure 140 – Values read one hour apart. 121
Figure 141 – Setting the default English language. 121
Figure 142 – Reading carried out using the English language. 121
Figure 143 – Rotating digit after pre-processing. 122
Figure 144 – Highlighted edges of the digit to be analysed. 122
Figure 145 – Correct reading of the rotating digit. 123
Figure 146 – Correct reading of the value 0590449. 123
Figure 147 – Algorithm intervention to correct reading error. 123
Figure 148 – Correct reading of the value 0590451. 124
Figure 149 – Seventh digit without white border. 124

Figure 150 – Seventh digit with white border. 125
Figure 151 – Constant water leak of the shower faucet. 127
Figure 152 – Water Measurements showed on Akenza. 128
Figure 153 – Consumption due to the loss of water. 128
Figure 154 – TARIFÁRIO para 2025, Águas de Coimbra [91]. 129

TABLES

Table 1 – Comparison of conventional water meters [2]. 5
Table 2 – Communication methods for meter reading [9]..... 13
Table 3 – Comparison of Raspberry Pi Models [46]..... 42
Table 4 – LoRa sub-gigahertz unlicensed bands [65]. 54
Table 5 – Parameters of LoRa. 55
Table 6 – LoRa Physical Layer message. 55
Table 7 – Data Flow parameters [85]..... 62
Table 8 – Structure of the Cayenne Low Power Payload..... 75
Table 9 – Data Types of the Cayenne LPP. 76
Table 10 – Example of the Cayenne LPP format. 77
Table 11 – Frame size for the acquired images. 108
Table 12 – Correspondence between the parameter and the size frame..... 109
Table 13 – Local Database. 125
Table 14 – Read values, actual values and absolute errors (m^3)..... 126
Table 15 – Relative error. 127

LIST OF ACRONYMS

ABP	Activation By Personalization
AEC	Automatic Exposure Control
AGC	Automatic Gain Control
AI	Artificial Intelligence
AI HLEG	High Level Expert Group on Artificial Intelligence
AMR	Automatic Metering Reading
CRC	Cyclic Redundancy Check
DCC	Data Communication Company
DEE	Departamento de Engenharia Eletrotécnica
EC	European Commission
FHSS	Frequency Hopping Spread Spectrum
FIFO	First In, First Out
FNN	Feedforward Neural Network
GFSK	Gaussian Frequency Shift-Keying
GPIO	General Purpose Input/Output
GPS	Global Positioning System
GSM	Global System for Mobile Communications
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of Things
IP	Internet Protocol
IPC	Instituto Politécnico de Coimbra
ISEC	Instituto Superior de Engenharia de Coimbra
ISO	International Standards Organization
L2CAP	Logical Link Control & Adaptation Protocol
LLC	Logical Link Control
LoRa	Long Range
LoRaWAN	Long Range Wide Area Network
LPP	Low Power Payload
LPWAN	Low Power Wide Area Networks

LSB	Least Significant Bit
LSTM	Long Short-Term Memory
MAC	Media Access Control
M-bus	Meter-Bus
MHDR	MAC header
MIC	Message Integrity Code
ML	Machine Learning
MQTT	Message Queuing Telemetry Transport
MSB	Most Significant Bit
OCR	Optical Character Recognition
OpenCV	Open-Source Computer Vision Library
OSI	Open Systems Interconnection
OTAA	Over-the-Air Activation
PD	Positive Displacement
PLC	Power Line Carrier
PSM	Page Segmentation Modes
PSRAM	Pseudo Static RAM
PWM	Pulse-Width Modulation
RF	Radio Frequency
RoI	Region of Interest
SBC	Single Board Computer
SiP	System-in-Package
SMETS	Smart Metering Equipment Technical Standard
SMS	Short Message Service
SoC	System on Chip
SPI	Serial Peripheral Interface
SWM	Smart Water Meters
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TTN	The Things Network
UDP	User Datagram Protocol
USB	Universal Serial Bus

1 INTRODUCTION

The efficient management of utilities such as natural gas, electricity or water is an extremely important topic for the modern societies, especially for the smart cities characterised by rapid urbanisation and a population growth [1]. The water usage is increasing more and more and the demand in the industrial and in the domestic fields are identified as the main causes for the increase in the water demand. The United Nations observed that, since the 1980s, the water usage worldwide is increasing by about 1% per year and that, at this rate by 2050, it will increase up to 30% above the current level. One of the most relevant problems that affects the water distribution efficiency are the water losses in supply systems, which can be caused by infrastructure deterioration, ineffective billing systems, inaccurate metering or unauthorised consumption. These water losses can reach up to about 50%, implying that the water distribution efficiency has to be improved. This can be done through the water distribution monitoring and the automatic metering.

1.1 Scope and motivation

Water management companies normally prefer to use fully mechanical water meters, since they are low cost and have good reliability, but they do not provide real-time nor automatic water consumption monitoring. The automatic metering reading technologies, instead, foresee the use of these smart water meters, which are part of a sensor network and measure the water consumption, store this data and communicate with a central hub. For the end users, this can be an effective way to raise consumer awareness, potentially leading to a considerable reduction in consumption. The operational costs can be reduced by the water management companies with human intervention, a better control of the distribution network, by limiting the water losses and by improving the overall efficiency. In recent years, there is a growing interest in sustainability, which is helping against the wastes of water, and also a growing interest in investigating the leakages located after the metering point, which can reach up to 13% of household water consumption.

Less effort has been dedicated to smart metering solutions for large-scale deployments. The Smart Water Meters (SWM) can be retrofitted or fully integrated. A retrofitted SWM is made by a new electronic device or module attached to a pre-existing mechanical water meter and it can perform automated water flow measurement, record water usage data and communicate it via wireless transmission. A fully integrated water meter is, instead, already equipped with these Automatic Meter Reading (AMR) capabilities. Nowadays, thanks to growing technologies and protocols dedicated to the Internet of Things (IoT), such as Low-Power Wide-Area Networks (LPWAN) with long-range and effective penetration, there is a growing

interest in the AMR with LPWAN technologies, which are able to transmit detailed meter readings.

1.2 Objectives

The objective of this project is to develop, implement and test a device, which does not substitute the already present conventional water meter, instead it turns the conventional one into an AMR meter. To proceed this way, it is necessary to adopt a camera, which periodically acquires images of the conventional water meter. The images are sent via Bluetooth to a computing device, where it will subsequently undergo a pre-processing step, aimed at cropping out only the part containing the value read by the meter and at making it as clear as possible. This new image will be the input of the Optical Character Recognition (OCR), which will extract the numerical value from it. Subsequently, this value will be both stored in a local database and sent to an external server, so that the data is also remotely available (Figure 1).

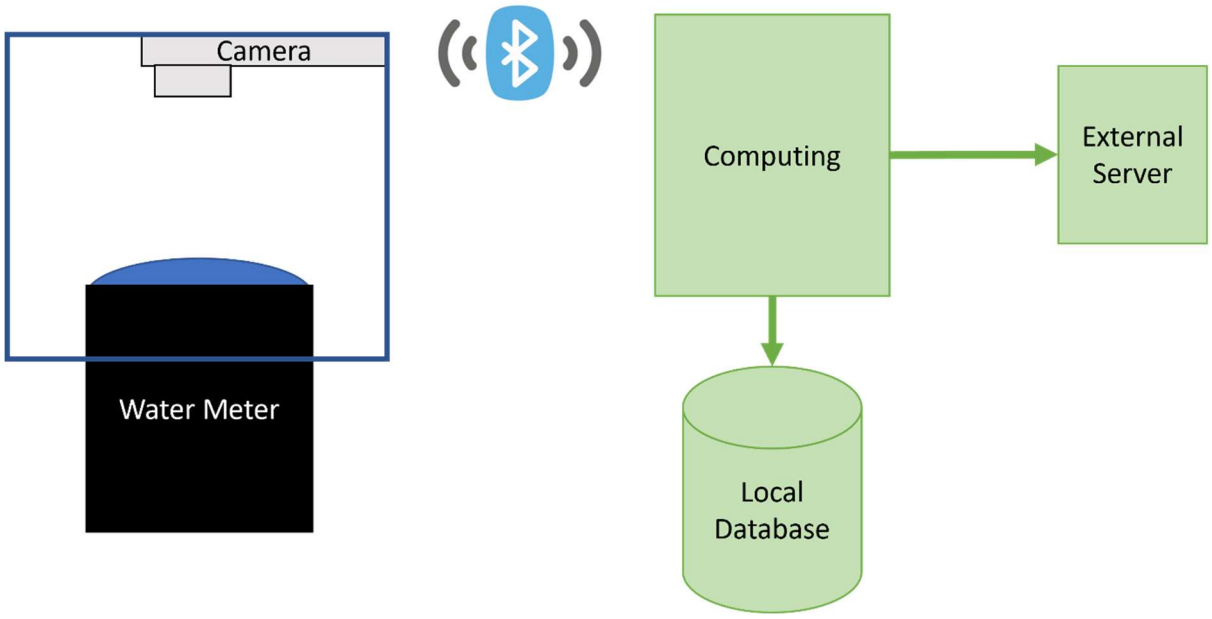


Figure 1 – Essential Architecture.

The device can also be used in condominiums where there are many water meters, for example, at the entrance (Figure 2).

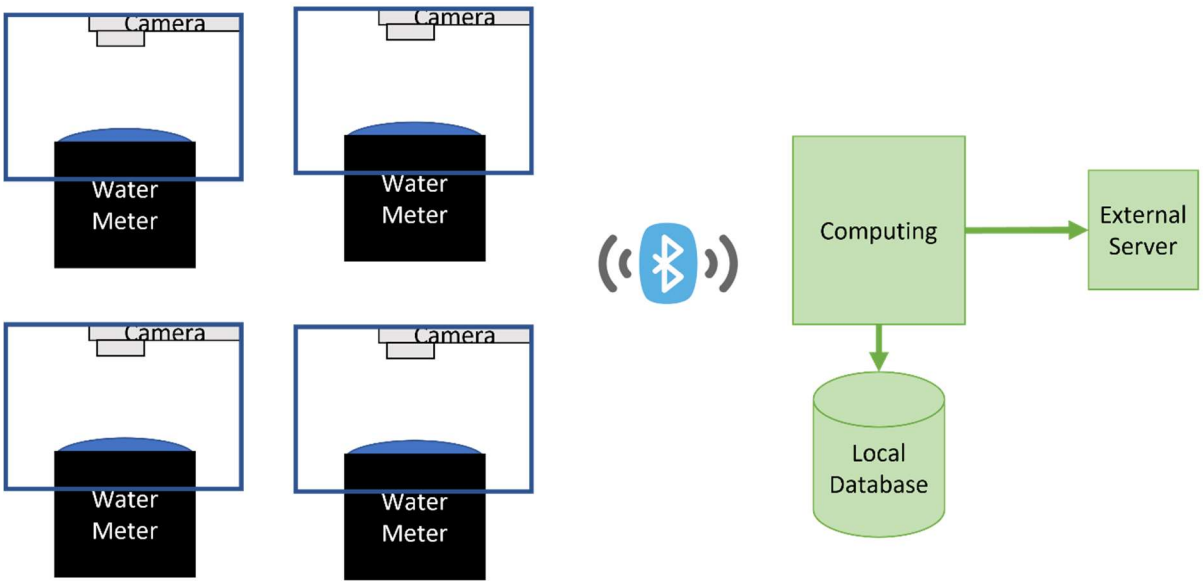


Figure 2 – Essential Architecture for multiple water meters.

In this case, it is possible to use a single computing device and as many cameras as the number of water meters that have to be read.

1.3 Document structure

This project report is structured in the following chapters:

Chapter 1, "Introduction", reports that water consumption is increasing, and above all that, there are many losses along the distribution. These leaks can be identified by improving water metering systems. Subsequently, the motivations and objective of this project are illustrated.

Chapter 2, "State of the Art", firstly describes the conventional water metering systems, then analyses the AMR meters and AMR smart meters. Next, an overview is given on M-Bus. The topic of Artificial Intelligence is subsequently discussed, going into more and more detail until the description of the LSTM Neural Network. The next sections cover OCR, Tesseract, OpenCV and Internet of Things.

Chapter 3, "Architecture of the Implemented System", begins by presenting the generic architecture of the system developed for this project. Subsequently, all the components are analysed: the ESP-EYE, the Raspberry Pi 4, the TTGO LoRa32 and the communications between them (USB and Bluetooth). After an analysis of the ISO/OSI system, all the protocols used in this project are explored: Wi-Fi, TCP/IP, MQTT, LoRa and LoRaWAN. Next, the chapter deals with the servers TTN and Akenza.

Chapter 4, "Implemented System", begins by illustrating the initial procedures for using the Raspberry, TTN, Akenza, Tesseract and OpenCV. Subsequently, the entire

procedure for training Tesseract and the algorithms used to recognise the digits are described. Finally, the chapter analyses the software developed for the three devices: the ESP-EYE, the Raspberry and the LoRa32.

Chapter 5, "Experimental and Field Tests Results", presents both various tests to demonstrate the correct functioning of the individual parts of the system and a complete test during which the developed system was used on a real functioning water meter.

Finally, in Chapter 6, "Conclusion and Future work", the conclusions are drawn and the future developments that the system may receive are exposed.

2 STATE OF THE ART

This chapter begins with presenting the conventional water metering systems and their main typologies. Next, the AMR meters and AMR Smart meters are analysed. The chapter continues with an overview of Artificial Intelligence, exploring all the necessary information up to the description of the Long Short-Term Memory (LSTM) neural network used by the OCR Tesseract. Finally, it deals with the Open-Source Computer Vision Library (OpenCV) and IoT.

2.1 Conventional Water Metering systems

An important device in the management of water supply networks is the water meter, which is used to measure the volume of water supplied from a public water distribution system to a residential or commercial building [2]. In general, there are three types of water meters, namely mechanical water meters, electromechanical water meters and fully electronic water meters. Table 1 briefly summarises advantages and drawbacks of these three major types of water meters.

Table 1 – Comparison of conventional water meters [2].

Types	Advantages	Drawbacks
Fully Mechanical	Simple design, low-cost reliable operation	Narrow measurement range, reduced accuracy at low flow rates cumulative measurement only, lack of real-time information
Electromechanical	Real-time information	Requires extra protection for the electronic component reduced stability
Fully Electronic	High accuracy, real-time information	Requires extra waterproof protection and power supply

Most countries adopt fully mechanical water meters due to their low cost and good reliability. However, they require labour intensive manual meter reading. For fully mechanical meters, the measurement of the water flow can be based on its velocity or displacement.

In the past two decades, electronic circuit components were progressively integrated into mechanical water meters. These are known as the electromechanical water meters and their measurement principle is still mechanical. For this category, an example are the turbine-based meters.

Recently, fully electronic water meters were designed using new measurement principles, such as electromagnetic and ultrasonic meters.

The next subsections analyse the differences in their functioning and their advantages and disadvantages.

2.1.1 Positive Displacement Flow Meters

Positive Displacement (PD) flow meters are the only flow measuring technology to directly measure the volume of fluid that passes though the flow meter. It achieves this by trapping pockets of fluid between rotating components housed within a high precision chamber [3]. This can be compared to repeatedly filling a beaker with fluid and pouring the contents downstream while counting the number of times the beaker is filled (Figure 3). The rotational velocity of the rotor is directly proportional to the flow rate, since the flow of fluid is causing the rotation. In electronic flow meters, the rotating components contain magnets that activate various sensors located outside the fluid chamber. Mechanical positive displacement flow sensors rely on the rotation to drive either a magnetic coupling or a direct gear train connected to the mechanical counter. PD flow meters do not require a power supply for their operation and do not require straight upstream and downstream pipe runs for their installation. The slippage between the flow meter components is reduced and the metering accuracy is, therefore, increased as the viscosity of the process fluid increases. Since the process fluid must be clean, the particles greater than 100 microns in size must be removed by filtering. Millions of such units are produced annually and they are most widely used as household water meters.

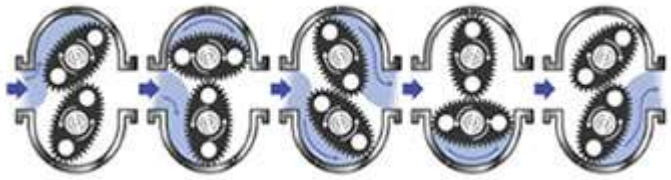


Figure 3 – Positive Displacement flow meter [3].

One of the main benefits of using a PD flow meter is the high level of accuracy that it offers. The high precision of the internal components means that the clearances between sealing faces are kept to a minimum. The smaller these clearances are, the higher the accuracy will be. The only fluid that is able to bypass this seal does not get counted and is known as 'by-pass' or 'slippage'.

If the correct flowmeter has been selected for an application, it can be expected to perform without error for many years. Frequently PD flow meters are sent in for servicing and recalibration that have been in the field for 10 sometimes 20 years of

continuous use. This reliability is largely due to the fact that this same proven technology has been in use for over 60 years, allowing the major advances to be focused within the fields of tribology and achieving the required precision at a reasonable cost.

The level of recommended maintenance is heavily influenced by the application. For example, if a flow meter is processing a fluid that shows lubricating properties, i.e. oil, then the routine maintenance can virtually be eliminated. If, however, the fluid has very poor lubricating properties, then it would be preferable to discuss the maintenance requirements with the distributor. It is very uncommon that any maintenance to a positive displacement flow meter would be more frequent than that of other equipment within the same system.

2.1.2 Turbine Flow Meters

The mechanical part of the Turbine flow meter has a turbine rotor placed in the path of a flowing stream [4]. The only moving part of the Turbine meter is the mechanical rotor. The rotational speed of the rotor depends on the flow velocity. The rotor blades are usually made of stainless steel. In most Turbine flow meters, the magnets are fitted to the blades and a magnetic pickup sensor is used to create the pulses, as it is shown in Figure 4.

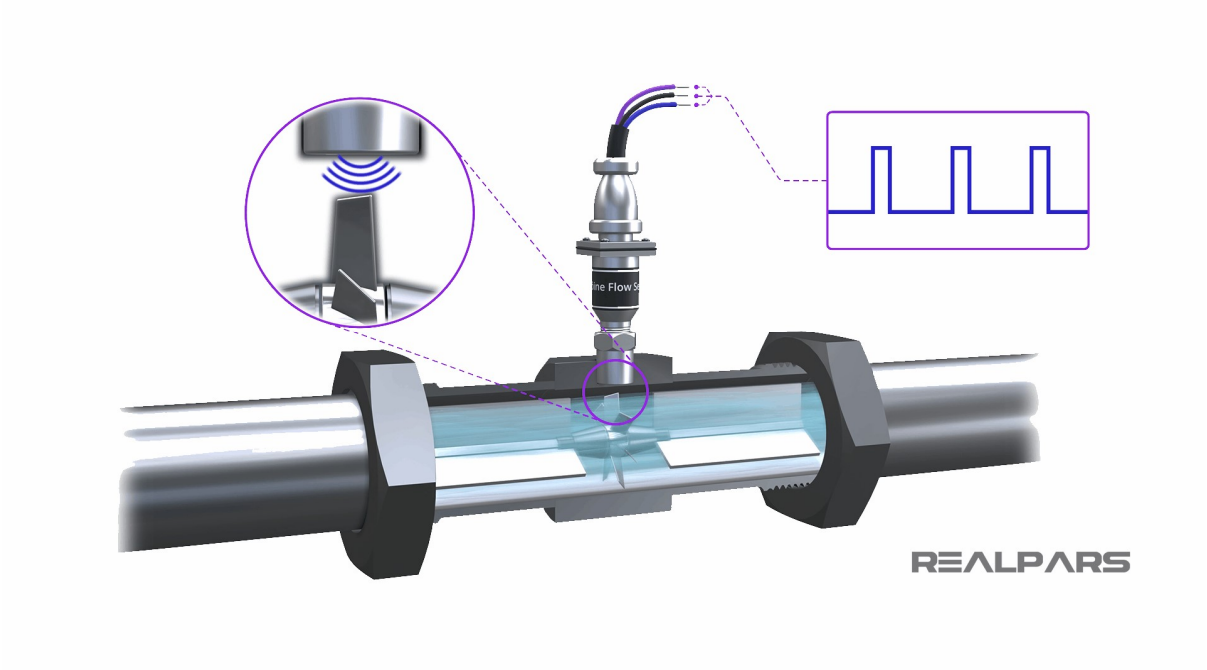


Figure 4 – Turbine flow meter [4].

The higher the rate of flow, the faster the rotor turns and the greater is the number of pulses.

The Turbine meter can only be used in clean, lubricating fluid, because suspended particles can easily damage the device. The turbine rotor must be positioned in the exact centre of the flow and the laminar flow is critical, often requiring straightening vanes.

There are mainly two different categories of Turbine meters: Single-jet and Multi-jet (Figure 5).

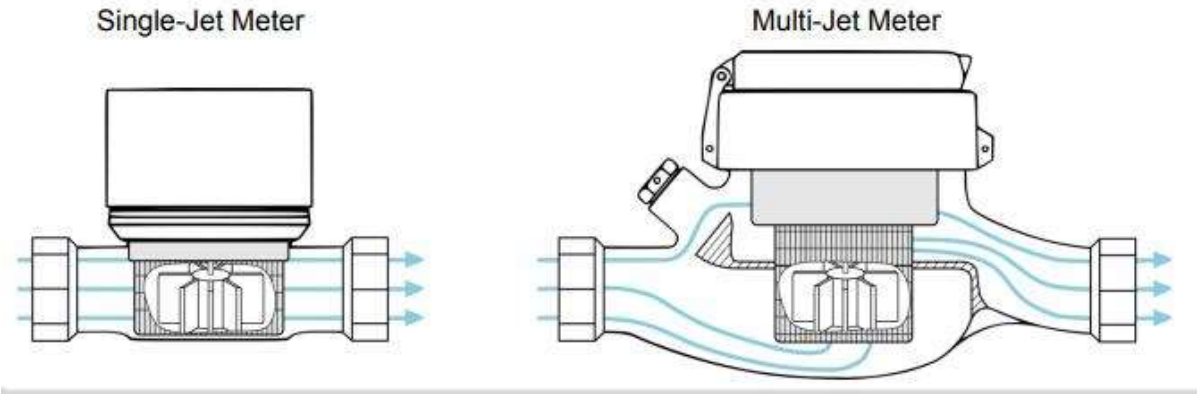


Figure 5 – Single and Multi-Jet flow meters [5].

Single-Jet Meters

Single-Jet meters are a low-cost option, because there is a direct impact to the rotor by the water flow [5]. They make use of one port to create a jet of water, which makes the turbine rotate. There is a single stream of water directed towards a turbine mounted in a radial position within the meter. The turbine starts rotating and transmitting the motion to the display mechanism, which allows to measure the volume of water passing through the meter.

Single-Jet water meters are simple, have an internal filtering element that protects the jet of the meter from clogging. Their bodies and outer casings are typically made of bronze or plastic, while the internal measuring parts are made of stainless steel and thermoplastics, thus making them highly reliable products with a competitive price.

Multi-Jet Meters

Unlike the Single-Jet meters, which have one port, the Multi-Jet meters have multiple ports that surround an internal chamber to create a flow of water against the turbine. Because the load is evenly placed across the rotor, Multi-Jet meters retain accuracy even at lower flow rates and are more long-lasting. Because of their design, Multi-Jet meters are ideal for diameters between 1-1/2" to 3", for large water users, for water utilities, as well as for industrial water installations.

Multi-Jet water meters also have an internal strainer that protects the meter's jet ports from clogging. Their bodies and outer casings are typically made of bronze alloy, and the internal measuring parts are made of stainless steel and modern thermoplastics.

2.1.3 Electromagnetic Flow Meters

Electromagnetic flow meters (Figure 6), also called Mag meters or Magnetic meters, are volumetric flow meters which do not have any moving parts to wear, thus reducing the need for maintenance or replacement [6].

Figure 6 – Electromagnetic flow meter [6].

Accuracy over a wide flow range can be as good as ±0.5% of flow rate or better. They feature an obstruction-free design, which eliminates flow impediment and operate with electrodes embedded on opposite sides of the flow tube or with a sensor, that pick up the signal. Mag meters perform extremely well and have become the meter of choice for measuring conductive liquids, such as water or slurry. All the Mag meters operate under the principle of Faraday's Law of Electromagnetic Induction to measure the velocity of the liquid. The principle of operation states that a conductor moving through a magnetic field produces an electric signal within the conductor, which is directly proportional to the velocity of the water moving through the field. In fact, as fluid flows through the magnetic field, conductive particles in the fluid create changes in voltage across the magnetic field. This variation is used to measure and calculate the velocity of water flow through the pipe.

The main advantages of using electromagnetic are that they require less maintenance and can be ordered for very large line sizes. They can be installed with a simple hot tap, which does not require shutting down the line. Mag meters also have the benefit of adding a level of trust for consumers: they hold the service provider accountable for the actual water being produced and/or billed.

2.1.4 Ultrasonic Flow Meters

An Ultrasonic flow meter utilises ultrasound to measure the velocity of a fluid and is employed in a variety of fluid applications [7]. Ultrasonic flowmeters are ideal for water and other liquids. Ultrasonic flow meters achieve high accuracy at low and high flows, save time with no pipe cutting or process shutdown and are not affected by external noise. Ultrasonic flow meters use sound waves at a frequency beyond the range of hearing (typically 0.5, 1, or 4 MHz). This ultrasound signal is sent into a stream of flowing liquid by using (Insertion) transducers that make direct contact with the liquid or external (Clamp-on) transducers that send the ultrasound through the pipe wall. Clamp-on Ultrasonic flow meters allow users to measure the volumetric flow rate of a fluid in a pipe, without having to penetrate the pipe, thus decreasing installation and maintenance costs.

A typical Transit-time Ultrasonic liquid flow meter utilises two ultrasonic transducers that function as both ultrasonic transmitter and receiver (Figure 7). The Ultrasonic flow meter operates by alternately transmitting and receiving a burst of ultrasound between the two transducers and by measuring the transit time that it takes for sound to travel between the two transducers in both directions. The difference in the transit time (Δtime) measured is directly proportional to the velocity of the liquid in the pipe.

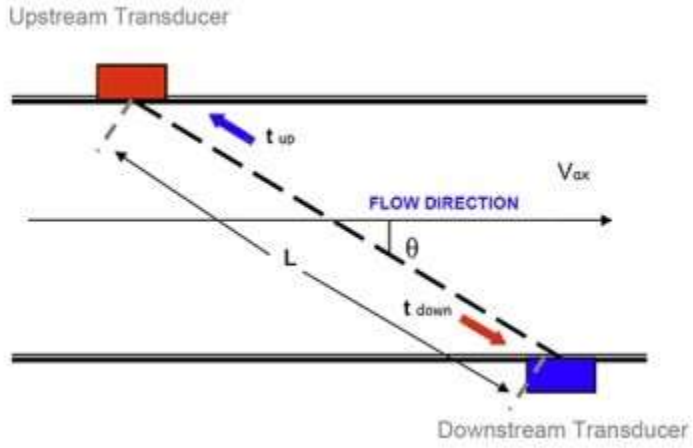


Figure 7 – Ultrasonic flow meter [7].

Figure 8 is a drawing of a typical application using the most common, V (2 pass) mounting method with clamp-on transducers.

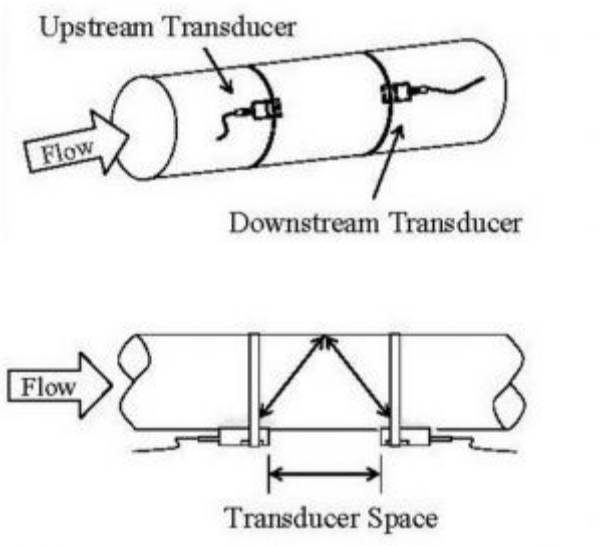


Figure 8 – Ultrasonic flow meter, V mounting method [7].

In this application, the ultrasound is transmitted from the first transducer and travels through the pipe wall, through the liquid, then reflects off the back wall of the pipe, travels through the pipe wall, and is picked up by the second transducer.

The same process is, subsequently, repeated in reverse when the second transducer transmits the ultrasound. The time difference between the times of flight up and down is the ΔTime. When the liquid in the pipe is not moving, the ΔTime is zero.

To calculate the velocity of the liquid, it is necessary to convert the raw ΔTime into the velocity of the liquid in the pipe. The angle of the ultrasound path is calculated by knowing the speed of sound in the pipe and the liquid. This angle is used in a trigonometry equation to convert the ultrasound path into a straight line in the pipe, which allows to obtain the velocity of the liquid in the pipe.

Once the velocity is known, it is necessary to calculate the flow rate by multiplying the velocity times the cross-sectional area of the pipe, as with any velocity-based flow meter.

The largest benefit of any Clamp-on Transit-time flow meter is the simplicity of installation. The process does not require shutting the system down to install it, making maintenance of Transit-time meters much more advantageous.

There are no blades that can wear out or bearings that may need a replacement as in turbine meters, and also there are no electrodes that can be damaged over time as in magnetic flow meters, meaning that a low maintenance is required for this type of meters.

2.2 AMR Meter and AMR Smart Meter

Traditionally, electromechanical meters or basic electronic meters have been used to measure the water consumption. These types of meters require sending the suppliers to the water meter location for meter readings and other management tasks such as meter disconnection. This is complicate, considering the high number of customers. The service provider person goes to the place of the meter and notes down the reading at the end of every billing cycle. But the traditional meter reading process not only wastes human labour, but also is error-prone. The procedures of sending the bills to the customers are very laborious, cumbersome and time-consuming. Another major problem in this system is that the readings cannot be taken if no one is available at the home or where the meter is located.

2.2.1 Automatic Meter Reading (AMR)

An AMR meter works by creating a connection channel between a business customer and its water supplier [8].

For an AMR meter, the communication only goes from the meter to the supplier. The water supplier, in fact, receives meter readings once per month, so there is no need for manual meter readings, thus ensuring accurate billing and allowing the customers to analyse their water usage data.

The main benefits of AMR meters over conventional meters are:

- they send accurate meter readings to the energy supplier so there are no more estimated bills;
- improved security and tamper detection for equipment.

In general, the transmission, or telemetry, of data and control signals between the meter interface units and the metering system is done with various methods as summarised in Table 2, namely touchpad, mobile data networks such as Global System for Mobile Communications (GSM), Walk-by, Drive-by and Power Line Carrier (PLC) [9].

Table 2 – Communication methods for meter reading [9].

Type	Disadvantages	Advantages
Touch Technology	Requires meter readers. Higher personal costs.	Simple to install and operate. Low initial investment.
GSM	Dependent on an external service provider. Requires cell coverage.	Read frequency selectable. Consumer-friendly.
Walk-by and Drive-by	Quality of radio signal dependent on many factors.	More advantageous than touch pad.
Power Line Carrier	Reliability of signal transmission.	Relatively inexpensive. Two-way communication to multiple devices.

Touch Technology

With touch-based AMR technology, a meter reader carries a handheld computer or data collection device with a probe [9]. The device automatically collects the readings from a meter, by touching or placing the reading probe in close proximity to a reading coil enclosed in the meter. When a button is pressed, the probe sends an interrogating signal to the meter to collect the meter reading.

GSM

The GSM network provides a global coverage across countries, thus enabling communication to every corner, without the need to implement a new communication infrastructure solely for this purpose [10]. Apart from seamless coverage, the GSM technology also provides services like Short Message Service (SMS) for requesting of reading from individual houses back to the water provider, wirelessly. Moreover, the GSM is a more efficient, reliable and secure communication standard, that is being widely used for more than several years now without any technical issues. Other features of the GSM system are the low cost, the simple setup and the wide operating distance.

Walk-by and Drive-by

A meter reader carries a handheld computer with a built-in or attached receiver/transceiver to collect meter readings from an AMR meter [11]. Since the

meter reader walks by the locations where meters are installed, this is referred to as Walk-by meter reading.

On the contrary, if the reading device is installed in a vehicle, it is called mobile or Drive-by meter reading. The meter reader drives the vehicle, while the reading device automatically collects the meter readings. Often, for mobile meter reading, the reading equipment includes navigational and mapping features provided by Global Positioning System (GPS) and mapping software. With mobile meter reading, the reader does not normally need to read the meters in any particular route order, but just drives the service area until all meters are read. Components often consist of a laptop with a specific program, a RF receiver/transceiver, and external vehicle antennas.

Power Line Carrier (PLC)

The PLC technology allows data to be transferred between the smart meter and the utility company using the utility power lines [10]. PLC is cost-effective for rural lines, so it is employed over long distances. Compared to the RF technology, PLC has long latencies for data transmission, less bandwidth and a higher cost in cities.

2.2.2 AMR Smart meter

A smart meter is an electronic device used to record and transmit the information of water consumption, and it is the most technologically advanced version of the AMR technology [12]. An AMR smart meter has to be produced according to an industry standard, referred to as the Smart Metering Equipment Technical Standard (SMETS). The latest generation of the AMR smart meters is SMETS2, which features many benefits over both conventional and AMR meters.

They share the same functionality of AMR meters by automatically sending a meter reading and diagnostic data to the supplier; however, they use a centralised Data Communication Company (DCC) for their communications to the supplier. This allows them to both send and receive messages from their provider.

The AMR smart meter also comes with an optional Smart Energy Display, which shows exactly in real time how much water is being used. This allows businesses to have more control over their water consumption and bills.

The benefits of a smart meter are:

- it sends accurate meter readings to the supplier, so there are no more estimated bills;
- it offers improved security and tamper detection for equipment;
- it gives customers the option of having a smart display, which shows real time water usage in terms of money;

- it gives customers feedback on their water usage, allowing them to adjust their habits in order to lower their water bills;
- it can be operated as a pre-payment meter, if in need of help managing the personal payments and budgeting.

Figure 9 illustrates the difference between AMR smart meters and conventional meters. Smart meters provide efficient communication between customers and utility companies. By using smart meters, the system will automatically send the information from the client unit to the central unit and vice versa. The utility company has the ability to produce a billing for the customer, which has all the information about the water consumption.

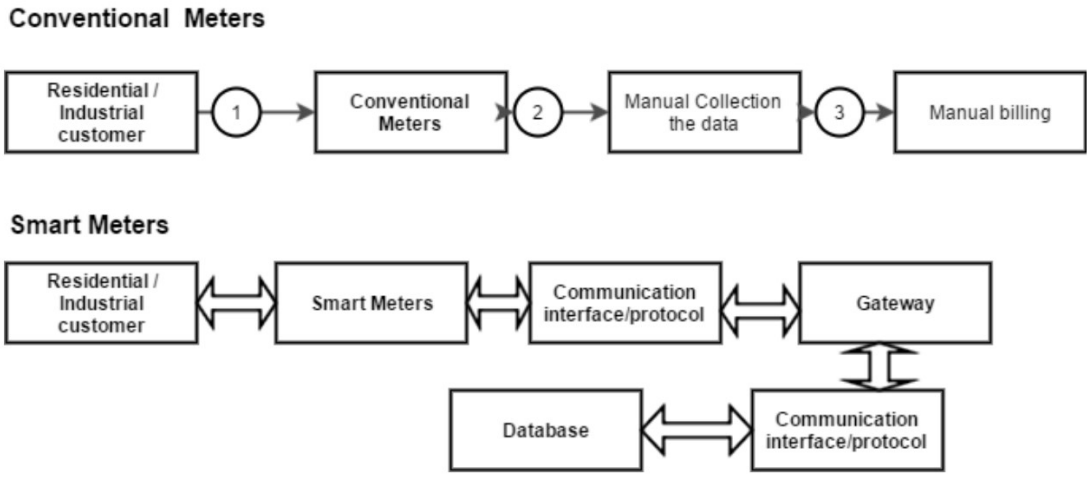


Figure 9 – Conventional Meters Vs. Smart Meters [13].

Smart meters have become increasingly popular for calculating, controlling and measuring the water consumption [13]. The data is, then, normally stored on a server, which will be used for further operations like calculating the consumption fees, displaying consumption statistics or other information to the customer. Using the information from the smart meter, significant energy and financial savings can be achieved. The large-scale installations of smart meters generate a massive amount of data, which can offer the company a unique insight of the water consumption of different costumers. This information can be used to help consumers shift their consumption from peak hours, which can result in significant savings of the water. Moreover, this information can be used to detect the water leakage and even to individuate where it happens.

2.3 Meter-Bus (M-bus)

The Meter-Bus (M-bus) is the Standard for Remote Reading of Smart Meters and was developed by Professor Dr. Horst Ziegler of the University of Paderborn in cooperation with Texas Instruments Deutschland GmbH and Techem GmbH [14].

The concept was based on the ISO-OSI Reference Model, in order to realise an open system (Figure 10).

Layer	Functions	Standard	Chapter
Application	Data structures, data types, actions	EN1434-3	6
Presentation	empty		
Session	empty		
Transport	empty		
Network	extended addressing (optional)	-	7
Data Link	transmission parameters, telegram formats, addressing, data integrity	IEC 870	5
Physical	cable, bit representation, bus extensions, topology, electrical specifications.	M-Bus	4

Figure 10 – The M-Bus layers in the OSI-Model [15].

Since the M-Bus is not a network and, therefore, does not need a transport or session layer, this means that the levels four to six of the OSI model are empty. Only the physical, the data link, the network and the application layer are provided with functions.

2.3.1 Principles of Operation

The M-Bus is a hierarchical system, where the communication is controlled by a master (Central Allocation Logic) [16]. The M-Bus consists of the master, a number of slaves (end-equipment meters) and a two-wire connecting cable, as illustrated in Figure 8. The slaves are connected in parallel to the transmission medium, that is to say the connecting cable.

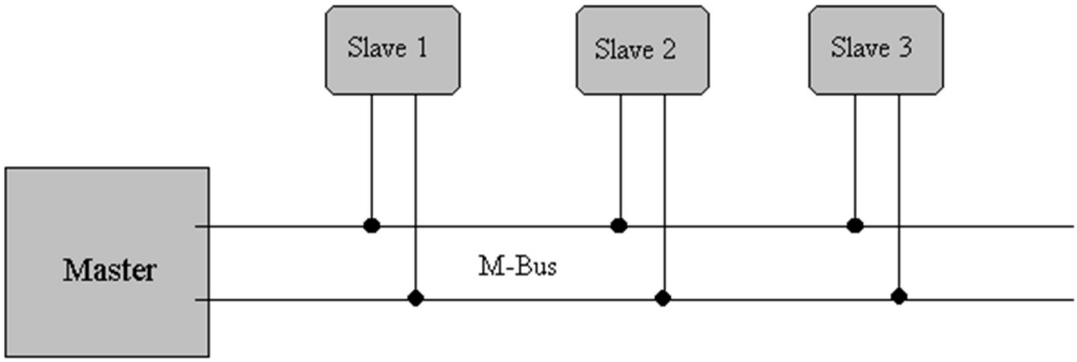


Figure 11 – Block diagram showing principle of the M-Bus System [13].

The communication bus is composed of a two-wire cable and the communication consists in a serial data transfer. The bits transmission on the bus is represented in Figure 12.

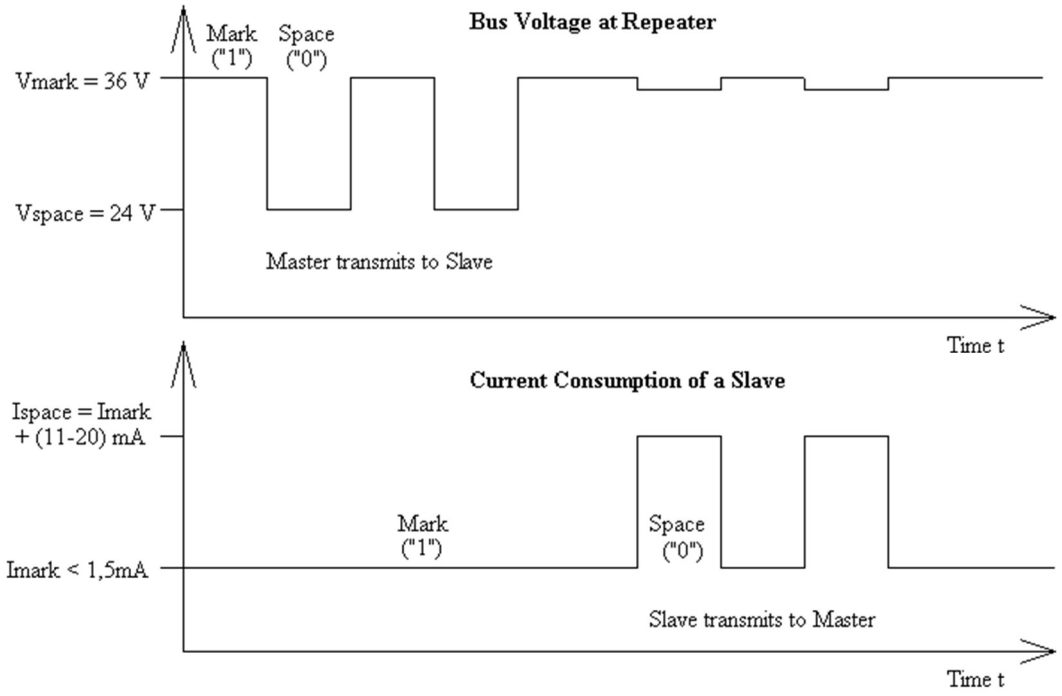


Figure 12 – M-Bus communication [16].

The transfer of bits from master to slave is accomplished by using two different voltage levels: the logical "1" (Mark) corresponds to a nominal voltage of +36 V; the logical "0" (Space) corresponds to a nominal voltage of +24 V.

On the other hand, the bits sent in the direction from the slave to the master are coded by modulating the current consumption of the slave. The logical "1" (Mark) is represented by a constant current of up to 1.5 mA and the logical "0" (Space) by increasing the current consumption of the slave of additional 11-20 mA. The Mark state current can be used to power the meter itself.

The transmission is possible only in one direction, either from the master to the slave or from the slave to the master (Half Duplex).

The protocol of the data link layer is based on the international standard IEC 870-5, which defines the transmission protocols for telecontrol equipment and systems.

2.4 Overview on Artificial Intelligence (AI)

This section is aimed at defining what Artificial Intelligence (AI) is, and at describing its aspects referred to this project. Figure 13 depicts an AI hierarchy.

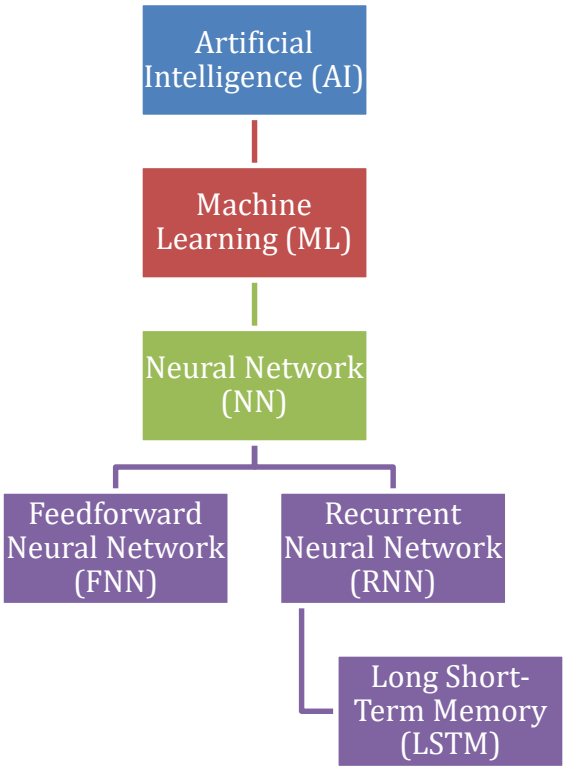


Figure 13 – Artificial Intelligence hierarchy.

The Artificial Intelligence is the main concept and Machine Learning is a specific application of it. A Neural Network is a machine learning algorithm. The most basic neural network is the Feedforward Neural Network (FNN), while the Recurrent Neural Network (RNN) and, in particular, the Long Short-Term Memory, are more advanced neural networks.

2.4.1 Artificial Intelligence

There is no generally accepted definition of Artificial Intelligence. It is difficult to define clearly what AI is, because it is an imitation of something still not fully understood: human intelligence [17]. A common definition is that AI is a technology that enables machines to imitate various complex human skills. In this definition those "complex human skills" are not specified. Other definitions go further in explaining these skills, for example the definition given by the High-Level Expert Group on Artificial Intelligence (AI HLEG) of the European Commission (EC) is: "Systems that display intelligent behaviour by analysing their environment and taking actions, with some degree of autonomy, to achieve specific goals". The conception

of AI is dated back to 1956, during a summer school at Dartmouth College in New Hampshire, but already had a long history before being seriously investigated as a scientific discipline. Figure 14 shows the three phases in AI history.

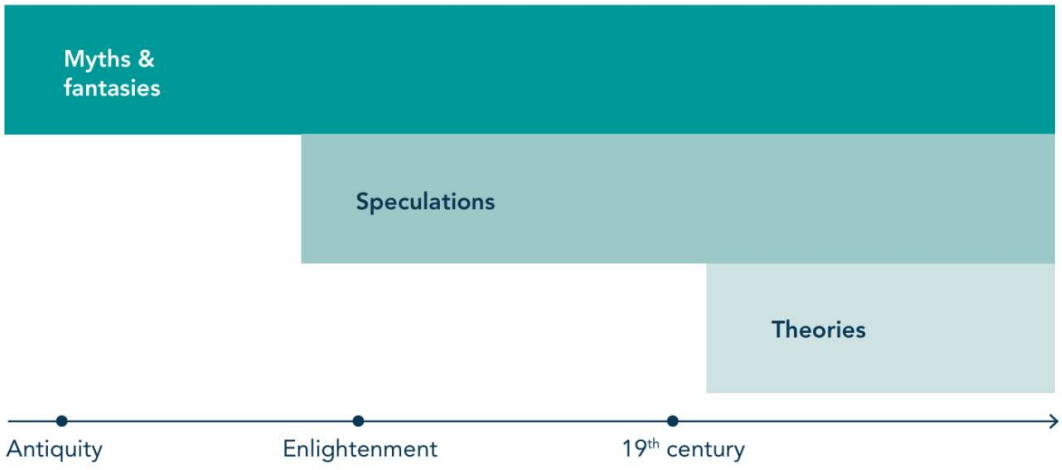


Figure 14 – Three phases of AI in the history [17].

The first phase (Myths and fantasies) already begins with the ancient Greeks. In their mythology, they celebrated a multitude of characters, who can be characterised as artificial forms of intelligence. The second phase (Speculations) begins with the "mechanisation of the world". AI was still far beyond the realm of possibility, but the new devices led to speculation about its creation which was no longer mythical, but mechanical in nature. The third phase (Theories) begins in the second half of the nineteenth century, in parallel with the theorisation and construction of the first computers. AI became less fantastical and received serious theoretical consideration.

The latest technology is based on AI, which has gained huge interest in recent years. One of the major drivers of this has been progress in a specific area of the field: Machine Learning (ML).

2.4.2 Machine Learning (ML)

Machine Learning is an application of AI that gives the systems the ability to automatically learn and improve from experience, without being explicitly programmed [18].

The Machine Learning algorithms are divided into four categories: Supervised Learning, Unsupervised Learning, Semi-supervised Learning, and Reinforcement Learning (Figure 15) [19].

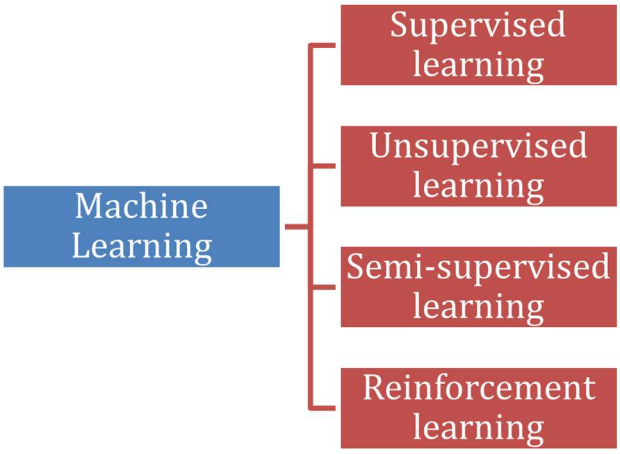


Figure 15 – Machine Learning algorithms.

With supervised learning, machine learning learns a function that links together an input and an output, based on labelled training data and a collection of training examples. The most common supervised tasks are: "classification", that separates the data and "regression", that fits the data.

Unsupervised learning analyses unlabelled datasets without the need for human interference and it is used for extracting generative features, identifying meaningful trends and structures, groupings in results, and for exploratory purposes.

Semi-supervised learning is a hybridization of the supervised and unsupervised methods and operates on both labelled and unlabelled data.

Reinforcement learning is a type of machine learning algorithm that enables machines to automatically evaluate the optimal behaviour in a particular context or environment, to improve its efficiency.

2.4.3 Neural Network (NN)

An artificial Neural Network is a machine learning algorithm based on the structure of the human brain [20]. The basic unit of the brain structure is the neuron (Figure 16).

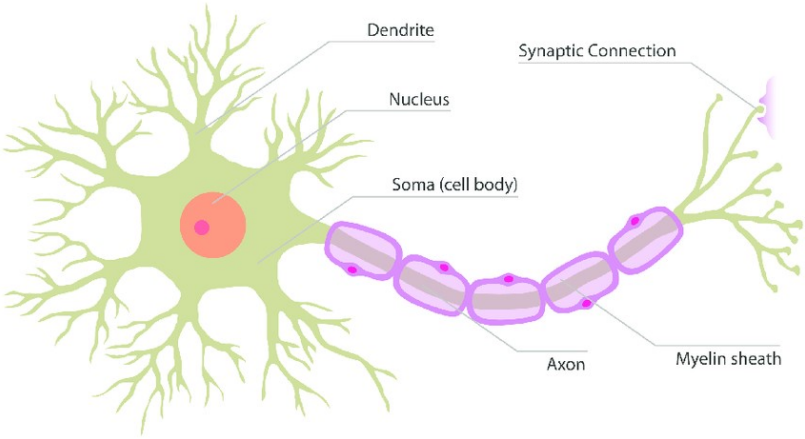


Figure 16 – Structure of a Real Neuron [21].

A neuron transmits an electric signal. The dendrites are the place where the input signals reach the neuron. The axon primarily carries nerve signals away from the soma. In this way, the electrical signals continue to be transmitted across the synapse from one neuron to another. A neuron receives inputs from multiple neurons and transmits signals to multiple neurons. The human brain has approximately 100 billion neurons [20].

An Artificial Neuron, represented in Figure 17, is the basic unit of an artificial neural network.

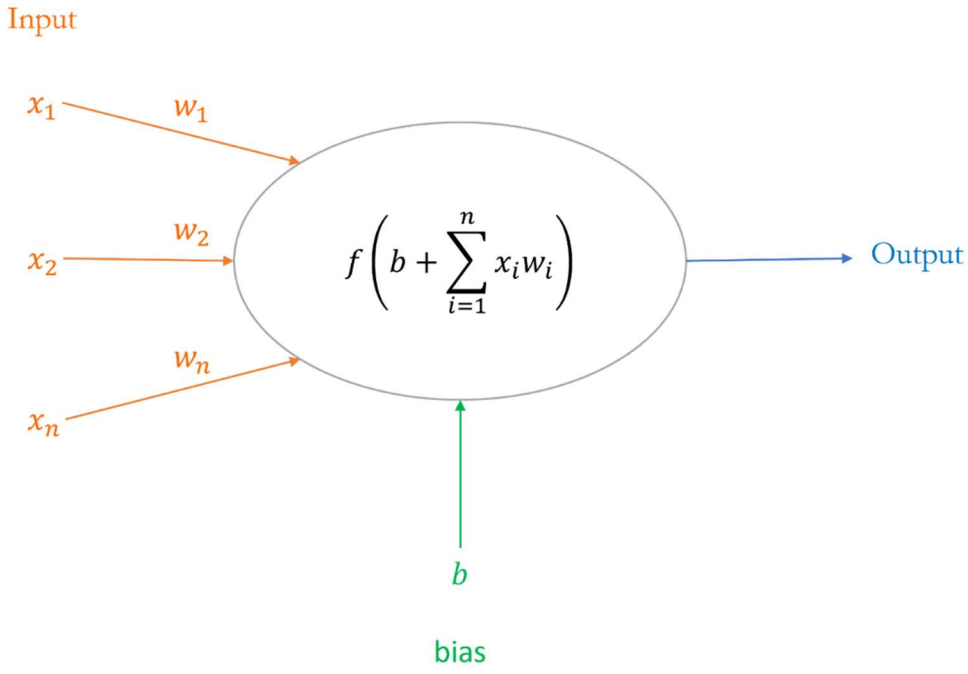


Figure 17 – Artificial Neuron. Adapted from [20].

Exactly as it is for a real neuron, the artificial neuron has a vector of many inputs $(x_1, x_2, \dots x_n)$. Each input has a weight associated with it $(w_1, w_2, \dots w_n)$ and there is also the bias (b). To obtain the output, the function f must be applied to the sum between the bias b and the weighted summation of the inputs. The function f in the simplest case is a step function (Figure 18), therefore the output becomes 1 if a certain threshold is exceeded. Other generally used functions are the Sigmoid function and the Hyperbolic function (Figure 18).

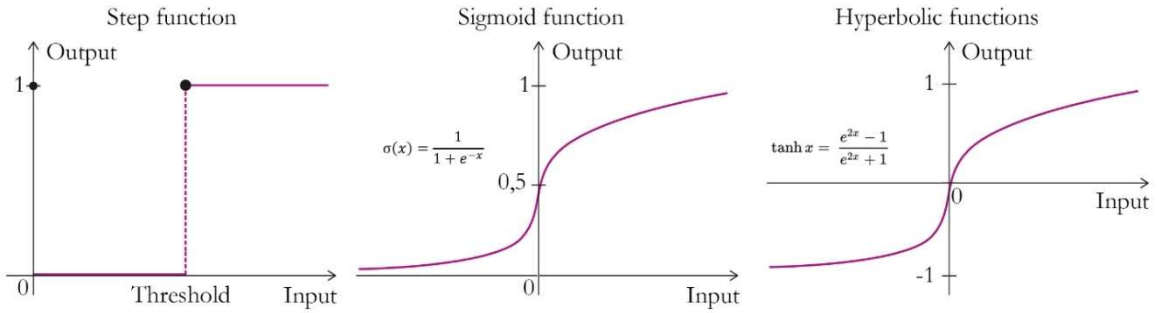


Figure 18 – Step, sigmoid and hyperbolic functions. Adapted from [20].

As shown in Figure 18, when the weighted sum is small, the output value is 0. If the weighted sum value rises above the threshold (step function) or it becomes quite big (sigmoid and hyperbolic functions), a non-zero output value appears. Thus, the responses of biological neurons and artificial neurons (nodes) are similar. The machine starts with arbitrary weights $w_1, w_2, \dots w_n$ values and alter them gradually, so that the cost (the difference between the predicted value and the answer obtained) is reduced until it reaches a minimum. For a binary classification problem, the outputs of 1 or 0 are used to indicate the two different classes. The limitation of the neuron stands in the fact that it is only capable of solving classification problems that are linearly separable, thus obtaining a separation by a line in a two-dimensional space (Figure 19) if there are only two inputs, a plane in three-dimensional space if there are three inputs and a hyperplane in p-dimensional space if the inputs are p.

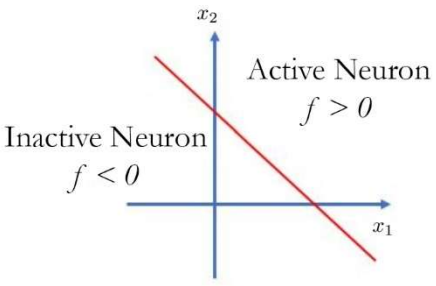


Figure 19 – Classification by a neuron with a step logic transfer function. Adapted from [22].

The position and inclination of the line depends on both the weights and the threshold used. The values of x_1 and x_2 that are to the right of the line activated the neuron, while for those that are to the left, the neuron remained inactive. The division can only be linear and this is a limitation that is overcome in Neural Networks. The artificial neuron is the base of the Neural Networks. Among the many types of NN, those of interest for this project are going to be analysed in the next subsections:

- Feedforward Neural Network;
- Recurrent Neural Network and, more in detail, a particular type of this category, namely the Long Short-Term Memory.

2.4.4 Feedforward Neural Network (FNN)

A Feedforward Neural Network is a layered structure composed of an input layer, one or more hidden layers with one or more neurons, and an output layer [23]. Figure 20 illustrates an example of FNN having two inputs, two hidden layers of two neurons each and a single neuron in the output layer.

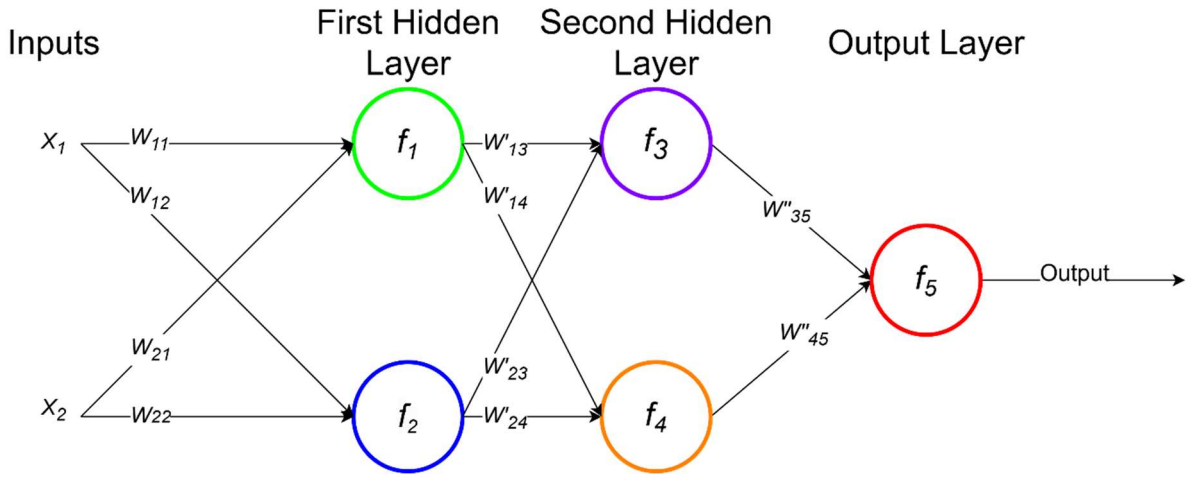


Figure 20 – Example of FNN. Adapted from [23].

The neurons are grouped into consecutive layers [24]. Neurons within the same layer do not interact, while all neurons between two consecutive layers are connected to each other. In a FNN there are not back-action or cycles. Each neuron in layer i receives inputs at the time t, and the outputs of the neurons are forwarded to the neurons of the next layer $i + 1$ only. After a cascade of hidden layers, the user inputs data arrives to the output layer, that returns the response of the FNN. The FNN architecture is used for tasks like pattern recognition and classification.

The two neurons (green and blue) of the first hidden layer act in parallel, therefore, in the specific case of only two inputs, both produce a separation line in the input

space, as shown in Figure 21. The inclination and position of the two lines depend on the weights and thresholds used.

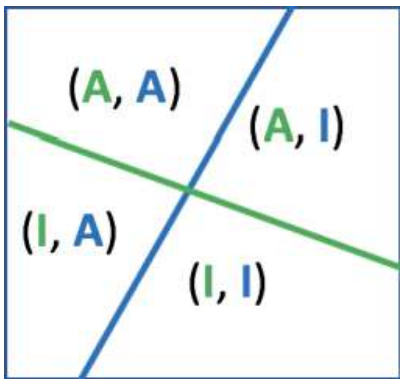


Figure 21 – Input space division by two parallel neurons. Adapted from [24].

The green line is produced by one neuron and the blue line by the other one. A and I stand for Active Neuron and Inactive Neuron.

The output of these two neurons becomes the input of both neurons (orange and violet) of the subsequent hidden layer. These two neurons of the second hidden layer divide the previous four subareas into four parts, as illustrated in Figure 22. If only one neuron is considered, for example the orange line, it is possible to see that the input space has been divided into two parts, but by a bent line (purple or orange) and no longer by a straight line.

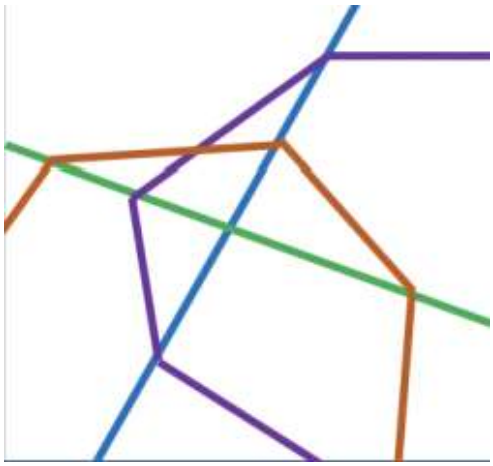


Figure 22 – Input space division by two hidden layers made up of two neurons each. Adapted from [24].

The two outputs coming out respectively from the two neurons are inputs to the last neuron (red), which belongs to the output layer. Since this is a single neuron, all the previous subareas are divided into two parts.

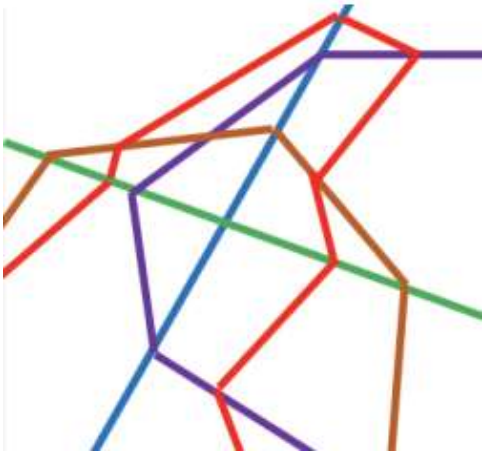


Figure 23 – Input space division by two hidden layers made up of two neurons each and a single neuron output layer. Adapted from [24].

Looking only at the red line, the output layer has divided the entire input space into two parts. However, unlike what a single neuron would do, this time the division is not linear, instead it is a piecewise continuous linear function. If the neurons had used an activation function other than the step, e.g. sigmoid or hyperbolic tangent, the lines obtained would have been smooth.

One way to use this network to determine a digit present in an image is described below [25]. Considering having an image of 28×28=784 pixels. To analyse this image, it is necessary to have an FNN with 784 inputs that go to all the neurons of the hidden layer (each input with its weight), as illustrated in Figure 24.

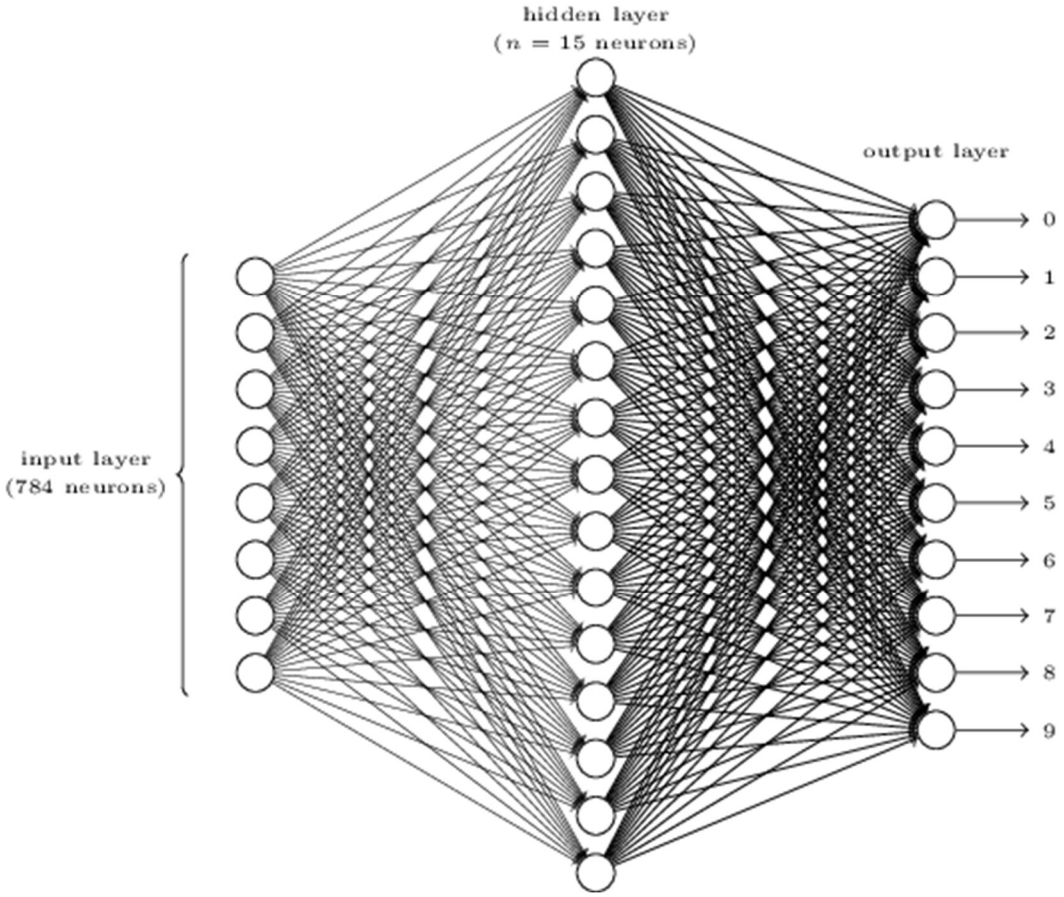


Figure 24 – FNN with 784 inputs, 15 neurons in hidden layer and 10 neurons in output layer [25].

The output neurons are numbered from 0 to 9. The output with the highest activation indicates which digit was recognised in the input image.

2.4.5 Recurrent Neural Network (RNN)

Recurrent neural networks are a superset of feedforward neural networks, augmented with the ability to pass information across time steps [26]. Figure 25 represents a simple example of RNN.

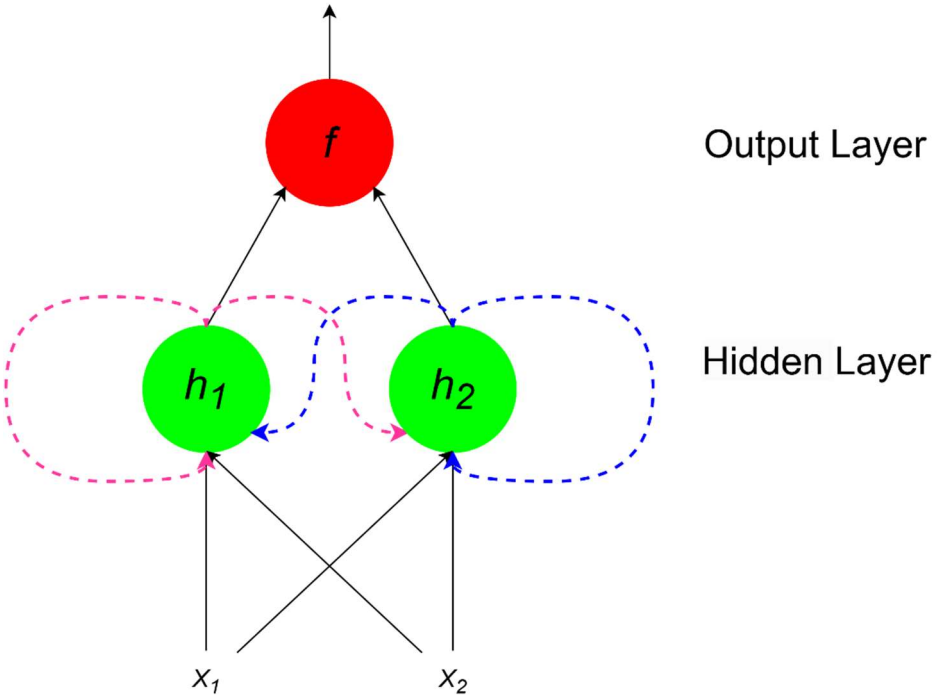


Figure 25 – RNN Example. Adapted from [26].

Analysing neuron 1 of the hidden layer, its output is given by:

$$h_1(t) = f_1\left(b_1 + \sum_{i=1}^{n} x_i w_{i1} + \sum_{i=1}^{n} h_i(t-1) w_{i1}\right)$$

The output of a neuron at time t depends on its activation function applied to the sum between the bias, the weighted summation of the inputs, its output and the output of the second neuron, both at time $t-1$. This can be observed in Figure 26.

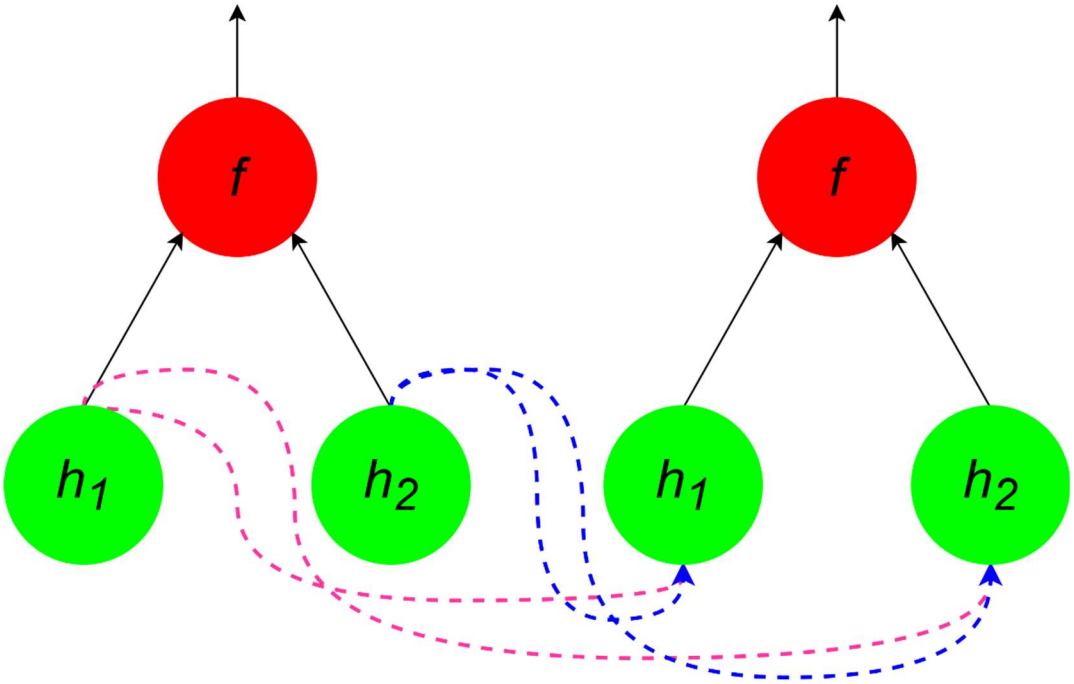


Figure 26 – RNN evolution in time. Adapted from [26].

Figure 26 shows two successive times, and in this way it possible to notice how the outputs at $t = 1$ become inputs at $t = 2$.

Figure 27 depicts vanishing gradient problem of this type of NN.

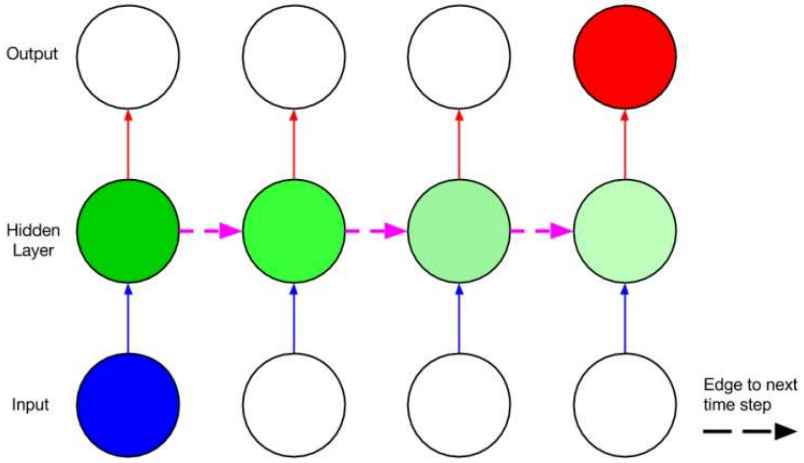
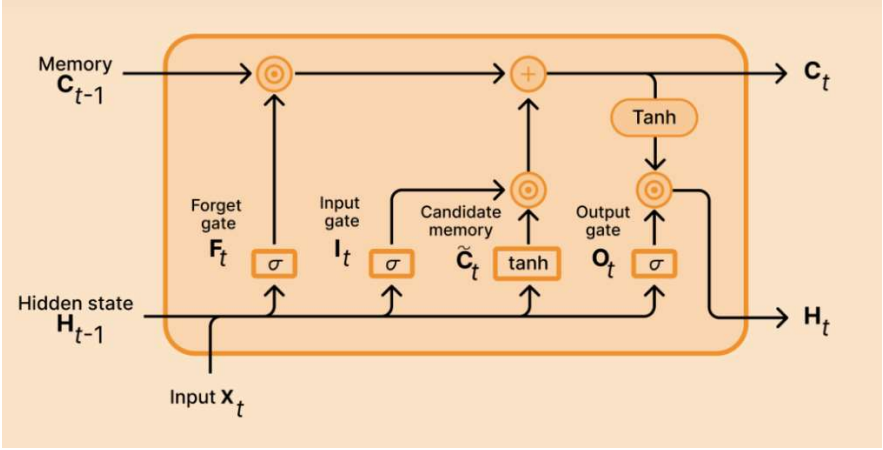


Figure 27 – Vanishing gradient problem [26].

If the weight used for $h(t-1)$ is less than 1, the effect of the input $x(t=0)$ will be very small at $t = 3$. This problem is solved with Long Short-Term Memory NNs, which are going to be analysed in detail in the next subsection.

2.4.6 Long Short-Term Memory (LSTM)

LSTMs are able to remember an information for a long time [27]. The typical LSTM module has four neural network layers interacting, how illustrated in Figure 28. The fundamental component of LSTMs is the c_t line, which runs from the Memory of the Previous Block, to the Memory of the Current Block. It allows the information to flow straight down the line. The network can decide the amount of previous information that has to flow.



Symbol	σ	tanh			Tanh
Description	Neural Networks		Pointwise operations		

Figure 28 – LSTM Architecture [28].

The pointwise multiplication between the vectors $[x_1\ x_2\ x_3]$ and $[y_1\ y_2\ y_3]$ is $[x_1y_1\ x_2y_2\ x_3y_3]$, the pointwise addition is $[x_1{+}y_1\ x_2{+}y_2\ x_3{+}y_3]$, and the pointwise $tanh$ is $[tanh(x_1)\ tanh(x_2)\ tanh(x_3)]$.

Figure 29 represents a more detailed description of the LSTM architecture.

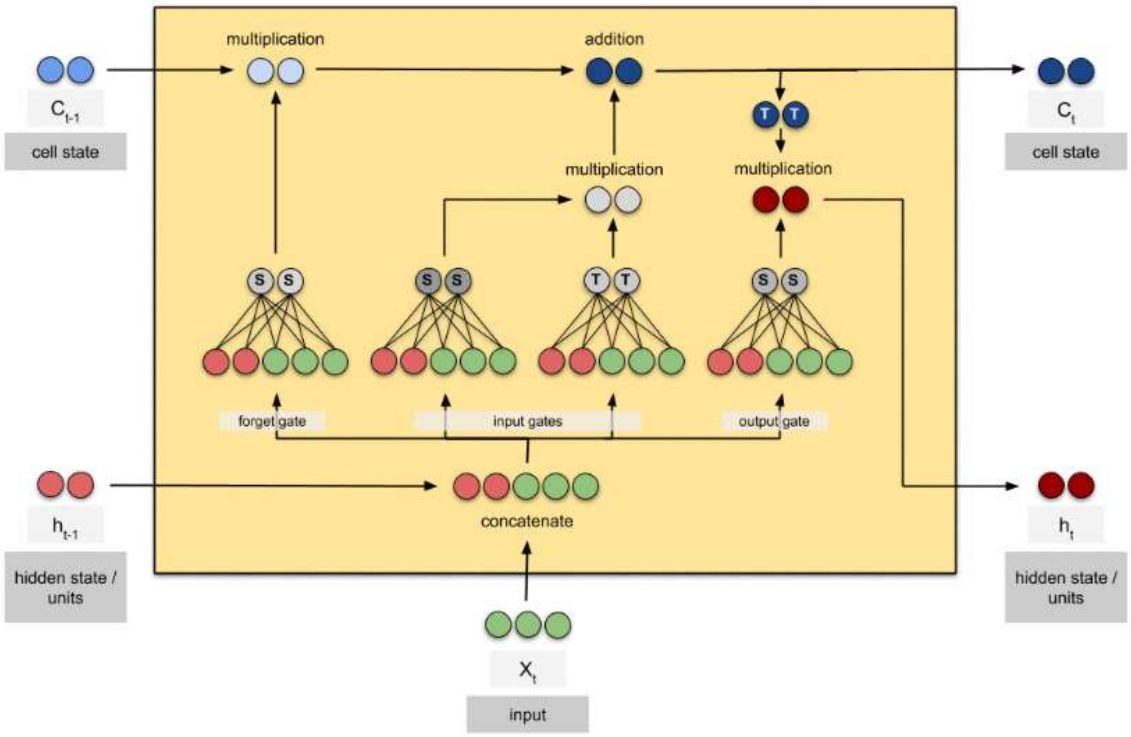


Figure 29 – Detailed LSTM Architecture [29].

The forget gate is used to decide which information of c_{t-1} should be forgotten. The neural network of this first gate has two neurons and has x_t and h_{t-1} as inputs. Since the two neurons behave as described before, the output of this neural network is a vector of two elements, and each element can assume the value of 0 or 1. By doing the pointwise multiplication between the output of this first neural network and the vector of two elements of c_{t-1}, a two-element vector is obtained. This vector maintains only the values that have been multiplied by 1, while reducing to 0 the elements that have been multiplied by 0.

The input gate decides which new elements should be stored in c_t. The sigmoid neural network of this gate produces a vector of two elements, and each element has a value between 1 and 0. The $tanh$ produces a vector of two elements with values from -1 to 1 (these are the candidates to be stored). The pointwise multiplication between these vectors decides which elements remain and which elements are set to 0. At this point, the pointwise addition creates the new c_t, made from the elements of c_{t-1} that remained in memory, and from the elements added now.

The output gate decides which values of c_t should also be output of the block. The pointwise multiplication between c_t (formerly pointwise multiplied to have only values between -1 and 1) and the output of the third sigmoid neural network, decides the output h_t.

2.5 Optical Character Recognition (OCR)

Optical Character Recognition is the technology used to convert transcribed, handwritten or printed text documents, such as scanned pages and images taken by a camera or a phone, into a text data, which can be edited and reused. In the subprocess called character recognition, OCR is able to recognise the different alphabets, numbers or other characters on the photo of a text document. Thereafter, the characters from the image are fetched and converted to text sentences for further use. Generally, the process of OCR has three stages:

- Accessing (Scanning) the image document;
- Recognising the text data;
- Saving it into any convenient format or displaying it directly to the user for further use.

2.5.1 History of OCR

In 1870, Carey invented the retina scanner, which is an image transmission system that uses a mosaic of photocells. Later, in 1890, Nipkow invented the sequential scanner, which was a major turning point both for modern television and for reading machines [30].

In 1900, the Russian scientist Tyurin envisioned the first OCR machine as an aid to the blinds, but never managed to develop it.

In 1914, Emanuel Goldberg developed a machine that read characters and converted them into standard telegraph code.

Around the same time, Edmund Fournier d'Albe developed the Optophone, a handheld scanner that, when moved across a printed page, produced tones that corresponded to specific letters or characters.

In the late 1920s and into the 1930s, using an optical code recognition system, Emanuel Goldberg developed the "Statistical Machine" for searching microfilm archives.

The modern version of OCR appeared in the middle of the 1940s, with the development of the digital computers. The principal motivation for the development of OCR systems was the need to manage, in the business world, the enormous amount of paper generated by the expanding technological society, such as bank cheques, commercial forms, government records, credit card imprints and mail. The OCR machines have become commercially available since the middle of the 1950s.

In 1974, Ray Kurzweil started the company Kurzweil Computer Products Inc. and continued the development of omni-font OCR, which could recognise text. Kurzweil decided that the best application of this technology would have been to create a reading machine for the blind people, able to read text to them out loud.

This device required the invention of two technologies: the CCD flatbed scanner and the text-to-speech synthesizer.

In the 2000s, OCR was made available online as a service (WebOCR), in a cloud computing environment and in mobile applications, like real-time translation of foreign language signs on a smartphone.

The OCR used in this project is the open-source OCR Tesseract. Tesseract extracts the text from the image that has as its input. However, in order not to have errors in recognising the text, it is mandatory for the image to go through a series of pre-processing steps (which are analysed in the subsequent chapters). To proceed this way, it is also necessary to use the OpenCV library.

2.6 Tesseract

Tesseract is an open-source OCR engine that was developed in the HP Labs, in Bristol, between 1984 and 1994 [31].

Although originally created as a PhD research project, Tesseract was successively seen as a possible software and/or hardware add-on for HP's line of flatbed scanners. This happened because the commercial OCRs of that time were at their beginning and failed on anything. Tesseract achieved more accuracy than the commercial OCRs after a combined project between HP Labs Bristol and HP's scanner division in Colorado, but it was not a product yet. At the end of 1994, Tesseract was sent to the University of Nevada, Las Vegas for the 1995 Annual Test of OCR Accuracy, where it proved to be better than the commercial OCRs of that time. In late 2005, HP released Tesseract as open source, which is now available at http://code.google.com/p/tesseract-ocr. Thenceforth, new versions of tesseract were developed:

- Version 2.00 brought Unicode (UTF-8) support, six languages, and the ability to train Tesseract;
- Tesseract 3.00 added new languages, including Chinese, Japanese and Korean and introduced a new single-file based system of managing language data;
- Tesseract 3.02 added Bidirectional text support, the ability to recognise multiple languages in a single image and improved layout analysis;
- Tesseract 4 added a LSTM neural network based OCR focused on line recognition, while still supporting the legacy Tesseract OCR engine of Tesseract 3, which works by recognising character patterns. The latest stable version is 4.1.1, which was released on 26th December 2019.

2.6.1 Legacy Tesseract OCR engine

The processing used by the Legacy Tesseract OCR engine to read text follows a traditional step-by-step pipeline, as illustrated in Figure 30 [32].

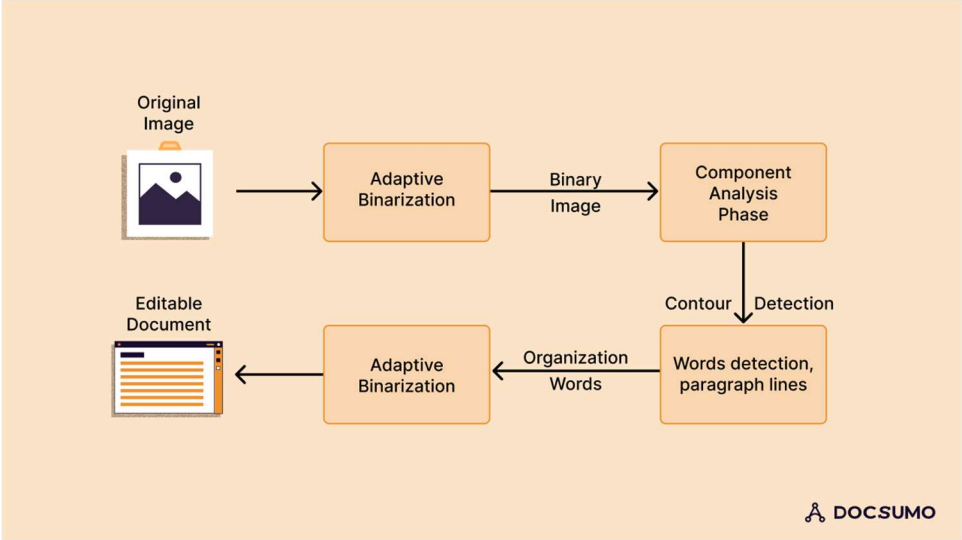


Figure 30 – Legacy Tesseract OCR engine [33].

Tesseract assumes that its input is a binary image with defined text regions.

The first step is a connected component analysis in which outlines of the components are stored.

At this stage, outlines are gathered together into Blobs. Blobs are organised into text lines, and the lines and regions are analysed for fixed pitch (a font whose letters and characters each occupy the same amount of horizontal space) or proportional text (a font where each character occupies only as much width as it needs). Text lines are broken into words differently, according to the kind of character spacing: the fixed pitch text is chopped immediately by character cells, while the proportional text is broken into words using definite spaces and fuzzy spaces.

Recognition, then, proceeds as a two-pass process. In the first step, an attempt is made to recognise each word in turn. Each word that is satisfactory is passed as training data to an adaptive classifier, which then gets a chance to more accurately recognise text lower down the page. Since the adaptive classifier may have learned something useful too late to contribute near the top of the page, a second step is to run over the page, in which words that were not recognised well enough are, now, recognised.

2.6.2 Neural net (LSTM) based OCR engine

From the fourth version Tesseract uses the Bi-LSTM neural network, which consists of two LSTM layers: one processing the input sequence in the forward direction and the other in the backward direction, as illustrated in Figure 31 [34].

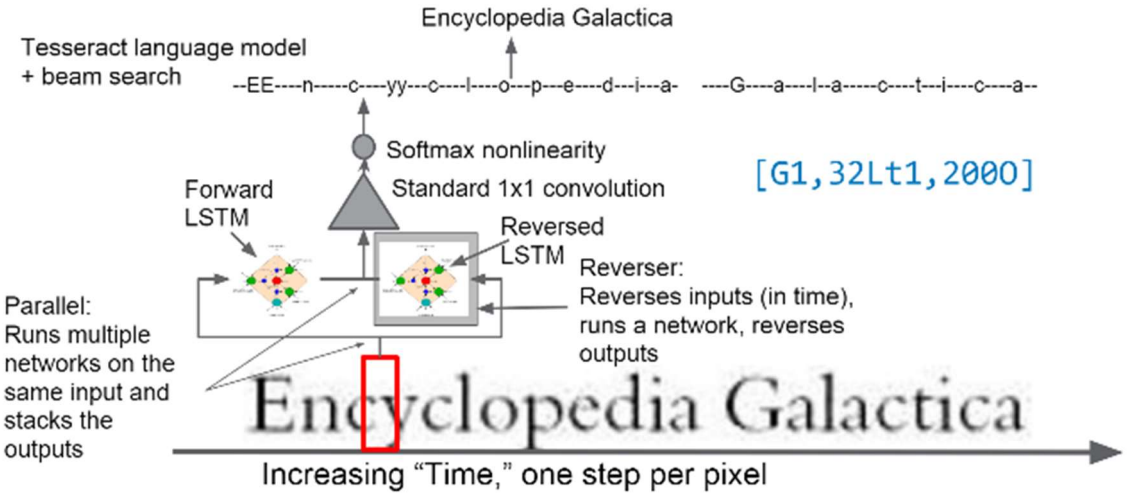


Figure 31 – How Tesseract uses LSTMs [35].

The output of a single LSTM is made up of as many outputs as the number of letters that the network can recognise. The neuron with the highest activation value says what character is read. The output of the Bi-LSTM layer is the combination of the hidden states from both the forward and backward LSTM layers at each time step. One step corresponds to one pixel step.

The outputs of the two LSTMs are concatenated in a stack, which becomes, then, the input of a fully connected network.

At the end, the SoftMax function is applied to the output of this new natural network (Figure 32) to produce a probability distribution, in the sense that the sum of the outputs is equal to 1 [36].

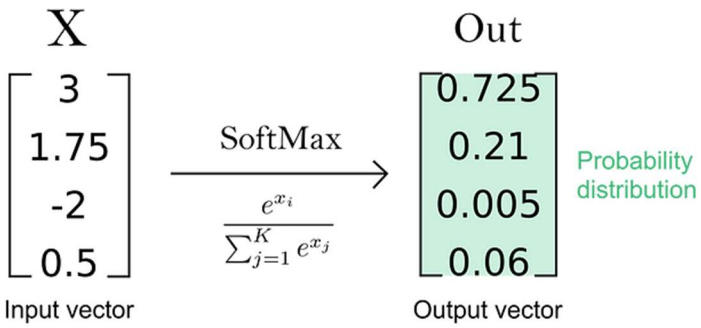


Figure 32 – SoftMax function [36].

By proceeding in this way, the network indicates the probability that the analysed character actually is a certain character.

2.7 Open-Source Computer Vision Library (OpenCV)

OpenCV is an open-source computer vision and machine learning software library aimed at providing a common infrastructure for computer vision applications and at accelerating the use of machine perception in the commercial products [37]. OpenCV is a BSD-licensed product, so it is easy for the businesses to utilise and to modify the code.

The library has more than 2500 optimised algorithms, which can be used to detect and recognise faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high-resolution image of an entire scene. Moreover, these algorithms are able to find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognise scenery and establish markers to overlay it with augmented reality. Among these algorithms there are both classic and state-of-the-art computer vision and machine learning algorithms. The library is used extensively in companies, research groups and by governmental bodies.

2.8 Internet of things (IoT)

The Internet of Things (IoT) is an emerging paradigm that enables the communication between electronic devices and sensors through the Internet. IoT uses smart devices and Internet to provide innovative solutions to various challenges and issues related to various business, governmental and public/private industries across the world [38].

Internet of Things (IoT) is a network that connects uniquely identifiable 'Things' to the Internet, using standard communication protocols. The "Things" have sensing/actuation and potential programmability capabilities. Information about the "Thing" can be collected and their state can be changed from anywhere, anytime, by anything [39].

In 1999 Kevin Ashton used for the first time the term "Internet of Things", when talking about the supply chain management with radio frequency identification RFID-tagged or barcoded items. In the RFID Journal of 22 June 2009, Ashton wrote:

"If we had computers that knew everything there was to know about things – using data they gathered without any help from us – we would be able to track and count everything, and greatly reduce waste, loss and cost. We would know when things needed replacing, repairing or recalling, and whether they were fresh or past their best" [39].

In the same year, in his work "When Things Start to Think", Gershenfeld described the evolution of the World Wide Web as a state in which "things start to use the Net so that people don't need to" [39].

Since not all the communication occurs via the Internet and the communications do not exclusively happen between things, but also between things and people, it would be more correct to use the more general expression "Net of Everything".

This chapter described the state of the art of all the technologies used in this project. It began with a description of the conventional water meters, analysing in detail their main types. Then, AMR meters, AMR smart meters and the M-bus protocol were explored. The focus then shifted to AI, dealing with all its aspects present in this project. Next, the chapter continued with describing OCRs and, in particular, the one used in this project, namely Tesseract. OpenCV was, subsequently, described and finally, there was an overview of IoT. The next chapter is going to deal with the implemented system architecture.

3 ARCHITECTURE OF THE IMPLEMENTED SYSTEM

This chapter deals with the architecture of the implemented system. Figure 33 represents all the components and the communication between them. The chapter, subsequently, explores in detail each hardware component, the protocols and the servers used in this project.

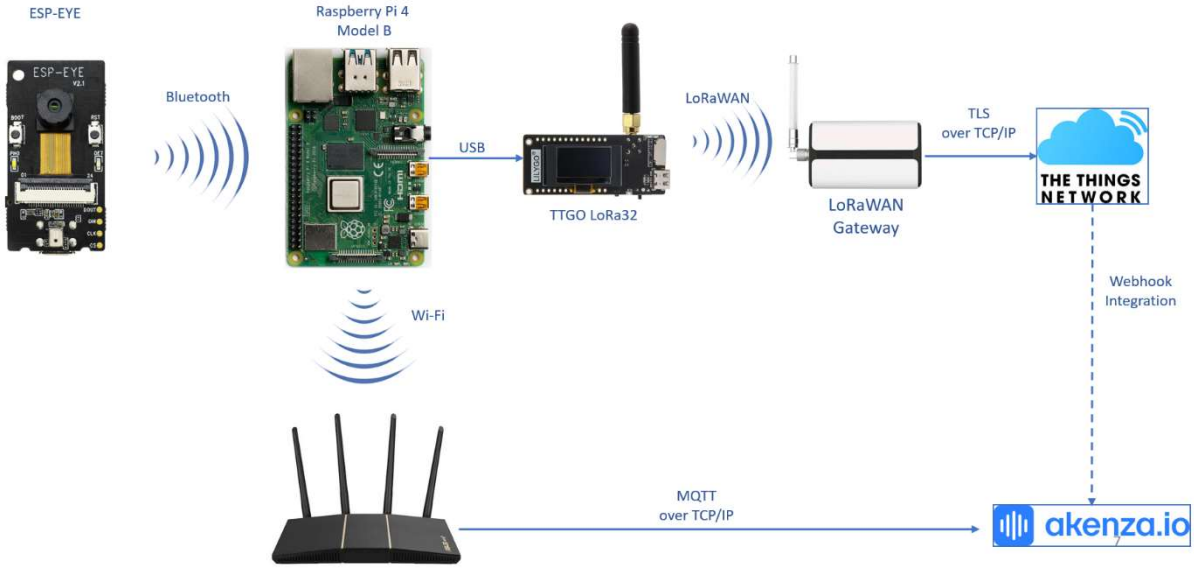


Figure 33 – Architecture of the implemented system.

The following analysis is based on the three-layers IoT architecture, so it is possible to distinguish the three levels: the things or devices layer, the connectivity layer and the application layer.

Figure 34 shows the three-layered architecture of IoT: the things or devices layer, the connectivity layer and the application layer [40].

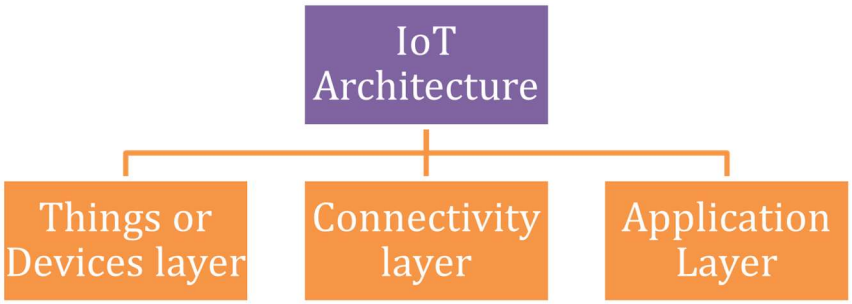


Figure 34 – Three-layered IoT Architecture.

The Things or Devices Layer

This layer is composed of the sensors, the actuators, and the microcontrollers, (collectively referred to as IoT devices) and focuses on gathering the data from the real world. The IoT devices are also called "edge nodes", because they are at the "edge" of the IoT network. Some newer devices already have the access to the Internet, so they are defined IoT-ready, while simpler devices need to be interfaced with a single-board computer, such as an Arduino or a Raspberry, using conventional analog or serial connections [41]. The collected data is 'given' to the connectivity layer for transmission. The sensor present in the system is the ESP-EYE, which is in charge of acquiring images of the water meter and of, successively, sending them to the Raspberry via Bluetooth. Together with the Raspberry, the LoRa32 is part of the edge computing devices and has the task of connecting the entire device to the LoRaWAN network.

The Connectivity Layer

The main function of this layer is the safe transmission of data received from the first layer over long distances. The transmission medium can be wired or wireless. The Connectivity layer includes all the communication protocols used by the system aimed at sending the data to the cloud. The connectivity layer contains all the protocols to create the connection with external servers. The Wi-Fi, MQTT, TCP, IP, LoRa, LoRaWAN, TLS protocols are, therefore, part of this layer.

The Application Layer

The application layer consists of the cloud, which provides management of the servers and databases that allow data storage, big data processing, filtering, analytics, alerts, monitoring and user interfaces. In this system there are two external servers, namely The Things Network (TTN) server and Akenza server. Akenza is used to graphically show the data sent by the developed system. The user can access these values remotely through the Internet from anywhere. Furthermore, the water supply company can also access these values in order to better manage the distribution network.

Figure 33 shows that the data produced by the system can reach Akenza by following two different paths. If the meters are in an area covered by a Wi-Fi network, then the simplest way to send the data to Akenza is using the Wi-Fi to reach the home router and, then, MQTT over TCP/IP to reach the Akenza server. On the contrary, if it is not possible to access a Wi-Fi network, then the best choice is to use the LoRaWAN network to send the data to The Things Network server and then, also in this case, to send the data to Akenza, through a Webhook integration. A third alternative could be to use the Wi-Fi network as the main network and to switch to the LoRaWAN network in case of Internet connection problems.

The following section is dedicated to the description of the ESP-EYE sensor.

3.1 ESPRESSIF ESP-EYE

The ESP-EYE (Figure 35) is an ESP32-based development board designed by ESPRESSIF, that integrates a digital microphone, an 8 Mbytes Pseudo Static RAM (PSRAM) and a 4 Mbytes flash, while also providing an external 2 Megapixels camera [42]. The board can also support image transmission over Wi-Fi and debugging through a Micro USB port. It is used for applications such as face detection, face recognition and speech recognition and for the development of advanced AI solutions.

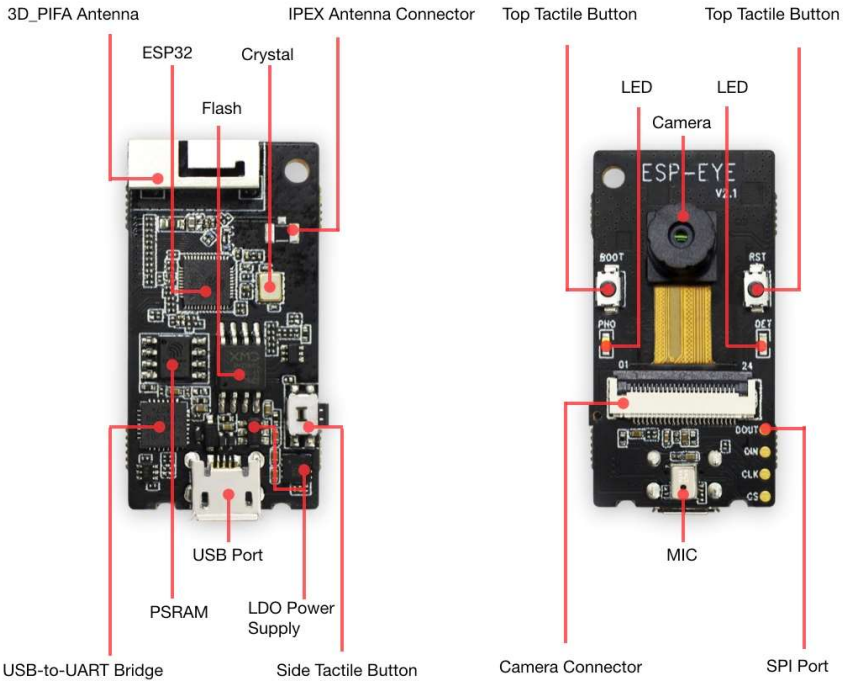


Figure 35 – Key components, interfaces and functions of the ESP-EYE [42].

The core of the ESP-EYE is the ESP32 Chip. It is a 2.4 GHz Wi-Fi and Bluetooth combo chip. The Crystal Oscillator provides an external clock to the ESP32, while the Flash and the PSRAM provide memory for storing applications. A camera connector is used to connect the external camera to the ESP-EYE module. This specific ESP-EYE uses the OV2640, a 2 Megapixels camera module. More specifically, the OV2640 is a low voltage CMOS image sensor that provides the full functionality of a single-chip UXGA (1632x1232) camera and image processor in a small footprint package. The USB port provides power supply to the whole system, and it is also suitable for uploading to the ESP-EYE code written on the computer, using the Arduino IDE. For this purpose, the CP2102 USB-to-UART Chip is

present to convert the USB signals to UART signals. A 3D PIFA antenna is used for both Bluetooth transmission and Wi-Fi transmission. It is also possible to connect an external IPEX antenna via the IPEX connector. Among the other features of the ESP-EYE, there are two tactile buttons, two LEDs (white and red), a digital microphone and, finally, an SPI port.

3.1.1 Arduino IDE

The software used to program both the ESP-EYE and the LoRa32 is the Arduino Integrated Development Environment 2 (IDE 2). Compared to its previous IDE version, this new second version improved the user interface and has new features, including autocompletion and a built-in debugger [43]. Other important features are illustrated in Figure 36.

Figure 36 – Arduino IDE 2.0. Adapted from [43].

Sketchbook is used to manage sketches, which are the code files produced by the Arduino IDE and are in *.ino* format. The Arduino IDE requires these files to be contained within a folder having the same name as the file. To compile and upload a sketch onto a board, the board must be selected via the Boards Manager. If the board is not among those selectable, it can be installed. The same applies to libraries which, to be used within sketches, must firstly be installed via the Library Manager. When the ESP-EYE or the LoRa32 are connected to the computer and execute the *serial.print()* command, the output data will be displayed on the Serial Monitor.

3.2 Edge computing

In the developed system, the edge computing is the task that the Raspberry Pi and the LoRa32 have to accomplish. The Raspberry collects the data from the sensors, stores it in its local database and sends it:

- directly to the Akenza server via Wi-Fi and Internet or

- to the LoRa32, through the USB connection, for the transmission through LoRaWAN.

Edge computing results in lower latency, because computing takes place at or near the physical location of the data source. This implies a faster possibility of action because the data produced by IoT sensors and devices can be quickly analysed, without the need to wait for the data to travel to a central server. Moreover, by using edge computing, some raw data is already analysed and only the resulting data is put in the network, thus saving bandwidth.

3.2.1 Raspberry Pi 4 Model B

The core of the developed system is the Raspberry Pi 4 Model B, represented in Figure 37.

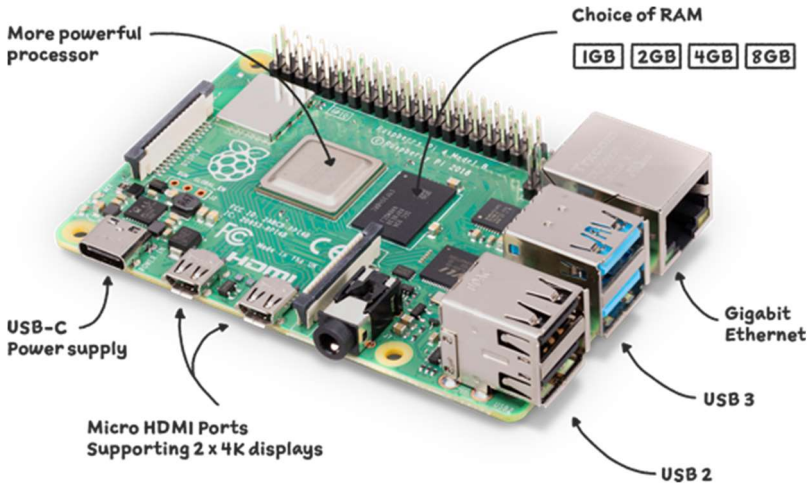


Figure 37 – Raspberry Pi 4 Model B [44].

The Raspberry Pi 4 Model B is a credit card-sized computer, and one of the latest products in the Raspberry Pi range of computers. This model will remain in production until at least January 2026 [45].

The speed and performance of the Raspberry Pi 4 is a step up from earlier models and it allows for a complete desktop experience, but on a smaller, more energy-efficient and much more cost-effective machine [44]. It is fanless, so it runs silently, and uses far less power than other computers. The Raspberry Pi 4 is equipped with a Gigabit Ethernet, along with Wi-Fi and Bluetooth. It also features two USB 2 ports, two USB 3 ports and a Micro SD card slot for loading the operating system and for data storage. The Raspberry is powered by 5 V DC through the USB-C connector or by 5 V DC through the GPIO header.

Table 3 shows the main features of all the Raspberry models produced up to the model used in this project.

Table 3 – Comparison of Raspberry Pi Models [46].

Model	SoC	Memory	GPIO (n. of pins)
Raspberry Pi Model B	BCM2835	256MB, 512MB	26
Raspberry Pi Model A	BCM2835	256MB	26
Raspberry Pi Model B+	BCM2835	512MB	40
Raspberry Pi Model A+	BCM2835	256MB, 512MB	40
Raspberry Pi 2 Model B	BCM2836	1GB	40
Raspberry Pi 3 Model B	BCM2837	1GB	40
Raspberry Pi 3 Model B+	BCM2837b0	1GB	40
Raspberry Pi 3 Model A+	BCM2837b0	512 MB	40
Raspberry Pi 4 Model B	BCM2711	1GB, 2GB, 4GB, 8GB	40

A very important feature of the Raspberry Pi is the presence of a row of 40 GPIO (general-purpose input/output) pins, shown in Figure 38 [46].

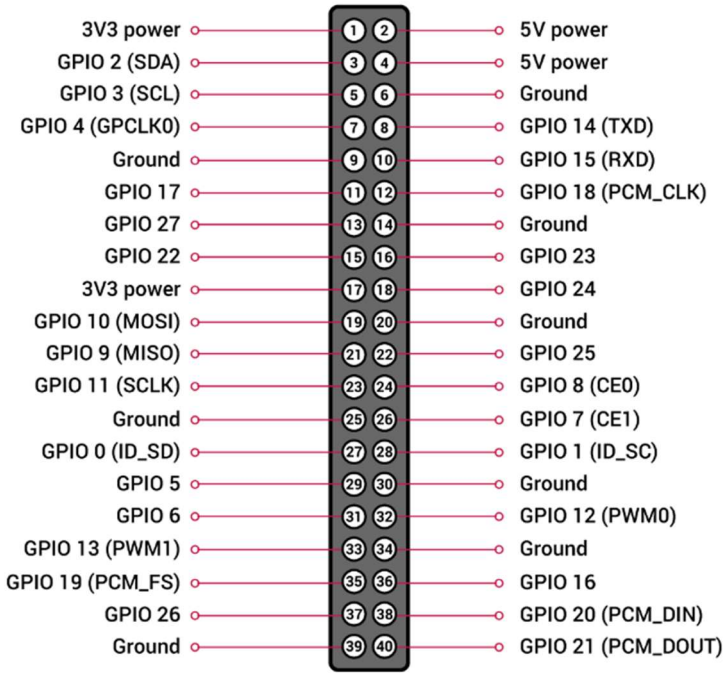


Figure 38 – Raspberry Pins. Adapted from [46].

The 5 V, 3.3 V and ground pins are unconfigurable, while all the other pins are configurable via software as input or output. A GPIO pin designated as an output pin can be set to high (3V3) or low (0V), while a GPIO pin designated as an input pin can be read as high (3V3) or low (0V). Some specific functions, such as PWM, SPI, I^2C and UART, require the use of certain specific GPIO pins, while for other functions it is possible to use all the pins indifferently.

3.2.2 LILYGO® TTGO LoRa32

After an analysis of the devices available on the market, the LILYGO TTGO LoRa32 T3 model, version 2.1 revision 1.6 was selected to add the possibility of transmitting data via LoRaWAN [47]. Figure 39 shows the two models currently manufactured by LILYGO.

Function \ Version	LoRa32 V2.1		T3S3 V1.0		
Product Image					
MCU	ESP32		ESP32-S3-FH4R2		
LoRa Module	SX1278	SX1276	SX1280	SX1276	SX1262
Version Optional	① 433MHz	① 868MHz ② 915MHz ③ 923MHz	① 2.4G	① 868MHz ② 915MHz	
LoRa Antenna Gain	2.0 dbi				
LoRa Antenna Base	SMA Holder				

Figure 39 – Two versions of TTGO LILYGO. Adapted from [48]

Figure 40 is a rear and frontal view of the actual board used in this project and it is indicated as T3_V1.6.

Figure 40 – Rear and frontal view of the LoRa32 T3 model.

The core of this device is the ESP32-PICO-D4, a System-in-Package (SiP) whose main component is the SoC ESP32. The ESP32 is a low-cost, low-power system on a chip (SoC) [49] equipped with Wi-Fi and Bluetooth capabilities, created by Espressif Systems. The ESP32 has a dual-core or single-core Tensilica Xtensa LX6 microprocessor with a clock rate of up to 240 MHz.

The LoRa32 presents a LoRa module, which is connected to the ESP32 through SPI communication with the pins IO18(SS), IO5(SCLK), IO27(MOSI), IO19(MISO). A 0.96 inches OLED display is connected to the ESP32 via I^2C (GPIO21 as SDA and GPIO22 as SCL). Finally, there is an antenna for both Wi-Fi and Bluetooth, which is already assembled on the module and an antenna for LoRa, which can be screwed on the module through a SMA connector.

The LoRa32 is connected to the Raspberry via USB. This means that, in order to use it in the Raspberry software, it is firstly necessary to uniquely identify it. In the command prompt of the Raspberry, the line *ls /dev/tty** was entered with the aim of listing all the connected devices, while the LoRa32 was not connected to the Raspberry. Figure 41 illustrates what was obtained.

Figure 41 – *ls /dev/tty** while the LoRa32 was not connected to the Raspberry.

Subsequently, the LoRa32 was connected via USB to the Raspberry and the line *ls /dev/tty** was entered again, now producing the result in Figure 42.

Figure 42 – *ls /dev/tty** while the LoRa32 was connected to the Raspberry.

The device ttyUSB0 appeared now on the list, thus identifying the LoRa32.

3.3 Bluetooth

Bluetooth is a new RF short-range wireless technology designed for wireless communication between different devices [50]. Bluetooth communication uses Frequency Hopping Spread Spectrum (FHSS) to spread energy across the ISM

spectrum in 79 hops shifted by 1 MHz, starting at 2.402 GHz and stopping at 2.480 GHz. The innovation of Bluetooth is the use of the convention characterised by the IEEE 802.15 standard. The modulation technique is a binary Gaussian frequency shift-keying (GFSK) and the baud rate is 1 Msymbol/s. Figure 43 illustrates the Bluetooth protocol stack.

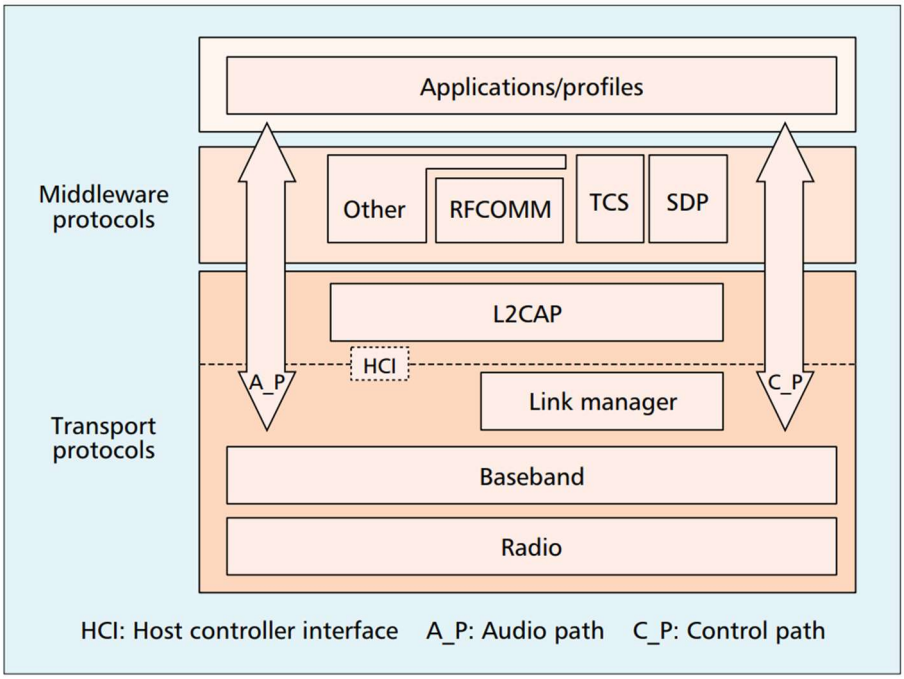


Figure 43 – Bluetooth protocol stack [51].

The protocols in the stack have been grouped in two categories [51]: the transport and the middleware protocols. The radio layer defines the technical characteristics of the Bluetooth radios. The baseband defines the key procedures that allow devices to communicate through Bluetooth wireless technology, as well as Bluetooth piconets and how they are created. A piconet is a collection of Bluetooth devices that can communicate with each other and lasts as long as its creator is available to communicate with other devices. A piconet contains at least one device, identified as the piconet master and a maximum of seven other devices, identified as slaves. Each Bluetooth device has a unique IEEE type 48-bit address assigned at the time of manufacture. The Bluetooth Device Address (BD_ADDR) is engraved on the Bluetooth hardware and cannot be changed. The Logical Link Control & Adaptation Protocol (L2CAP) layer shields the specifics of the lower Bluetooth layers and provides a packet interface to the higher layers. The RFCOMM layer emulates the signals on the nine wires of an RS-232 connection cable, thus enabling serial transmission via Bluetooth. RFCOMM allows legacy applications, written to operate over serial cables, to run over a Bluetooth connection without modification. Several applications developed for Bluetooth use the RFCOMM as part of their implementation stack.

3.4 Universal Asynchronous Receiver-Transmitter (UART) over Universal Serial Bus (USB)

The communication between the Raspberry and the LoRa32 utilises the Universal Asynchronous Receiver-Transmitter (UART) over USB protocol, so this section is going to describe the USB cable and connectors and the UART communication protocol.

3.4.1 USB Cable and Connectors

Figure 44 represents the main existing USB connectors [52].

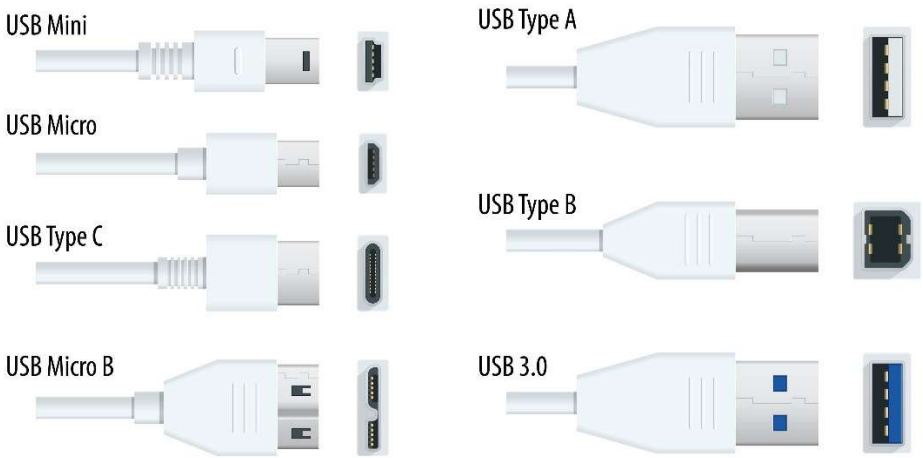


Figure 44 – USB connectors [53].

The most innovative USB connector is the type C, which allows to recharge devices that require more than 25 W. Moreover, it can be reversed, thanks to its symmetry. However, the LoRa32 only has one USB 2.0 type Micro-B port, which is going to be analysed in detail in this subsection. Figure 45 illustrates an image of a USB 2.0 cable, of its standard colours and of its isolation.

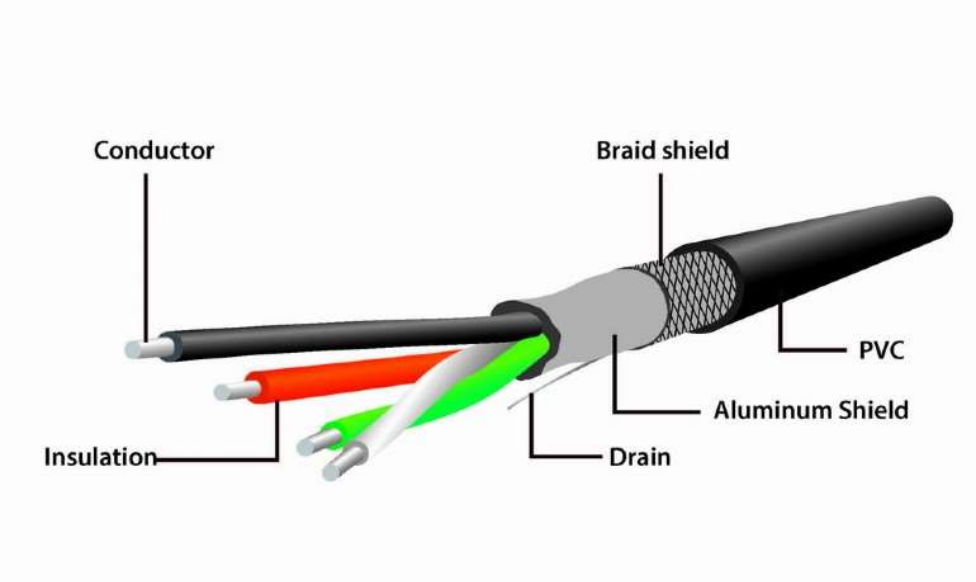


Figure 45 – USB 2.0 cable [54].

The USB 2.0 cable utilises four isolated conductors, two of whom are destined to the power supply (red and black), while the other two are used for the transmission of data (white and green). The red cable is the 5 V power supply and the black cable is the ground, while the green cable is DATA+ and the white cable is DATA-. For the purpose of limiting the unwanted external electromagnetic interference, the two conductors for the data transmission form a pair of twisted conductors, so that the voltage that is induced in each of the two wires is the same. This means that, by detecting at the receiver, the differential signal between the two conductors, the external electromagnetic interference can be cancelled.

3.4.2 Universal Asynchronous Receiver-Transmitter (UART)

UART is a serial communication protocol for sending serial data via TX/RX pins [55]. Normally, the TX pin of one device is connected to the RX pin of the other device and vice versa. However, in the communication between the Raspberry and the LoRa32, the protocol used is UART over USB so, in this project, DATA+ and DATA- are used as connection wires between the TX and the RX pins.

UART is a serial protocol and it operates asynchronous, so it uses predefined baud rates, which is the number of bits transmitted in one second (bps), to determine the speed at which the data is transmitted over the communication channel. For the communication to work, both the transmitting and receiving devices have to use the same baud rate. Figure 46 illustrates the UART frame format.

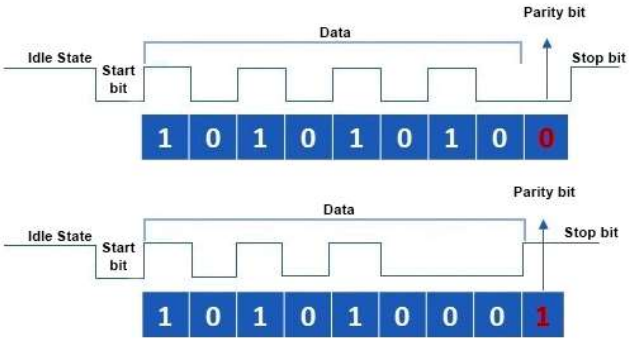


Figure 46 – UART frame format. Adapted from [56].

The TX idle state is the logic 1 and the communication follows the following order:

- The TX line is switched from logic 1 to logic 0 to initiate the communication. This is called start bit.
- Next, the actual data is sent one bit at a time. A common configuration for the actual data is 8 bits, so in this way there are 2^8=256 different combinations, allowing the sending of one out of the 256 ASCII characters.
- Thereafter, there is the parity bit.
- Finally, the frame ends with the stop bit.

Parity bit

The feature "parity" of the frames is an error-checking mechanism, to verify the integrity of the received data. It can be set to "odd" or "even". For example, in Figure 46 the parity feature is set to "even" [56], meaning that the number of bits whose value is 1, in both the actual data and the parity bit, must be even. In the first example of Figure 46, there are four 1s in the actual data. This means that in order for the number of 1s to be even, then the parity bit must be 0. On the other hand, in the second example, there are only three 1s in the actual data. In this case the number of 1s becomes even if the parity bit is 1. If the number of 1s does not match the expected parity, it means that there is an error.

3.5 ISO OSI Model and TCP/IP Model

The Open Systems Interconnection (OSI) is a conceptual model, developed by the International Standards Organization (ISO) to define a common basis that all the standards of communication in open systems have to follow. "Open" means that the system is open to communicate with other systems. The OSI model consists of seven layers, namely: Physical, Data Link, Network, Transport, Session, Presentation and Application, while the TCP/IP model has four layers: Network Access, Internet, Transport and Application. Figure 47 illustrates the ISO OSI model on the left and the TCP/IP model on the right [57].

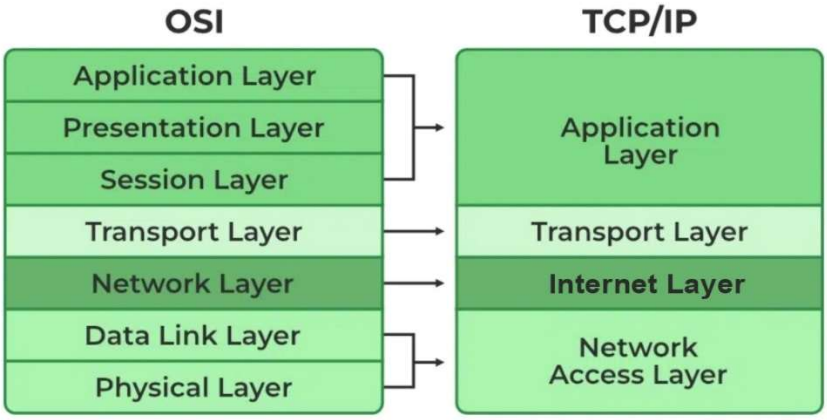


Figure 47 – OSI and TCP/IP Models. Adapted from [58].

A description of all the levels is presented below.

The application layer consists of all the applications and the programs that allow users to send each other files through a network.

Under the application layer there is the transport layer, where there is one out of two possible transmission protocols:

- The Transmission Control Protocol (TCP), which divides the data and encapsulates it into segments. Since TCP provides the retransmission of lost data segments, it guarantees that all the segments will be delivered from the source to the destination. Moreover, it provides error checking and control flow. TCP is the most appropriate protocol when the integrity of the data is more important than the data rate (e.g., images, data files and web pages). In this project the protocol under MQTT is TCP, as the transmission of the correct data is more important.

- The User Datagram Protocol (UDP) divides the data and encapsulates it into datagrams. UDP is preferred when the data rate is more important than the integrity of the data (streaming videos).

The next layer in the TCP/IP model is the Internet Layer, which corresponds to the Network layer of the ISO OSI model. This layer divides and encapsulates the information into packets, which will, subsequently, be sent to the destination, by using the IP addresses. The routers, through the routing algorithm and the routing tables [57] select the optimal path to ensure the transmission of the packets between the source and the destination.

Next, there is the Data Link layer (ISO OSI model) that is composed of two sublayers:

- The Logical Link Control (LLC). This sublayer hides from the network layer the differences that exist between the different IEEE 802 family protocols, so that the network layer can operate in the same way for all the transmission means of communication. It divides the received packets and encapsulates them with its layer information into frames of the LLC layer. Then, the data goes from the LLC sublayer to the MAC sublayer.
- The Media Access Control (MAC) has access to the medium and encapsulates the LLC frames into MAC frames, appropriate for the specific transmission medium. In this layer, the source and the destination, which are in the same network, are identified by their MAC address.

Finally, there is the Physical layer (ISO OSI model), which takes care of the transmission of bits on the communication channel between network nodes. The communication channel can be a specific physical cable or a wireless connection. In the TCP/IP model the two layers Data Link and Physical correspond to one layer called Network Access layer.

3.6 Wi-Fi

Wi-Fi is a short-range wireless access technology that converts signals from wired to wireless based on the IEEE 802.11 protocols [59]. It has become the preferred way for users to access the Internet both in the home and at work. The Wi-Fi Alliance (WFA) forecasts that by 2022, nearly 400 million of Wi-Fi devices will be serving the world, carrying more than half of the global data traffic. In 1997, the 802.11b version was the first widely accepted Wi-Fi standard. From the first version to the 802.11ax version, which was introduced in 2019, the 802.11 family of Wi-Fi standards has gone through six generations as summarised in Figure 48.

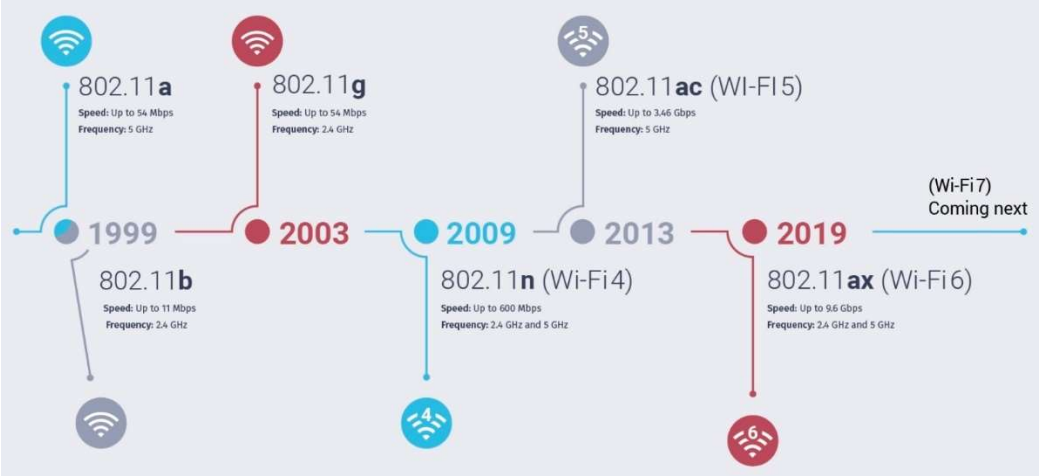


Figure 48 – Wi-Fi Evolution Timeline. Adapted from [60].

In October 2018, the WFA decided to name Wi-Fi generations in "Wi-Fi + number" format, so the currently used version is the Wi-Fi 6 (IEEE 802.11ax). The evolution of these Wi-Fi generations mainly revolved around increasing the theoretical rate.

3.7 MQTT

MQTT is a standard messaging protocol of the application layer, used on IoT applications [61]. It was designed for the communication between devices with resource constraints or limited network bandwidth, as in the Internet of Things. Usually, MQTT runs over TCP (Figure 49), because this transport protocol provides ordered, lossless and bidirectional communication.

In 1999, Andy Stanford-Clark and Arlen Nipper developed its first version.

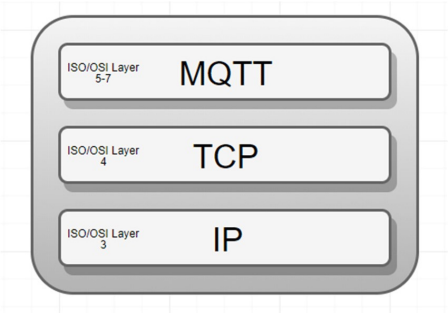


Figure 49 – MQTT on TCP/IP [62].

MQTT has a publish–subscribe software architecture, where the publishers categorise the published messages into topics, instead of sending the messages directly to the subscribers. The subscribers subscribe to one or more topics in order to receive a copy of all the messages that are in those specific topics.

The topics are logical channels named with strings separated by forward slashes, which indicate a topic level. The name of the topic is case-sensitive.

Moreover, between the publishers and the subscribers there is a software intermediary, called MQTT Broker, as illustrated in Figure 50.

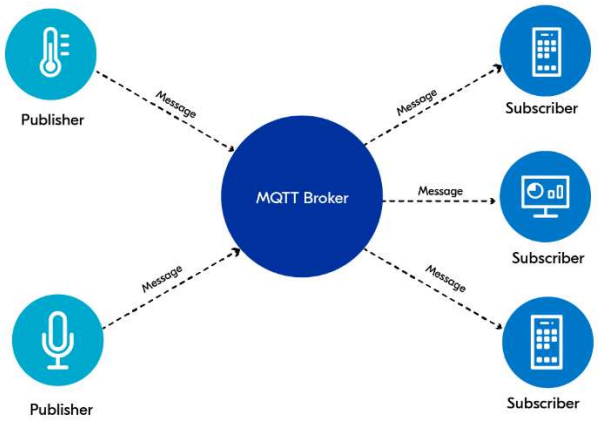


Figure 50 – MQTT communication [63] .

Both the publishers and the subscribers are MQTT clients. MQTT is a bidirectional communication protocol, because each client can both publish messages or subscribe to a topic to receive messages. Publishers post messages in a specific topic to the MQTT Broker, which performs a store and forward function on the messages, sending them to all the clients subscribed to that topic.

In this project, the broker is the Akenza MQTT Broker, the Raspberry is the MQTT publisher client and Akenza is the subscriber client, which is subscribed to the topic:

/up/hcvu8mlrcg6o2otd5g10mnp71bs4dvkk/id/3AEB0E59DE79BAEC

3.8 LoRa

LoRa is the de facto wireless platform of Internet of Things [64]. It is able to connect sensors to the Cloud and to enable real-time communication of data and analytics, which can be utilised to enhance efficiency and productivity. LoRa devices enable smart IoT applications that solve some of the biggest challenges faced by the planet: energy management, natural resource reduction, pollution control, and infrastructure efficiency.

LoRa is a wireless low-power wide-area network modulation technique based on the Chirp Spread Spectrum (CSS) technology.

The Chirp is a signal with a constant amplitude and a frequency that varies in the whole bandwidth from one end to the other end, in a certain time [65]. The chirp is a spread spectrum system, because to send a symbol it does not use one single frequency, but the entire bandwidth. Spread Spectrum, in fact, means that the transmitted signal is spread over a frequency band that is much wider than the bandwidth required to transmit the signal, with the aim of providing robustness to noise and to interference [66].

Figure 51 represents an up-chirp (the frequency changes from the lowest to the highest) and a down-chirp (the frequency changes from the highest to the lowest).

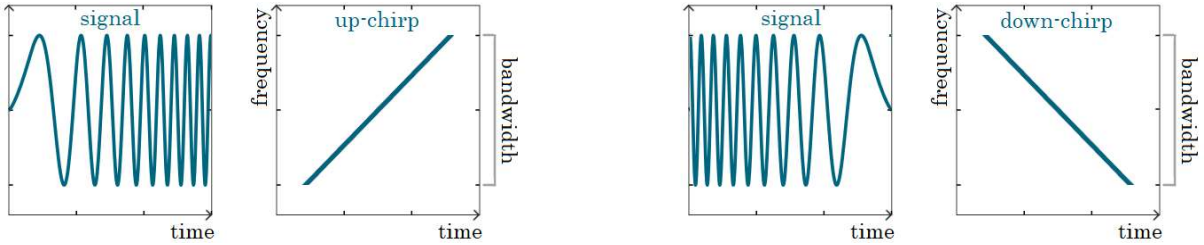


Figure 51 – Linear up-chirp and linear down-chirp [67].

By using LoRa, data can be transmitted over a longer range compared to technologies such as Wi-Fi, Bluetooth, or ZigBee, but at a lower data rate (Figure 52) [68].

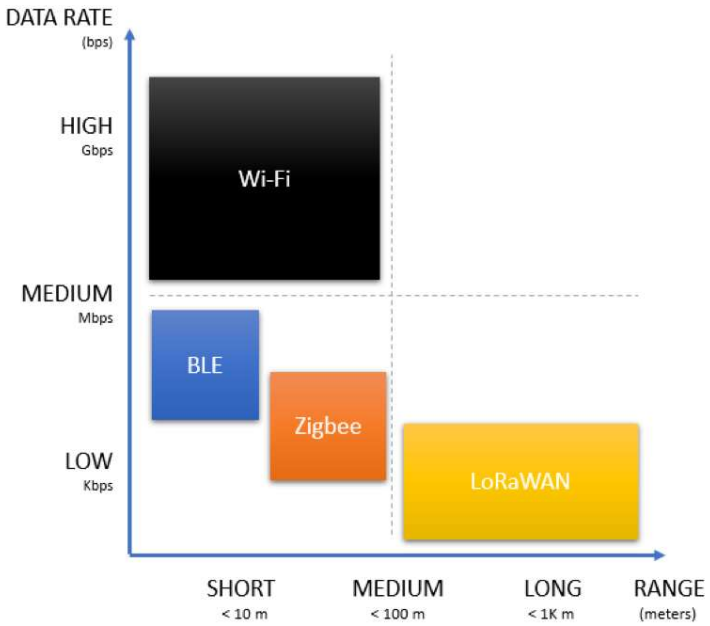


Figure 52 – Distance range versus data rate of various wireless technologies [68].

Analyst firm ABI Research estimates that, by 2026, LoRa will be the leading non-cellular low power wide area (LPWA) network technology and will account for more than half of all non-cellular LPWA connections [69].

LoRa has revolutionised IoT by enabling data communication over a long range while using very little power. LoRa devices and the LoRaWAN standard are flexible for rural or indoor use.

Table 4 shows the LoRa sub-Gigahertz unlicensed bands [70].

Table 4 – LoRa sub-gigahertz unlicensed bands [65].

Area	LoRa frequencies
Europe	EU433 (433.05-434.79MHz) EU863-870 (863-870/873 MHz)
Australia	AU915-928/AS923-1 (915-928 MHz)
North America	US902-928 (902-928 MHz)
India	IN865-867 (865-867 MHz)
Southeast Asia	AU915-928/AS923-1 and EU433

Using LoRa to send two bits at once means that its Spreading Factor (SF) is 2, so there are 4 symbols (see formulas in Table 5). Each symbol is distinguished from the others by its start frequency.

In this case the signal for the symbol 00 begins from the minimum frequency $f_0 = f_{min}$ and reaches the maximum frequency f_{max} at time t_s (Figure 53).

The symbol 01, instead, begins with the frequency f_1 and, after reaching the maximum frequency f_{max}, it begins again from the minimum frequency f_{min} until the frequency f_1. All of this always happens in the time t_s.

Figure 53 – Example of LoRa signal with SF=2.

The parameters of the Chirp Spread Spectrum are summarised in Table 5 [71].

Table 5 – Parameters of LoRa.

Name	Formula
Bandwidth [Hz]	$B = f_{max} - f_{min}$
Spreading Factor [bit/symbol]	SF
Number of symbols [symbols]	$n_{symbols} = 2^{SF}$
Time of duration of each symbol [s/symbol]	$t_s = \frac{2^{SF}}{B}$
Symbol Rate [symbols/s]	$R_s = \frac{1}{t_s} = \frac{B}{2^{SF}}$
Bit Rate [bit/s]	$R_b = \frac{SF}{t_s} = SF \cdot \frac{B}{2^{SF}}$

LoRa can use one of three different bandwidths: 125 kHz, 250 kHz and 500 kHz and the Spreading Factor SF assumes a value from 7 up to 12 [65].

3.8.1 LoRa Message Format

The LoRa Physical Layer message is composed of Preamble, PHDR (Physical Header), PHDR_CRC (Header Cyclic Redundancy Check CRC), PHYPayload and CRC (only in uplink) [72]. Table 6 contains the descriptions of all these fields.

Table 6 – LoRa Physical Layer message.

Name	Description
Preamble	Variable length between 6 and 65535 symbols followed by the synchronization symbols and 2.25 down-chirp symbols 0
PHDR (Physical Header)	Information about payload size and CRC
PHDR_CRC (Header CRC)	Error detecting code for correcting errors in PHDR
Payload	The message (the entire data of the upper MAC layer)
CRC	Check errors in the physical payload

Generally, the CRC can be enabled or disabled in the PHDR. In LoRaWAN, instead, it is present only in uplink.

Figure 54 shows an example of the transmission of a LoRa physical layer message.

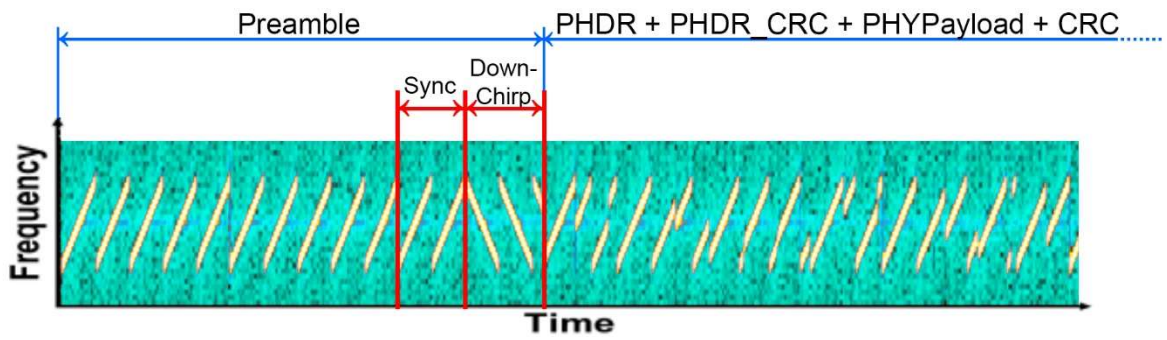


Figure 54 – LoRa Physical layer Message (Frequency versus Time). Adapted from [73].

In Figure 54, the Preamble is composed of ten symbols 0 (which begin from $f_0=f_{min}$), followed by two symbols 0 (Sync) and 2.25 down-chirp symbols 0. The other symbols are for PHDR, PHDR_CRC, PHYPayload and CRC.

3.9 LoRaWAN

LoRaWAN is a Media Access Control (MAC) layer protocol created by the LoRa Alliance in 2015 and it is recognised by the International Telecommunication Union (ITU), the United Nations specialised agency for information and communication technologies (ICTs), as a standard for LPWANs [74]. The LoRaWAN standard meets key Internet of Things (IoT) requirements such as bidirectional communication, end-to-end security, mobility, and geolocation services. In 2017, the LoRa Alliance developed its second version [75].

LoRaWAN is implemented as a software layer built on top of LoRa Physical layer as shown in Figure 55.

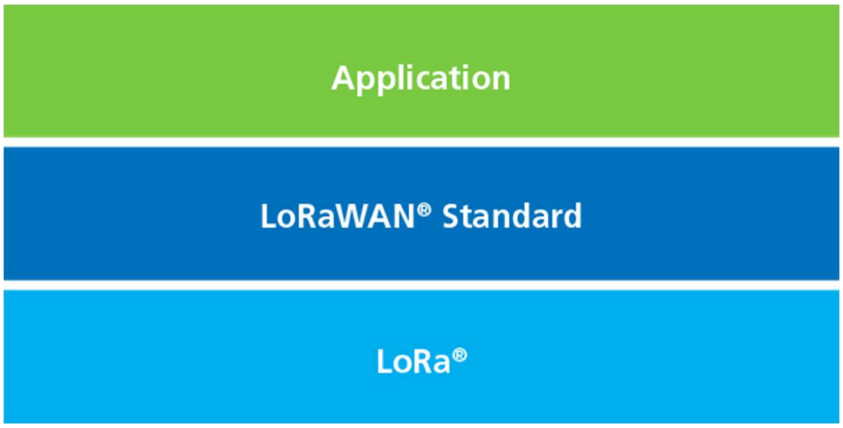


Figure 55 – LoRaWAN model [74].

It is suitable for sending small amounts of data (few bytes) at low data rates from 300 bps to around 50 kbps.

Since the nodes of a LoRaWAN network only communicate when they have data to send, the networks are asynchronous. Typically, these networks are laid out in a star-of-stars topology (Figure 56), in which gateways relay messages between end-devices and a central Network Server [76].

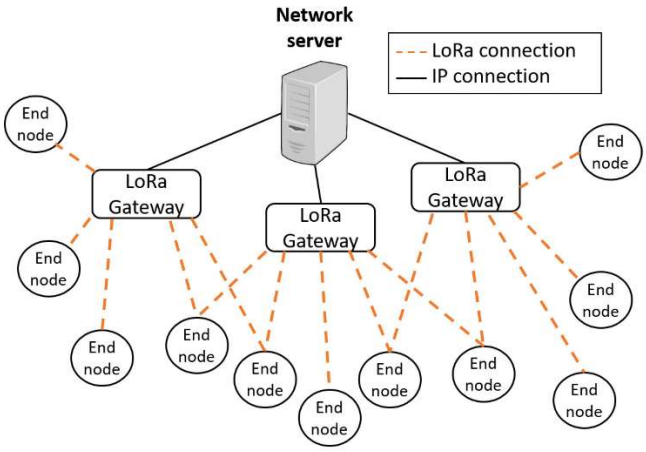


Figure 56 – LoRaWAN Star-of-Stars Topology.

3.9.1 LoRaWAN Message Format

The physical payload of the LoRa message (analysed in Subsection 3.8.1) is the entire data of the upper layer (MAC). The LoRaWAN frame can be one of the three types represented in Figure 57.

MHDR	MACPayload	MIC

MHDR	Join-Request or Rejoin-Request	MIC

MHDR	Join-Accept

Figure 57 – LoRaWAN Frame [77].

All the MAC messages start with a single-octet MAC header (MHDR). Next, there is the MAC payload, which can be an actual data (MACPayload), a Join-request or

Rejoin-request, or a Join Accept. Finally, only for the uplink frames there is a 4-octet Message Integrity Code (MIC).

3.9.2 LoRaWAN Architecture

Figure 58 illustrates an architecture for LoRaWAN.

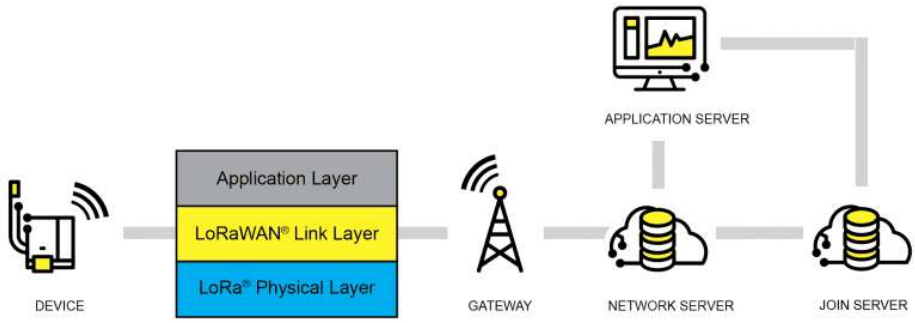


Figure 58 – LoRaWAN Architecture. Adapted from [78].

Network Server, Application Server and Join Server are co-located. The end-device uses the LoRaWAN communication to one or many gateways and the gateways are connected to the Network Server via the Internet connection. The communication is generally bidirectional but, generally, the uplink communication from the end device to the Network Server is the predominant one [79].

End-Device

The data sent by the end-device reaches the Network server via the Gateways. The connection between the Gateways and end-device is wireless. All the application layer payloads of the end-device are routed to its corresponding Application Server.

A device has two addresses associated to it: DevEUI and DevAddr [80].

DevEUI is a 64-bit unique ID, that the manufacturer commonly assigned to the device. This value cannot be altered because it is linked to the hardware, however some more economic devices do not have this address already associated to them.

DevAddr, instead, identifies the end-device within the current network after the device is added to the LoRaWAN network. A DevAddr value is a 32-bit number, consisting of a NwkID, which identifies the specific network, and a NwkAddr, which identifies the end-device within the specific network. Since DevAddr is not unique, multiple devices can have the same DevAddr (not simultaneously).

LoRaWAN Gateway

The LoRaWAN Gateway is a router able to receive LoRa frames, thanks to its LoRa concentrator. It can be mains-powered or can have backup batteries or solar panels and, in order to reach the Network Server, the gateway uses high bandwidth, such as Wi-Fi, Ethernet, cellular or satellite to the Internet. When an end-device transmits a message, all the gateways in that area will receive that message and will forward it to TTN, where the network server de-duplicates the messages. For the downlink communication the best gateway is selected. LoRaWAN operates on unlicensed bands so, in most countries, it is possible to own a personal gateway, while being aware of the eventual country-to-country restrictions on where to install it. The gateway can be indoor or outdoor. In this case, it was not necessary to install a personal gateway, because there are several gateways in the Coimbra city area, as shown in Figure 59.

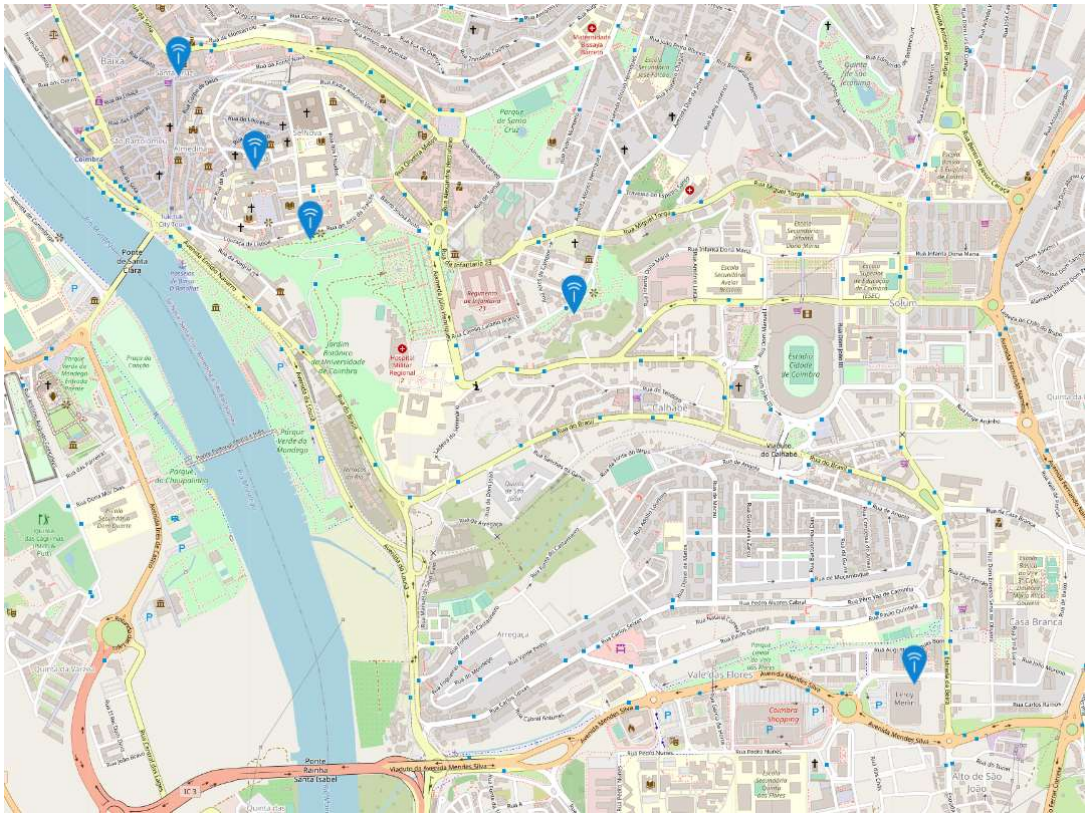


Figure 59 – LoRaWAN Gateways in Coimbra.

Network Server

The Network Server terminates the LoRaWAN MAC layer for the end-device connected to the network, so it responds to all MAC layer requests coming from the end-device. The Network Server filters redundant data coming from more LoRaWAN gateways and chooses the optimal gateway for the downlink communication. It automatically schedules acknowledges through the optimal

gateway and has also the functions of frame authentication and frame counter checks. It adaptively optimises data rates based on the end-device or network conditions. It acts as an intermediary in the communication between end-device and the Application server, forwarding uplink application payloads to the Application server and receiving downlink payloads from it. Finally, it forwards Join-request and Join-accept messages between the end-device and the Join Server.

Join Server

The Join Server ensures the storage of the device's root keys and the associated key derivation operations. The LoRaWAN protocol uses symmetric cryptography session keys derived from the device's root keys. The Join Server also communicates the Network Session Key of the end-device to the Network Server, and the Application Session Key to the corresponding Application Server. Finally, the Join Server manages the activation process of the device in TTN, which can be done in two possible modes: the Activation By Personalization (ABP) and the Over-The-Air Activation (OTAA). In ABP, a fixed DevAddr is hardcoded into the end-device, while in OTAA a dynamic DevAddr is assigned to the end-device in that specific network.

The OTAA activation method is the one used in this project. In order to get authorised to send uplink data and to receive downlink data, the end-device uses the OTAA activation procedure, represented in Figure 60 [80].

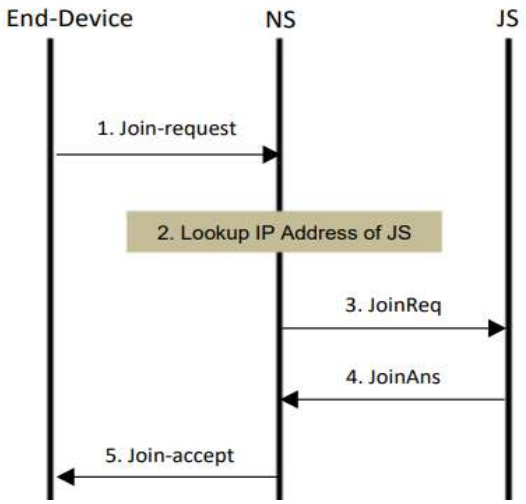


Figure 60 – Over-the-Air Activation (OTAA). Adapted from [80].

Five are the steps that the end-device has to follow to make a join procedure in the LoRaWAN network. The end-device transmits a Join-request message to the Network Server (1); the Network Server uses the DNS to lookup the IP address of the Join Server based on the Join-request message (2); the Network Server sends a

JoinReq message to the Join Server (3), carrying the LoRaWAN frame of the Join-request message, the DevEUI and the DevAddr; the Join Server processes the Join-request message and sends JoinAns to the Network Server (4), carrying Result=Success; finally, the Network Server creates and sends to the end-device a LoRaWAN frame with the Join-accept payload.

When a successful Join/Rejoin Procedure is performed, the end-device has a LoRaWAN session with the backend, which terminates only when the end-device performs the Deactivation (Exit) Procedure or when it performs another successful Join/Rejoin Procedure.

Application Server

The Application Server provides the application-level service to the end-user and generates all the application layer downlink payloads towards the end-device.

3.10 The Things Network

The Things Network is an Internet of Things ecosystem, based on The Things Stack, that creates networks using LoRaWAN [81]. The Things Stack comprises the Network Server, the Application Server and the Join Server (Figure 61).

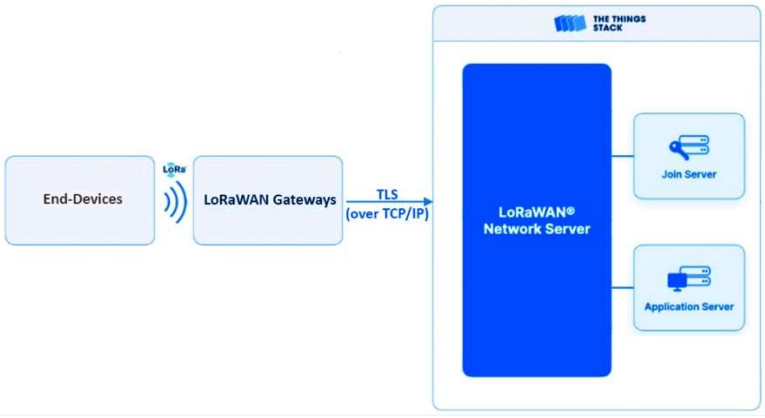


Figure 61 – The Things Stack. Adapted from [81].

The Things Stack was developed by The Things Industries and, since it is currently at version 3, it is also informally known as V3 [81]. The Things Stack also provides a Console, a web application used to register end-devices or gateways, to monitor network traffic or to create an integration. However, since there the console does not graphically display the data, it was necessary to use Akenza for this purpose.

Between TTN and Akenza there is a Webhook integration [82]. Webhooks are automated messages, sent from TTN to a unique URL toward a personal Akenza

account, when TTN receives new data. They are called webhooks: web because they work over the web and hooks because they are software functions that run when something happens.

Since the data sent to TTN has to be, subsequently, sent to Akenza, it was mandatory to send the data from the end-device to TTN already in the format required by Akenza.

As can be seen in Figure 61, the communication between the LoRaWAN gateway and the Network Server uses Transport Layer Security (TLS) on TCP/IP to provide a secure channel for the end-to-end communication [83]. TLS, however, requires that the underlying transport guarantees a reliable and in-order data stream, so the transport layer is TCP. The latter is, in fact, the one that provides the required feature. The TLS secure channel provides confidentiality, because the data sent over the channel is only visible to the endpoints and integrity, as the data sent over the channel cannot be modified by attackers without being detected.

3.11 Akenza

Akenza [84] is an IoT application platform to build IoT services. In order to use Akenza it is necessary to create an Organization and a Workspace. The Organization is a space where a business can collaborate across many workspaces at once, while the Workspace is where all the activities are done.

Akenza controls and manages, in the "Asset" option, the IoT devices. For each device it is necessary to define a Data Flow [85], which indicates the data processing path by using the parameters Device Connector, Device type and Output connectors. Their functions are described in Table 7.

Table 7 – Data Flow parameters [85].

Parameter	Description
Device Connector	The source of the data (MQTT or TTN Integration).
Device type	How the data has to be decoded (The data coming from the TTN Integration is in the Cayenne LPP format)
Output connector	The destination of the Data Flow

Storing the data in the AkenzaDB allows to have access to historical data at any time.

The MQTT topic, the MQTT username and the MQTT password are obtained when a MQTT Data Flow is created, and it is necessary to insert them into the code of the specific device which, in this case, is the Raspberry [86].

When creating the integration from TTN to Akenza, it is necessary to take both the appropriate Application ID and the API key with the necessary permissions, from TTN, in order to import on Akenza, the devices that already exist in TTN [87].

The Akenza Dashboard graphically shows the plots of the data stored in the AkenzaDB [88], which can be represented by using different visual components (KPI, Text, Line & Bar Chart, Table, Map and Floorplan).

This chapter described in detail all the hardware, protocols and servers used in this project. After an initial presentation of the architecture of the developed system, there was an analysis of the devices: the ESP-EYE, the Raspberry Pi and the LoRa32. Subsequently, the two technologies enabling the internal communication between the system components were explored, that is to say Bluetooth and UART over USB. The chapter, then, explored the protocols that allow data to reach the servers, namely Wi-Fi, MQTT, LoRa and LoRaWAN. Finally, the two servers used, TTN and Akenza, were analysed. The next chapter is going to deal in detail with the development of the system.

4 IMPLEMENTED SYSTEM

In this chapter there is a concrete analysis on how the sensors are connected in the implemented architecture. Then, there is a description of the initial configuration of the Raspberry and Akenza. Lastly, the software modules on the ESP-EYE, the Raspberry and the LoRa32 are analysed.

4.1 Initial configuration of the Raspberry

Since the Raspberry used in this project was completely new, the initial configuration was necessary. The first step was to install the operating system on the MicroSD card. In this specific case a 32GB MicroSD card was used. The Raspberry Pi Imager is the quickest and easiest way to install the Raspberry Pi OS on a MicroSD card. The Raspberry Pi OS (previously called Raspbian) is the recommended operating system. This operating system was selected and installed by using the aforementioned Raspberry Pi Imager, as shown in Figure 62.

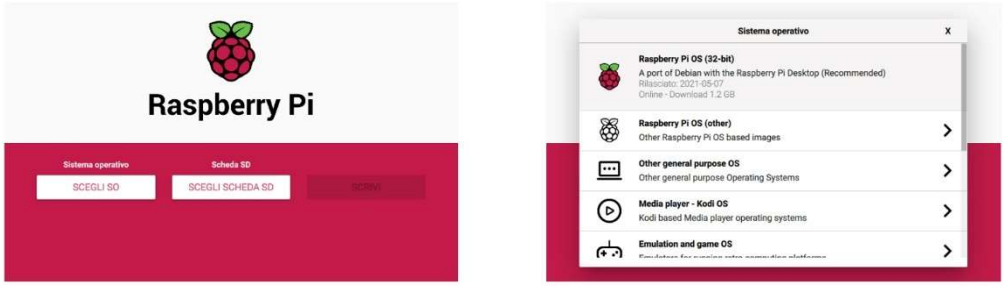


Figure 62 – Choosing the Raspberry Pi OS on the Raspberry Pi Imager.

After having installed the operating system, the next step comprises the insertion of the MicroSD card in the Raspberry; thereafter everything was ready for use. The simplest way to utilise the Raspberry is by connecting a monitor, a keyboard and a mouse to it. However, in this project, it was decided to use it remotely through a computer. Therefore, the procedure was as follows: to write an empty text file named "ssh" (no file extension) to the root of the directory of the card. When the Raspberry Pi OS sees the "ssh" on its first boot-up, it automatically enables the SSH (Secure Socket Shell) protocol, which allows a computer connected to the same network as the Raspberry to remotely access the Pi command line interface.

The subsequent step was to connect the Raspberry to the Wi-Fi network, which the computer is connected to. To proceed this way, it was necessary to create a text file called *wpa_supplicant.conf* and to place it in the root directory of the MicroSD card. In the file the text shown below was inserted:

```
country=PT
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1

network={
scan_ssid=1
ssid="your_wifi_ssid"
psk="your_wifi_password"
}
```

After having inserted the MicroSD card containing the installed operating system, the ssh file and the *wpa_supplicant.conf* file that was previously created, and after powering the Raspberry through its USB-C port, the Raspberry is turned on and is connected to the Wi-Fi network. At this point, it was necessary to establish an SSH connection. First of all, the software PuTTY Configuration was downloaded (since this software is only available on Windows, this operation was performed on another computer). The PuTTY Configuration is illustrated in Figure 63a.

The host name or the IP address of the specific desired device was inserted in PuTTY. After entering its host name *raspberrypi* and clicking on Open, the Raspberry Command Prompt seen from PuTTY appeared (Figure 63b).

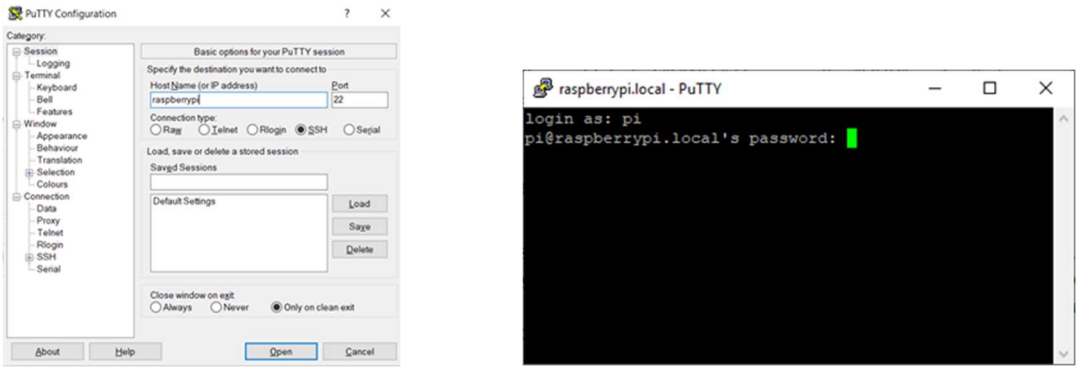


Figure 63 – a) PuTTY Configuration. b) Raspberry Command Prompt seen from PuTTY.

The word *pi* was inserted as the username and *raspberry* as the password to do the first login. All these steps until now were necessary to be able to enter remotely in the command prompt of the Raspberry.

By writing *sudo raspi-config* in the command prompt, it was possible to access the Raspberry Pi Software Configuration Tool. Once having accessed it, in the section interface, the Virtual Network Computing (VNC) interface was enabled.

The VNC interface allows, through the use of the software VNC Viewer, which was installed on the computer, to remotely control the Raspberry through its graphical interface.

Figure 64 shows how to access the graphical interface of the Raspberry using VNC Viewer.

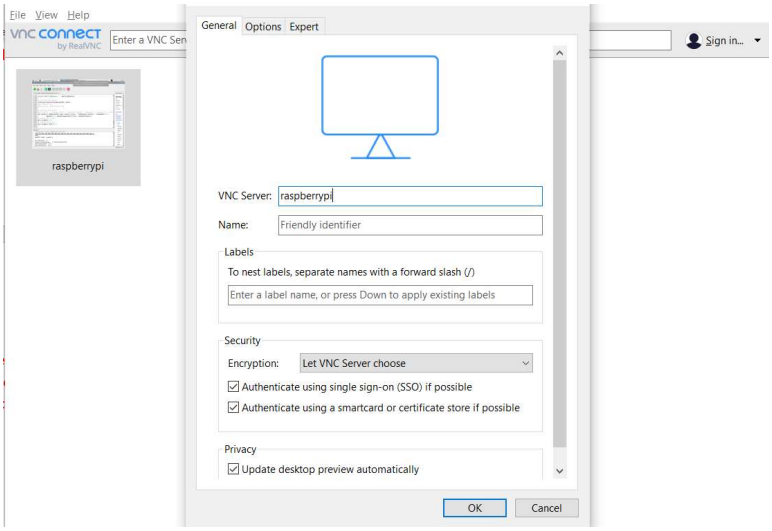


Figure 64 – The VNC Viewer Software.

4.2 Getting started with TTN version 3

This section describes the procedure to configure the version 3 of TTN. The first step was to install the Arduino IDE on the computer. The Arduino IDE is the software used to program the TTGO LoRa32.

Installing the Board TTGO LoRa32 in the Arduino IDE

Firstly, it was necessary to add the Board TTGO LoRa32 to the Arduino IDE, because the boards with processor ESP32 were not already present.

On the Arduino IDE, File → Preferences was selected. On the Additional boards manager URLs field, https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json was inserted (Figure 65a).

After selecting the Boards Manager icon in the left-side corner of the Arduino IDE, the ESP32 was chosen to be installed (Figure 65b).

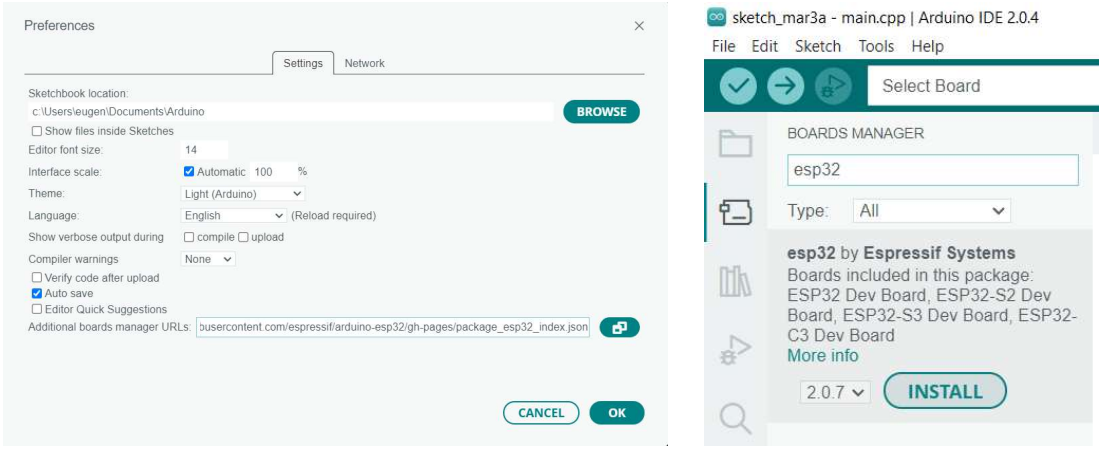


Figure 65 – a) Arduino IDE, File → Preferences. b) Arduino IDE, Boards Manager.

On Select board → Select Other Board and Port, TTGO LoRa32-OLED was chosen (Figure 66).

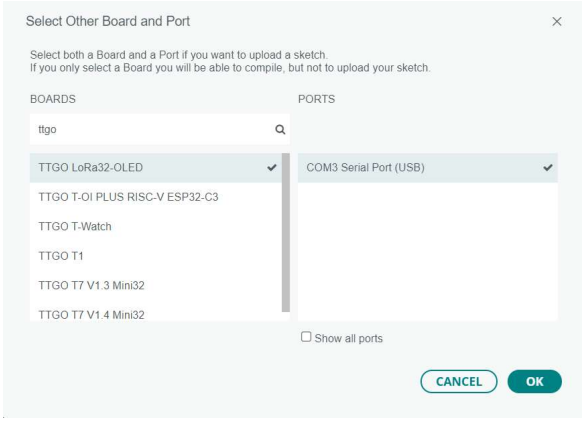


Figure 66 – Arduino IDE, Select board → Select Other Board and Port.

Installing the library mcci lmic

After the installation of the board, it was necessary to search for the correct library mcci lmic and to install it (Figure 67).

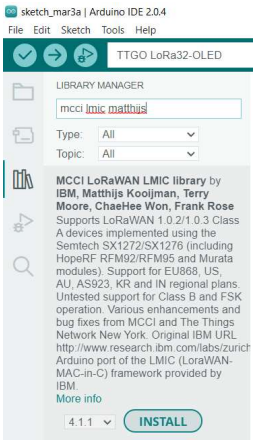


Figure 67 – Arduino IDE, Library Manager.

After installing the library, it was necessary to do some changes on it.

In \Arduino\libraries\MCCI_LoRaWAN_LMIC_library\project_config in the file *lmic_project_config.h*, the European setting was chosen and the following line *#define hal_init LMICHAL_init* was added, as shown in Figure 68.

```
// project-specific definitions
#define CFG_eu868 1
//#define CFG_us915 1
//#define CFG_au915 1
//#define CFG_as923 1
// #define LMIC_COUNTRY_CODE LMIC_COUNTRY_CODE_JP      /* for as923-JP; also define CFG_as923 */
//#define CFG_kr920 1
//#define CFG_in866 1
#define CFG_sx1276_radio 1
//#define LMIC_USE_INTERRUPTS

#define hal_init LMICHAL_init
```

Figure 68 – File *lmic_project_config.h*.

Code to send data from the TTGO LoRa32 to TTN

After the Arduino IDE was ready, it was necessary to write the code to send data from the TTGO LoRa32 to TTN. The example code in ZIP format was downloaded from https://github.com/squix78/TTGO-LoRa32-V1.0-TTN-OTAA and then extracted. This example code only contained the instructions to connect the TTGO LoRa32 to TheThingsNetwork over OTAA and to send simple data to TTN.

In the Arduino program, in Sketch → Add File the file \TTGO-LoRa32-V1.0-TTN-OTAA-master\TTGO-LoRa32-V1.0-TTN-OTAA-master\src\main.cpp was opened.

After copying the example code of the *main.cpp* file, and after pasting the code in a new sketch, this new file was saved as "main" (the format of the file is automatically .ino).

In order for the example code to work, firstly it was necessary to create the account on TTN and then to adapt the example code to make it able to send data to the specific TTN account.

Configuring TTN

At this point, on https://console.cloud.thethings.network/ the correct V3 region was selected and an account on TTN was created. Then, it was necessary to create an application on TTN, selecting +Add application (Figure 69).

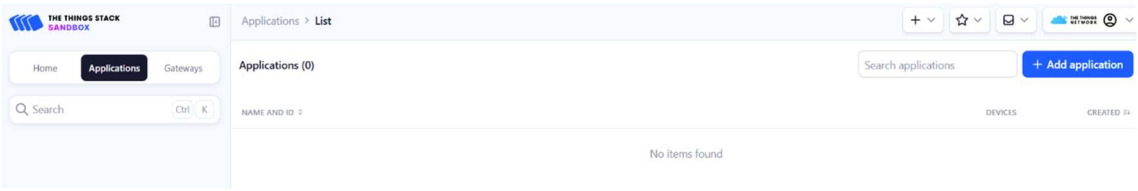


Figure 69 – TTN, Applications.

An Application ID and an Application Name were required before selecting Create application (Figure 70).

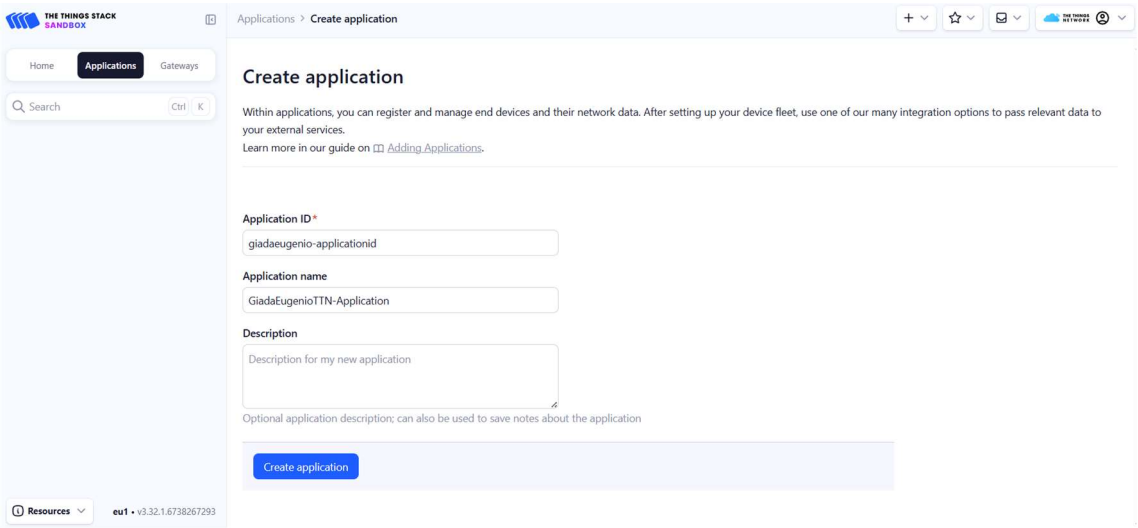


Figure 70 – TTN, Create application.

At this point it was necessary to create the LoRa32 device so, in Application overview, +Register end device was selected (Figure 71).

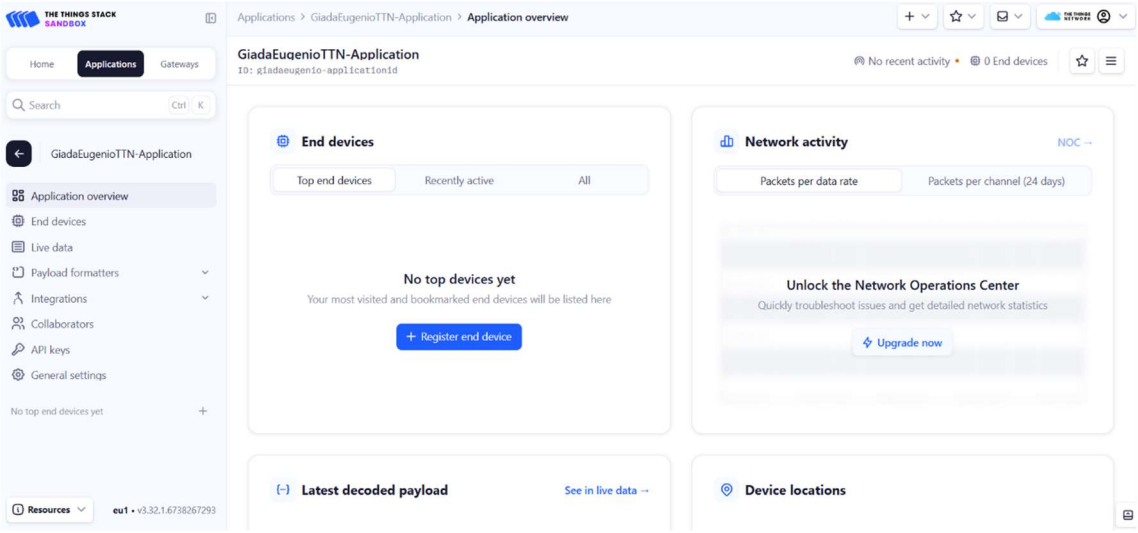


Figure 71 – TTN, Application overview.

DevEUI was already described in Subsection 3.9.2. Since the LoRa32 does not have a DevEUI, it was necessary to generate it on TTN.

The JoinEUI (AppEUI) is a 64-bit globally unique identifier assigned to the LoRaWAN network's Join Server. It identifies the Join Server that will be employed for the join procedure. As it was not already provided by the manufacturer of the LoRa32, it was mandatory to fill it with all zeros.

The AppKey is the encryption key used for messages during every over the air activation. It can be randomly generated by TTN and has to be known by both TTN and the LoRa32.

In order to create the LoRa32 device, it was necessary to choose the European LoRaWAN frequencies (Figure 72), to insert the JoinEUI (Figure 73) and to generate the DevEUI and the AppKey, before selecting Register end device (Figure 74).

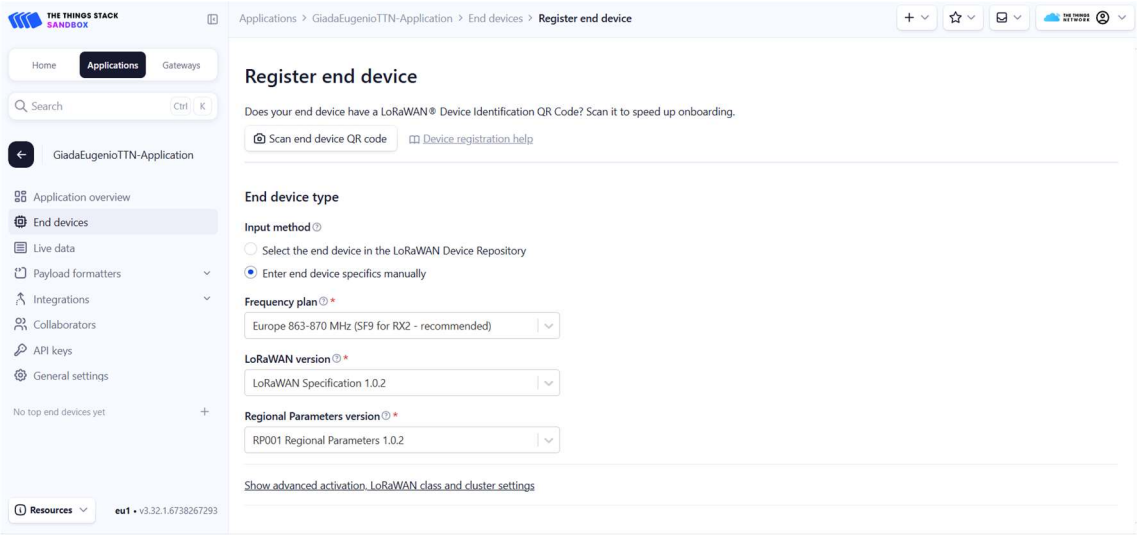


Figure 72 – TTN, Register end device (part 1).

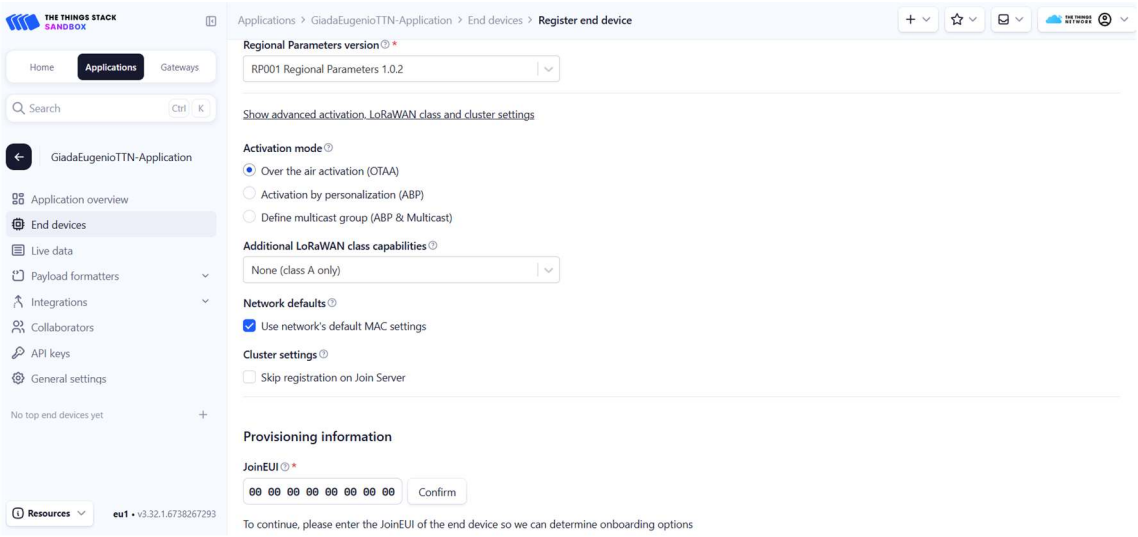


Figure 73 – TTN, Register end device (part 2).

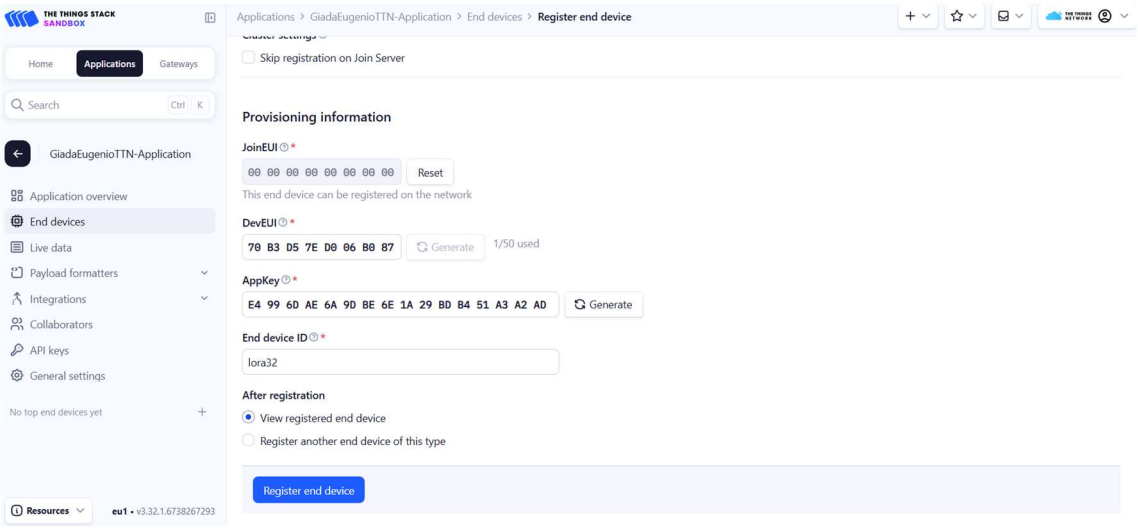


Figure 74 – TTN, Register end device (part 3).

In Device overview, AppEUI and DevEUI were ordered to start with their lsb, while AppKey had to start with its msb, before copying them (Figure 75).

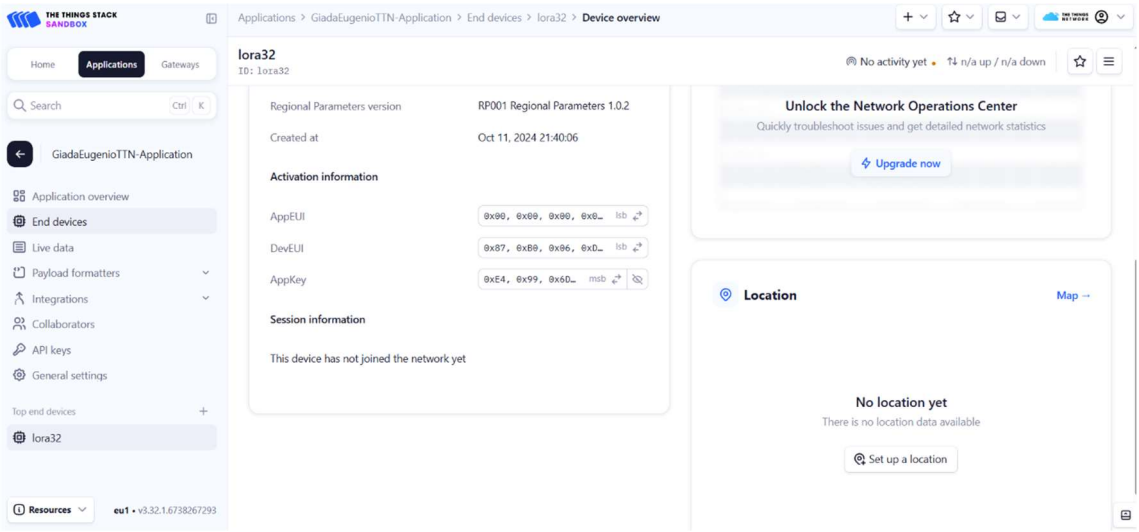


Figure 75 – TTN, LoRa32 Device overview.

Adapting the example code

The personal AppEUI, DevEUI and AppKey obtained on the TTN website were inserted into the example code (Figure 76).

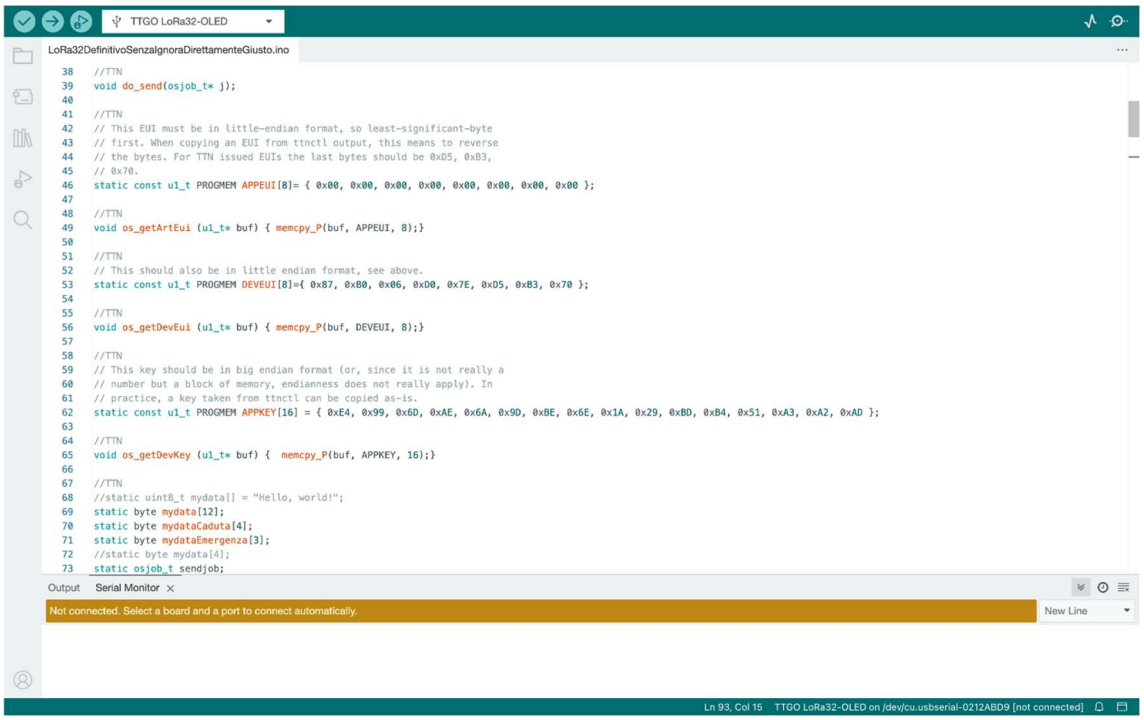


Figure 76 – Arduino IDE, inserting AppEUI, DevEUI and AppKey into the code.

In the adapted code, it was also necessary to write the correct pins of the specific TTGO LoRa32 board (Figure 77).

Figure 77 – Arduino IDE, writing the correct pins of the specific TTGO LoRa32.

Based on the example code, the complete code was written and then compiled and uploaded to the TTGO LoRa32.

4.3 Cayenne LPP Formatter Type

Akenza allows to graphically display the data sent to TTN. Since Akenza accepts the Cayenne Low Power Payload (LPP) standard, it was decided to directly send the data to TTN in this format [89].

The Cayenne Low Power Payload provides a convenient and easy way to send data over LPWAN networks such as LoRaWAN. The Cayenne LPP allows the device to send multiple sensor data at one time. Table 8 illustrates the structure of the Payload.

Table 8 – Structure of the Cayenne Low Power Payload.

1 Byte	**1 Byte**	**N Bytes**	**1 Byte**	**1 Byte**	**M Bytes**	**…**
Data1 Ch.	Data1 Type	Data1	Data2 Ch.	Data2 Type	Data2	…

Data Channel is the channel of the respective widget used on Cayenne. The channels are not perceived by Akenza, so this field is ignored.

Data Type identifies the respective data type that is being sent in the frame, e.g. “temperature”. Table 9 shows all the possible Data Types.

Table 9 – Data Types of the Cayenne LPP.

Type	Hex	Data Size	Data Resolution per bit
Digital Input	0	1	1
Digital Output	1	1	1
Analog Input	2	2	0.01 Signed
Analog Output	3	2	0.01 Signed
Illuminance Sensor	65	2	1 Lux Unsigned MSB
Presence Sensor	66	1	1
Temperature Sensor	67	2	0.1 °C Signed MSB
Humidity Sensor	68	1	0.5 % Unsigned
Accelerometer	71	6	0.001 g Signed MSB per axis
Barometer	73	2	0.1 hPa Unsigned MSB
Gyroscope	86	6	0.01 °/s Signed MSB per axis
GPS Location	88	9	Latitude: 0.0001 ° Signed MSB Longitude: 0.0001 ° Signed MSB Altitude: 0.01 m Signed MSB

It is possible to send data collected by different sensors in the same frame, by specifying the respective channel and type for each of them.

4.3.1 The Cayenne LPP in TTN

It was necessary to indicate on TTN that the type of data that TTN will receive will be in the Cayenne LPP format. Therefore, on Payload formatters → Uplink, Cayenne LPP was chosen as Formatter Type (Figure 78).

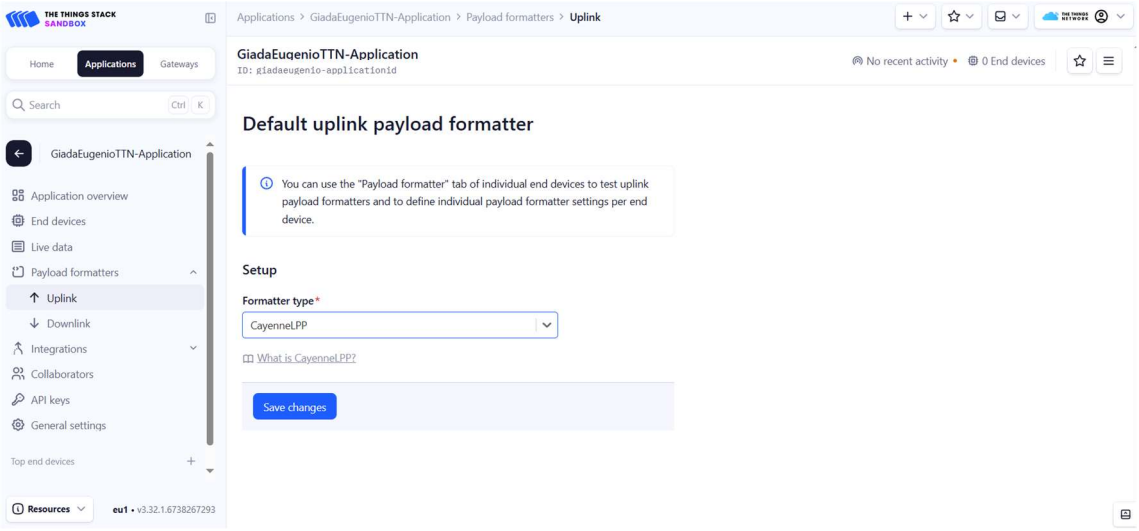


Figure 78 – TTN, setting the Cayenne LPP Formatter type.

This step is needed to visualise also in TTN the data in the correct way. If this step is skipped, for example a temperature value of 27.2 °C sent to channel 3, will be visualized on TTN as "03 67 01 10", because TTN does not understand the real meaning behind those numbers. This example is explained in Table 10.

Table 10 – Example of the Cayenne LPP format.

Payload (Hex)	03 67 01 10	
Data Channel	**Type**	**Value**
03 ⇒ 3	67 ⇒ Temperature	0110=272 ⇒ 27.2 °C

By specifying, instead, to TTN that a payload is being sent in the Cayenne LPP format, it is possible to visualise on TTN the channel, type and the value of the data.

4.4 Getting started with Akenza

This section describes the procedure that was carried out to configure Akenza.

After creating an account on the Akenza website and after doing the login, Start the setup was selected (Figure 79).

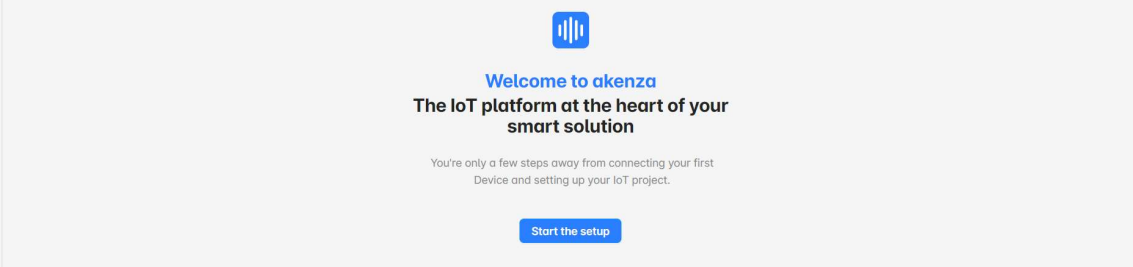


Figure 79 – Akenza, welcome page.

The first step to use Akenza, was to create an Organization, a space where a business can collaborate across many workspaces at once. The name of the organization was inserted before selecting Create Organization (Figure 80a).

The next step was to create a Workspace, which is the space where the actual work happens. All the activities of Adding Data Flows, Rules or managing devices are, in fact, done within a workspace. A Workspace name was inserted before selecting Create Workspace (Figure 80b).

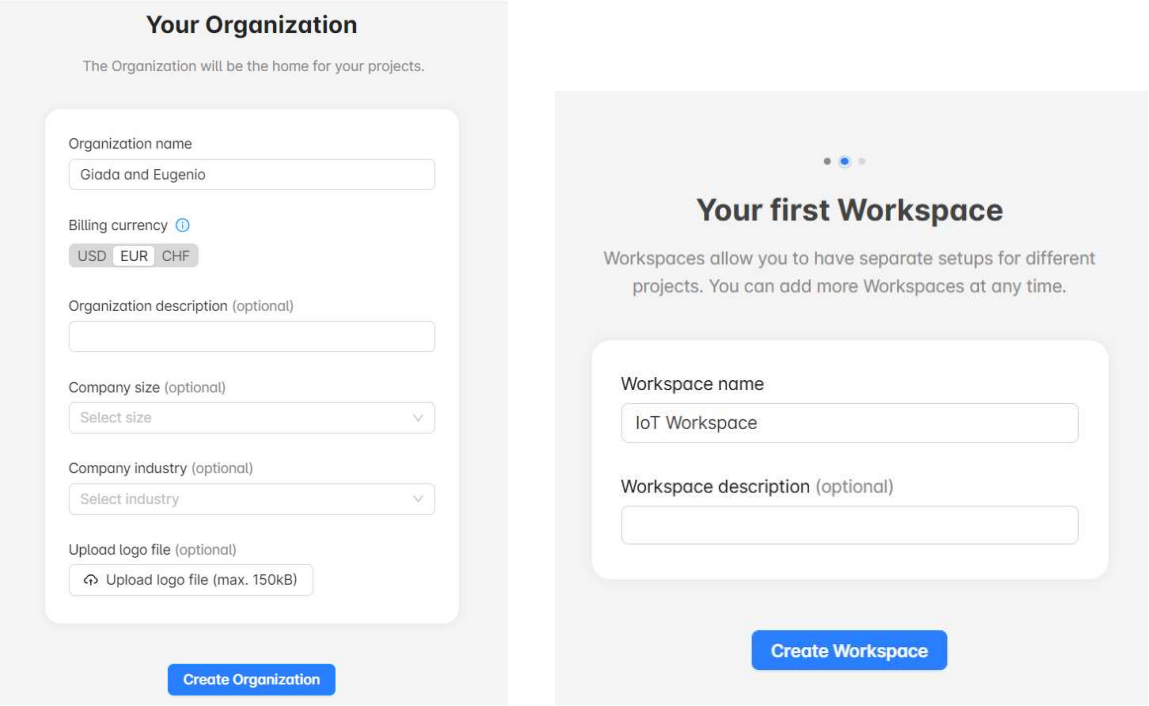


Figure 80 – a) Akenza, creating an Organization. b) Akenza, creating a Workspace.

At this point it was necessary to connect the Raspberry to Akenza, so Connect your first device → Start now was selected (Figure 81a).

Data Flows define which is the data source and how this data is processed and then stored in Akenza. In order to create the first Data Flow, +Create Data Flow was selected on the menu tab Data Flows (Figure 81b).

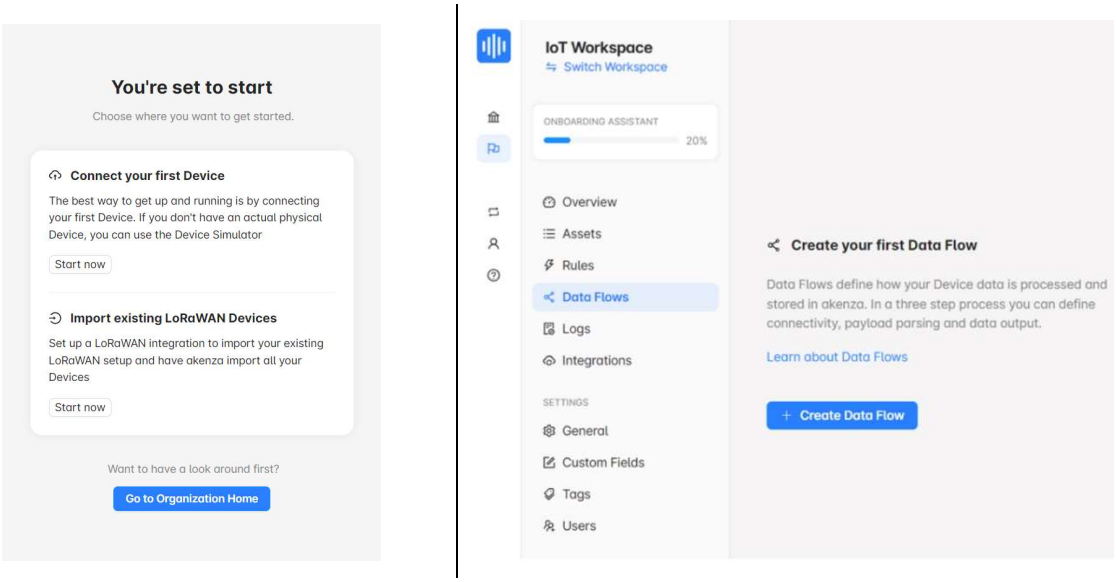


Figure 81 – a) Akenza, getting started. b) Akenza, Data Flows.

The Raspberry has to send the data via MQTT, so Connect a Device over MQTT was chosen (Figure 82).

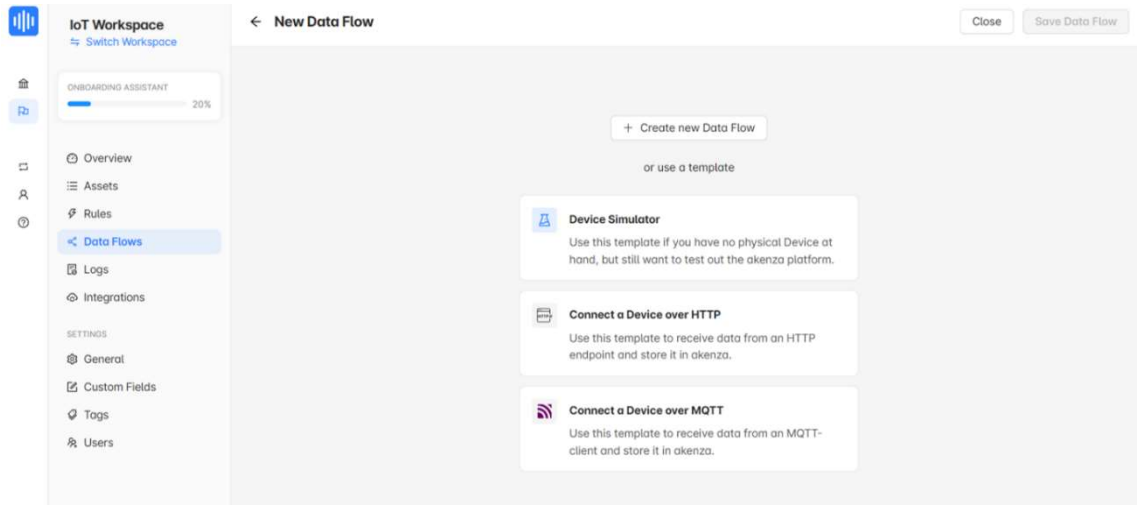


Figure 82 – Akenza, creating a Data Flow.

This specific data flow takes the data from a MQTT source (the Raspberry) and saves it on Akenza's Database without doing any kind of processing on it (Figure 83).

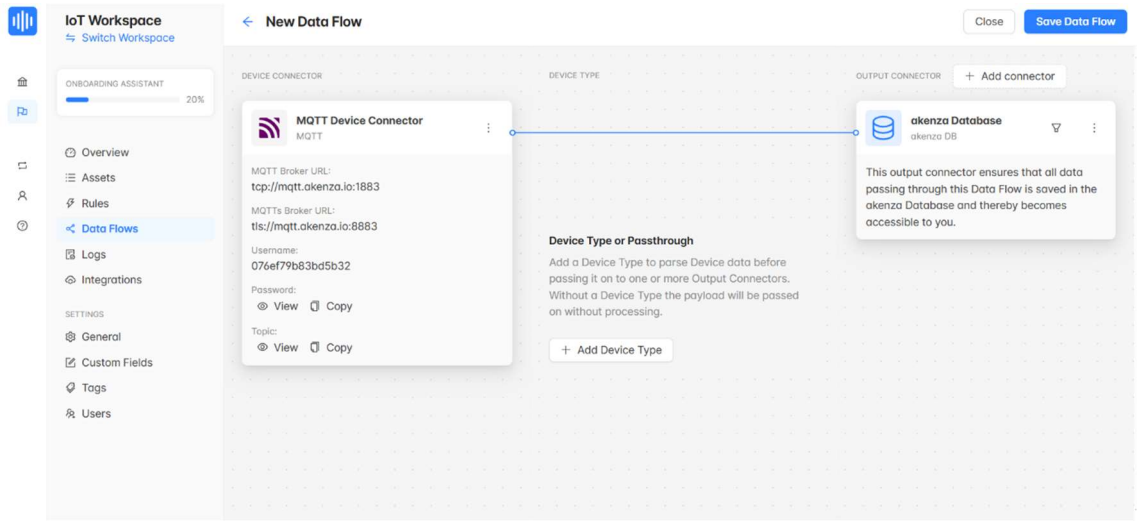


Figure 83 – Akenza, creating the MQTT Data Flow.

After creating the MQTT Data Flow, it was necessary to create the Raspberry device and to assign to it this Data Flow.

While saving the created Data Flow, the option Create Device with this Data Flow was selected (Figure 84).

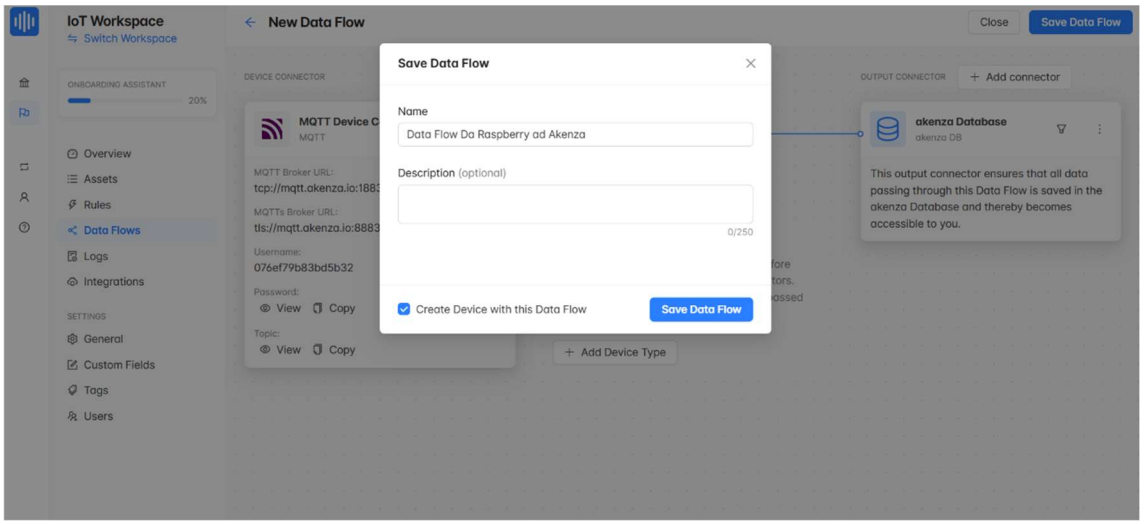


Figure 84 – Akenza, saving the MQTT Data Flow.

"Raspberry" was inserted as the name of the device (Figure 85).

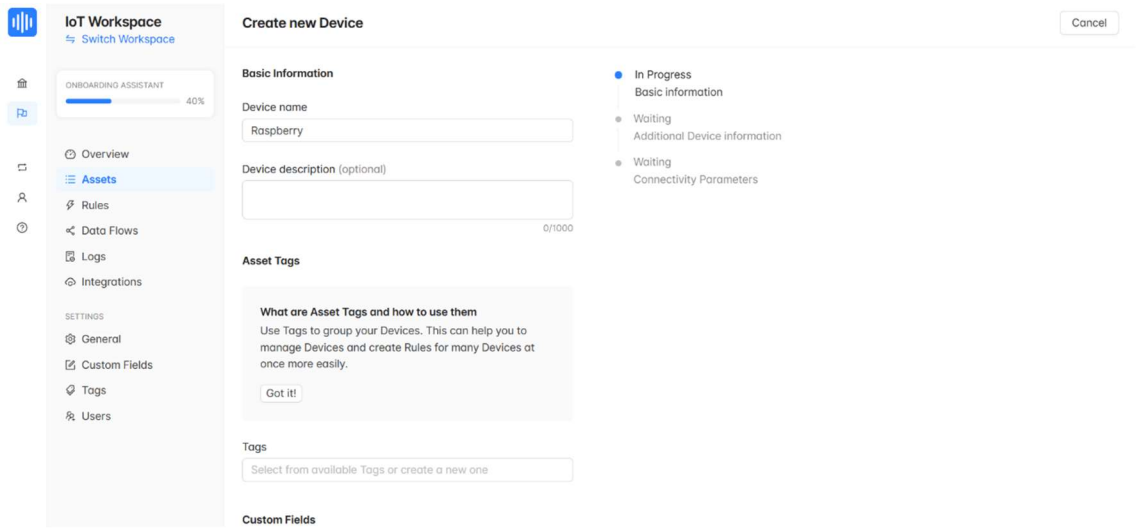


Figure 85 – Akenza, creating the Raspberry Device (part 1).

The created MQTT Data Flow was chosen (Figure 86).

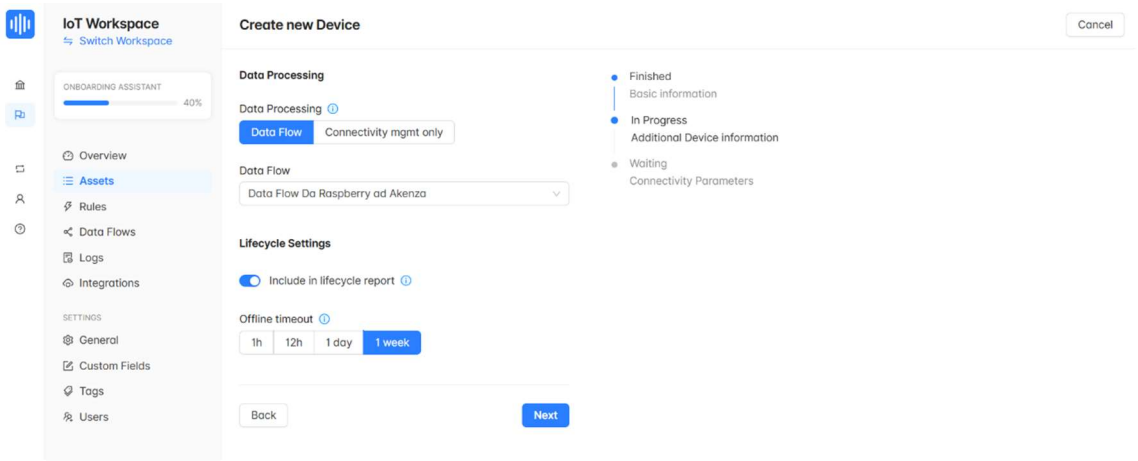


Figure 86 – Akenza, creating the Raspberry Device (part 2).

After selecting Generate ID, the Raspberry device was finally created (Figure 87 and Figure 88).

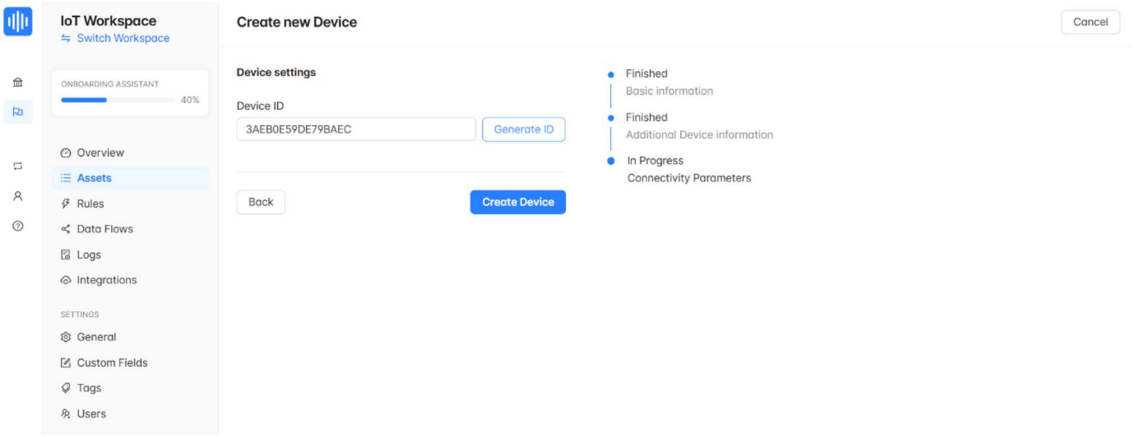


Figure 87 – Akenza, generating the Raspberry Device ID.

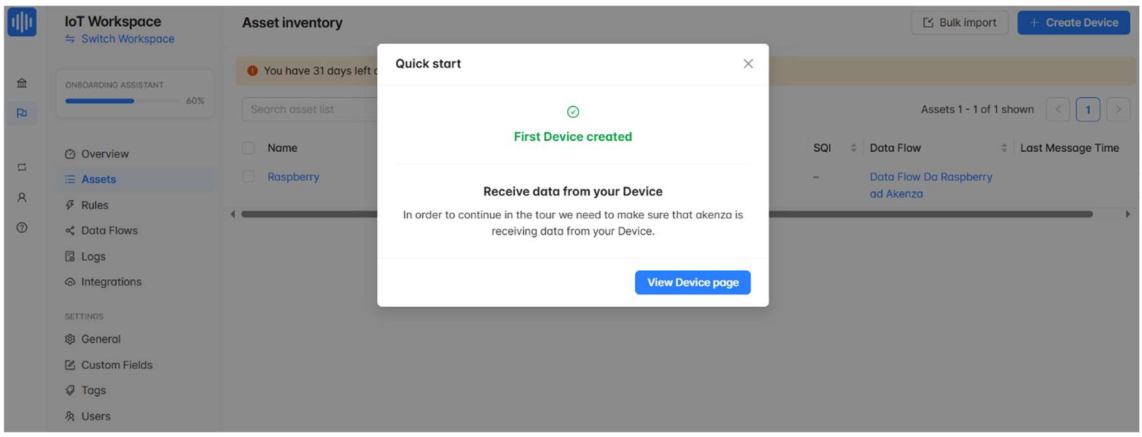


Figure 88 – Akenza, the Raspberry Device created.

Then on the command prompt of the Raspberry the line *pip3 install paho-mqtt* was inserted to install the libraries that allow the communication with Akenza via MQTT. Subsequently, in order for the Raspberry to send data to the specific Akenza account, it was necessary to write in its code the MQTT username, the MQTT password and the Topic that were obtained on Akenza (Figure 89).

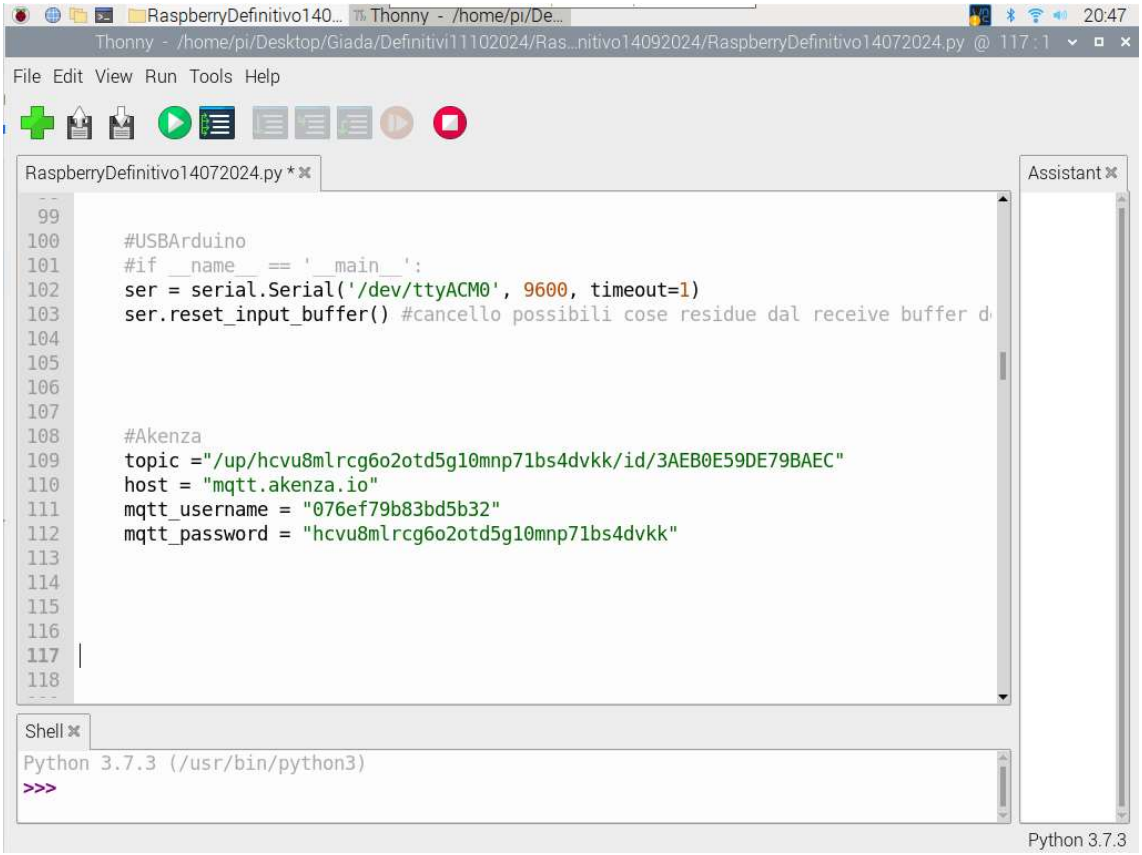


Figure 89 – Inserting the MQTT details into the Raspberry code.

4.4.1 Akenza Dashboard

In the Dashboard Overview, Start new was selected to create the personal Dashboard (Figure 90).

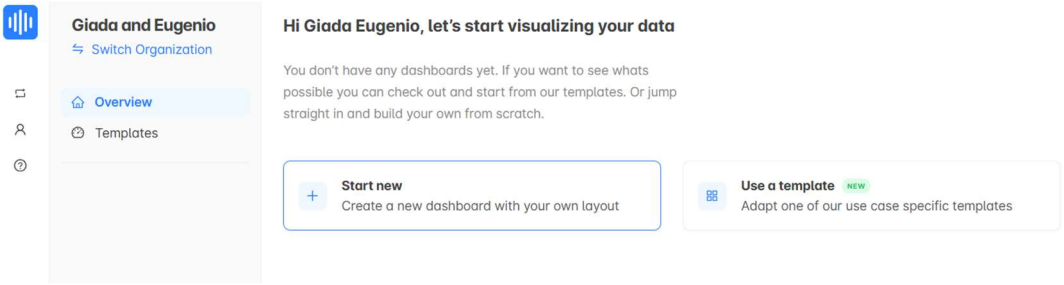


Figure 90 – Akenza, Dashboard homepage.

After inserting the name of the dashboard (Figure 91), it was necessary to define the Dashboard scope, which is the specific Workspace where the data to visualise is taken from (Figure 92).

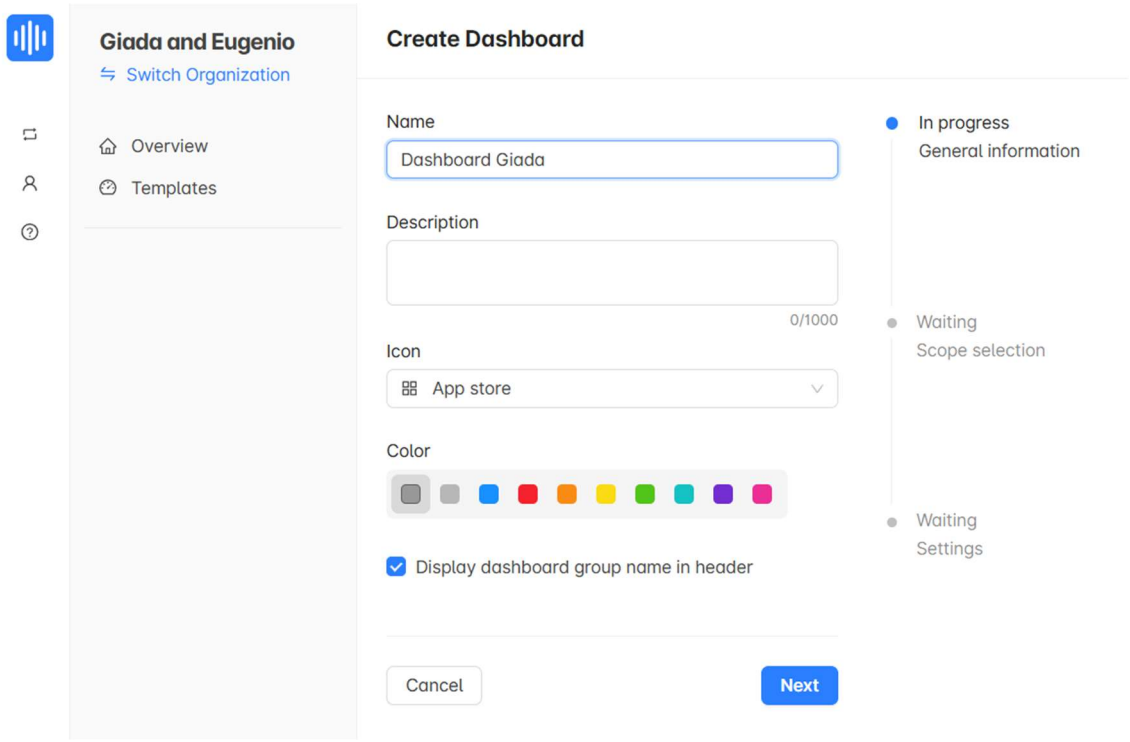


Figure 91 – Akenza, creating the Dashboard (General information).

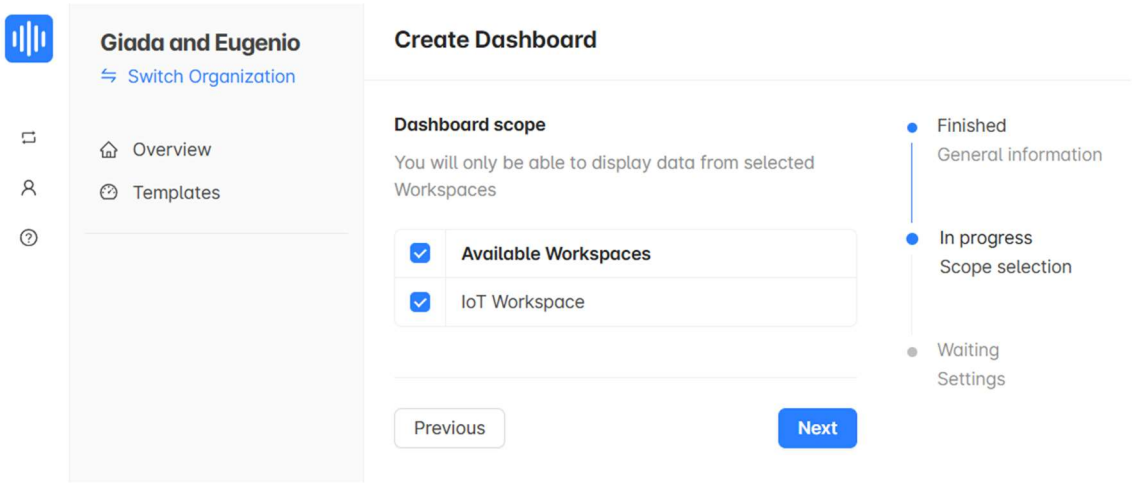


Figure 92 – Akenza, creating the Dashboard (Scope selection).

Finally, the default time range of the Dashboard and its Refresh time were chosen (Figure 93).

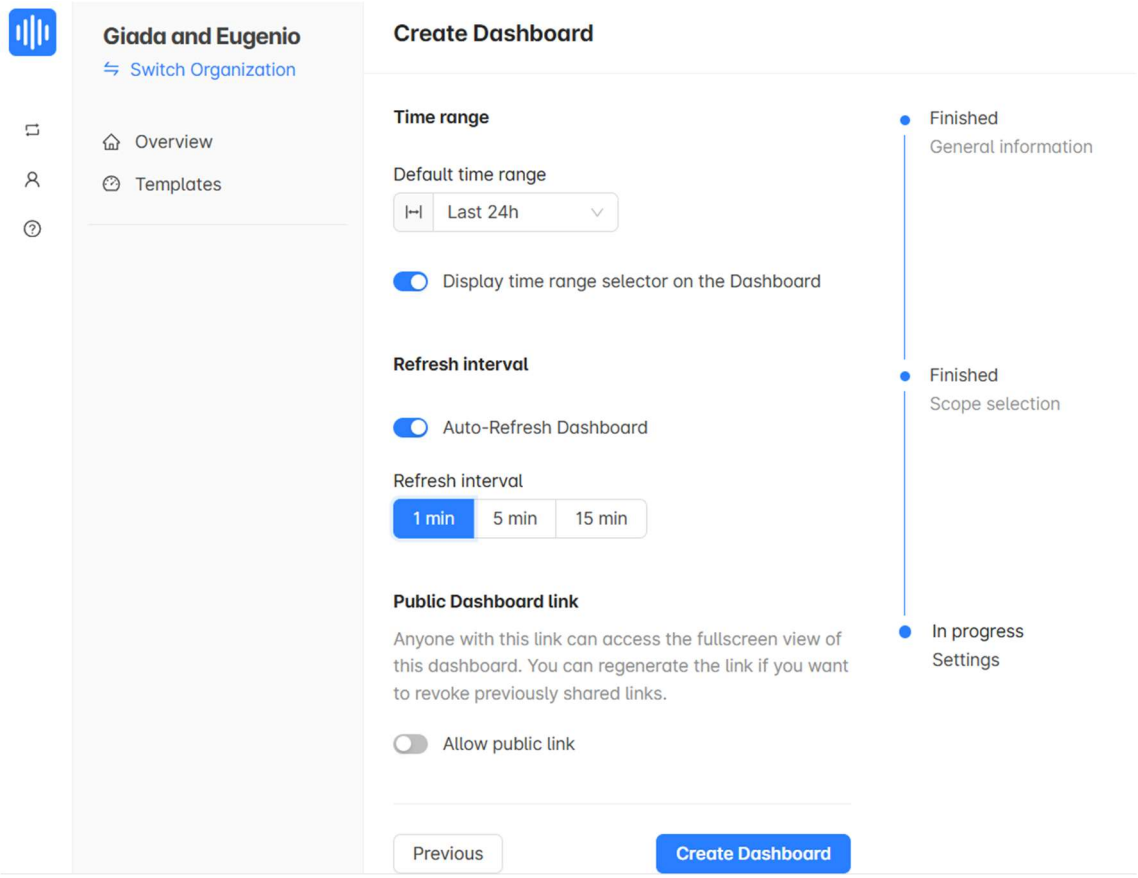


Figure 93 – Akenza, creating the Dashboard (Settings).

4.5 Integration between TTN and Akenza

This section describes how the Integration between TTN and Akenza was created. To create it, it was necessary to take the Application ID and the API key from the personal TTN and to insert them in Akenza.

The TTN Application ID was taken directly from Application → List on TTN (Figure 94).

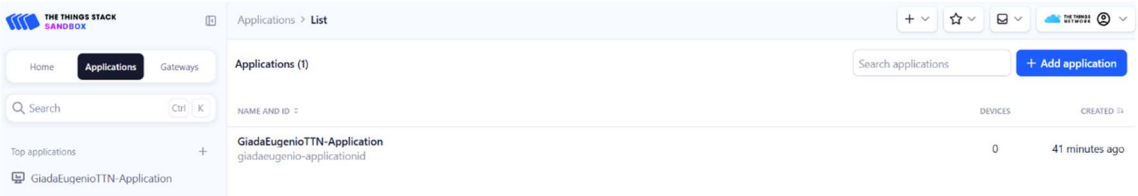


Figure 94 – TTN, Application ID.

The API key had to be created on TTN, by selecting API Keys → +Add API key. After choosing the correct settings (Figure 95), Create API key was selected (Figure 96).

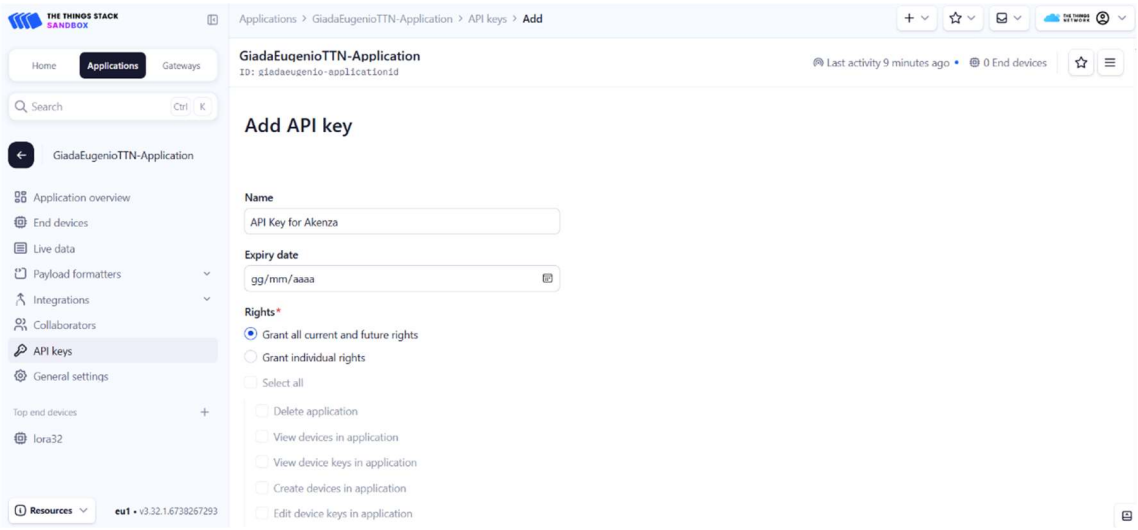


Figure 95 – TTN, creating an API key (part 1).

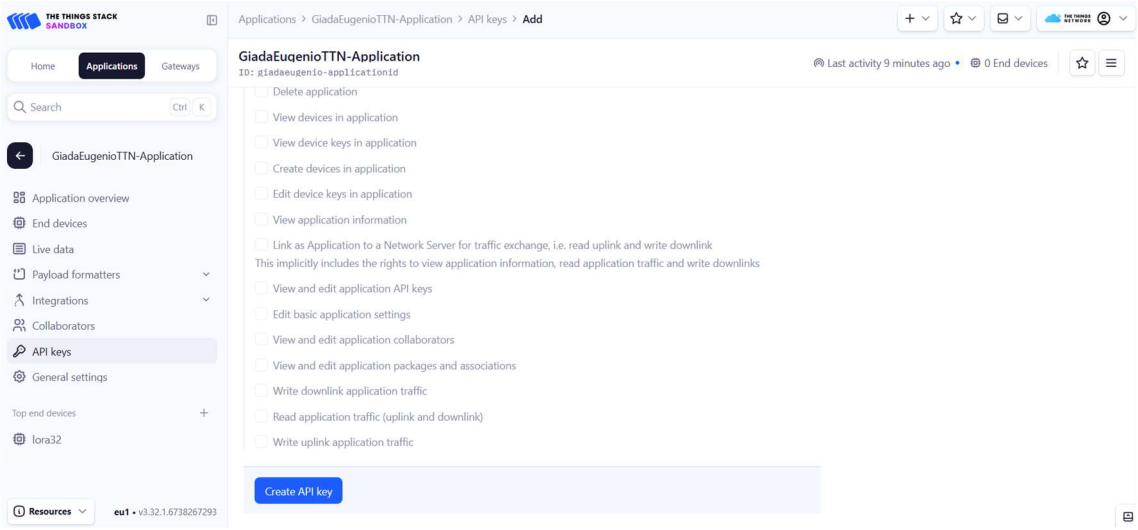


Figure 96 – TTN, creating an API key (part 2).

On the Akenza website Integrations → +Create Integration → TTN LoRaWAN was selected (Figure 97).

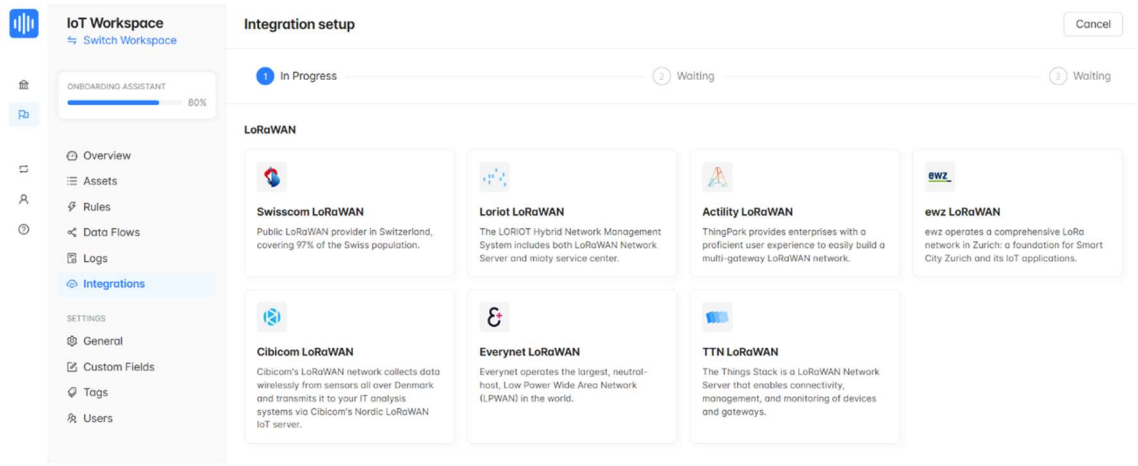


Figure 97 – Akenza, creating the Integration with TTN.

At this point the Application ID and the API key that were obtained from the TTN website, were inserted in Akenza (Figure 98).

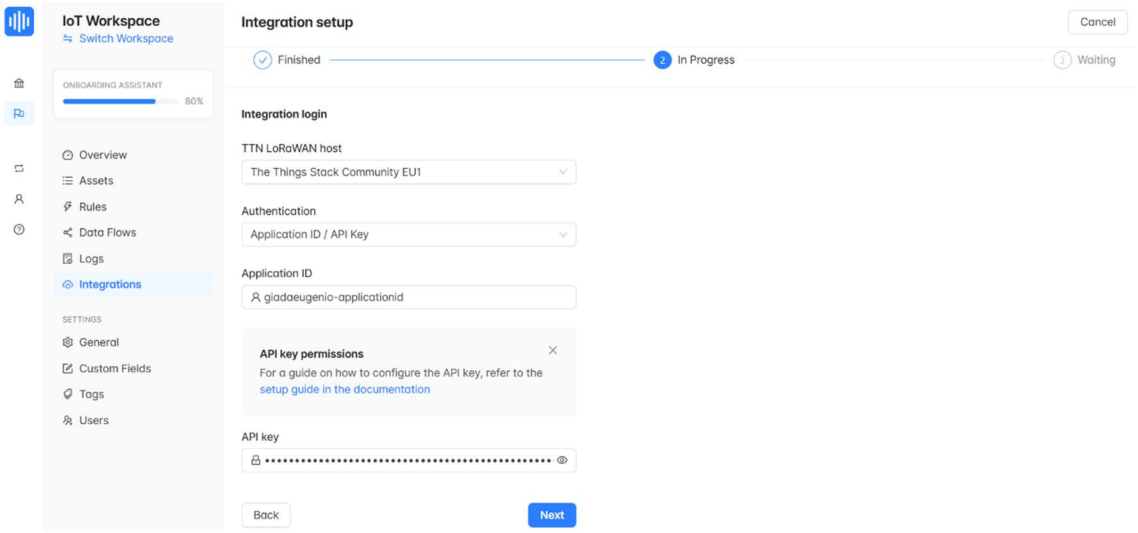


Figure 98 – Akenza, inserting the details from TTN.

Finally, an integration name was inserted, and the option Import Devices was automatically checked. This option allows Akenza to import the LoRa32 device that was created on the specific TTN Application (Figure 99).

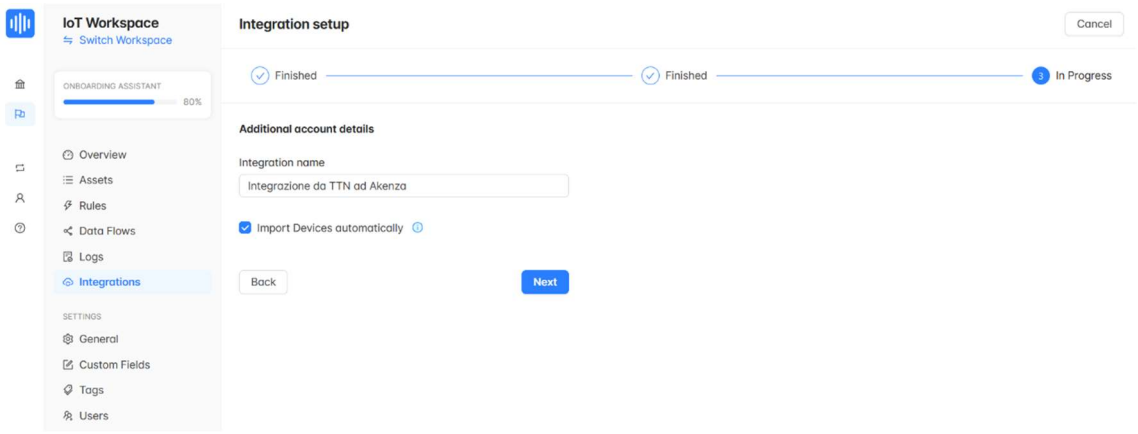


Figure 99 – Akenza, naming the Integration with TTN.

At this point, the integration appeared on both the Akenza (Figure 100) and the TTN (Figure 101) websites.

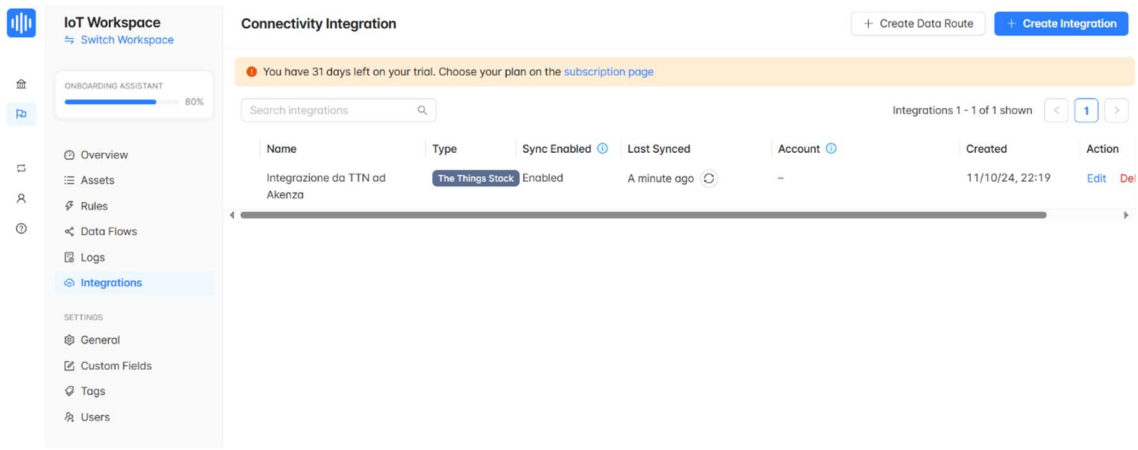


Figure 100 – Akenza, the created Integration.

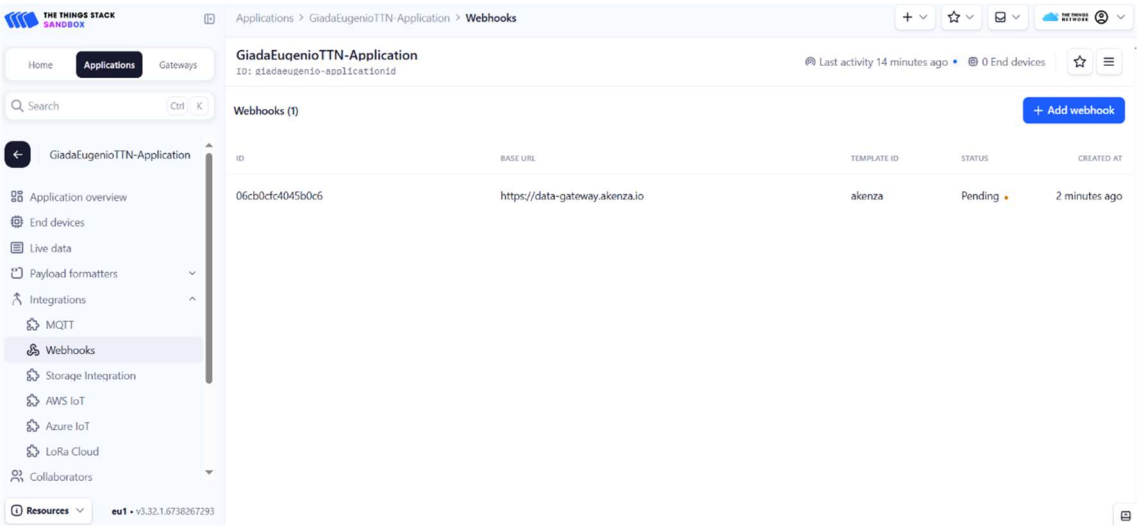


Figure 101 – TTN, the resulting Integration.

In Akenza, in Assets, the LoRa32 device appeared (Figure 102).

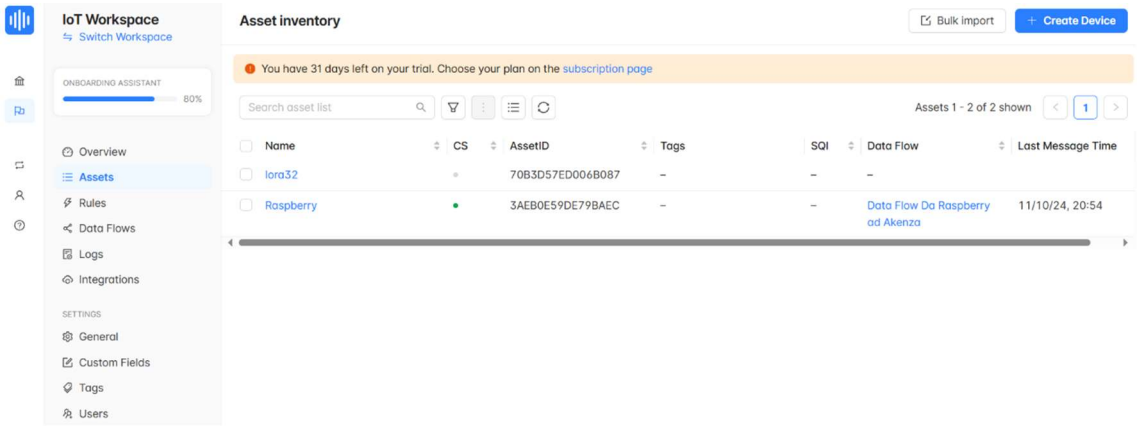


Figure 102 – Akenza, Assets.

The next step was to create the LoRaWAN Data Flow, which will be assigned to the LoRa32 device. "+Create Data Flow" was selected (Figure 103).

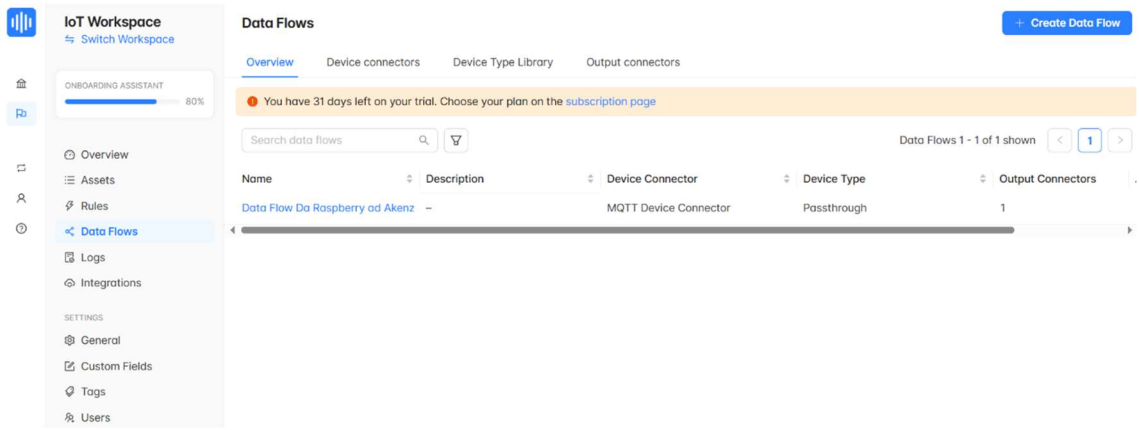


Figure 103 – Akenza, list of Data Flows.

In this second Data Flow, instead of using a template, it was necessary to create it manually, by choosing "+Create new Data flow" (Figure 104).

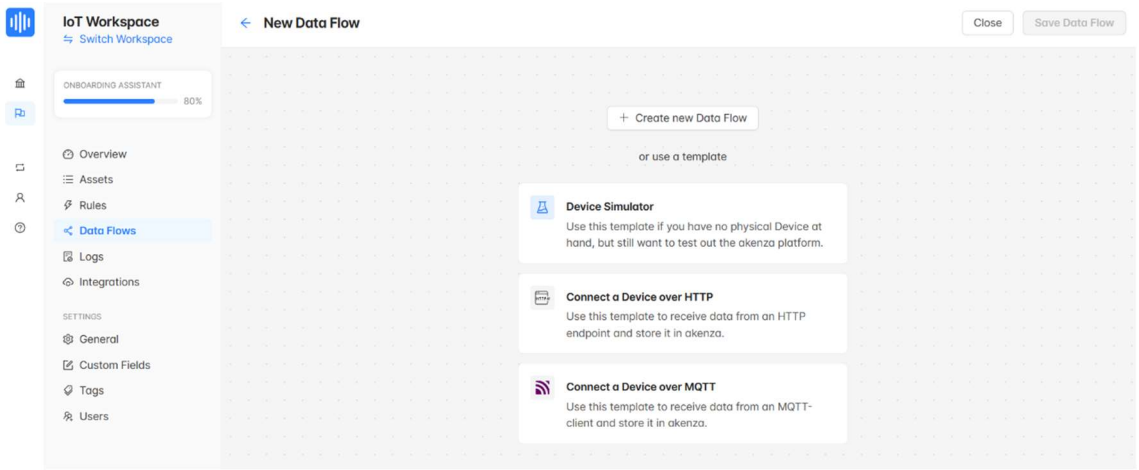


Figure 104 – Akenza, creating the LoRaWAN Data Flow.

In this second Data Flow, LoRa → The Things Stack → the created Integration with TTN was chosen as Device Connector (Figure 105).

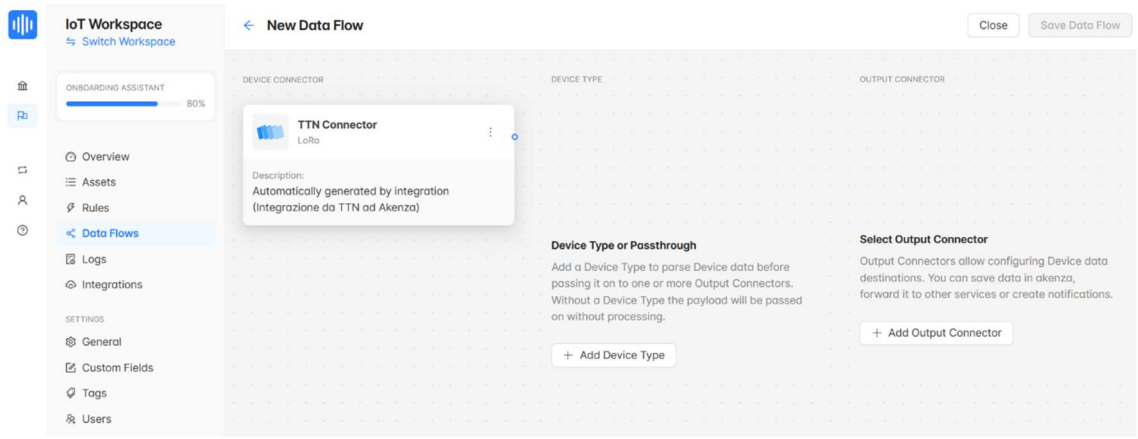


Figure 105 – Akenza, choosing the Device Connector.

Thereafter, Cayenne LPP was selected as Device Type (Figure 106).

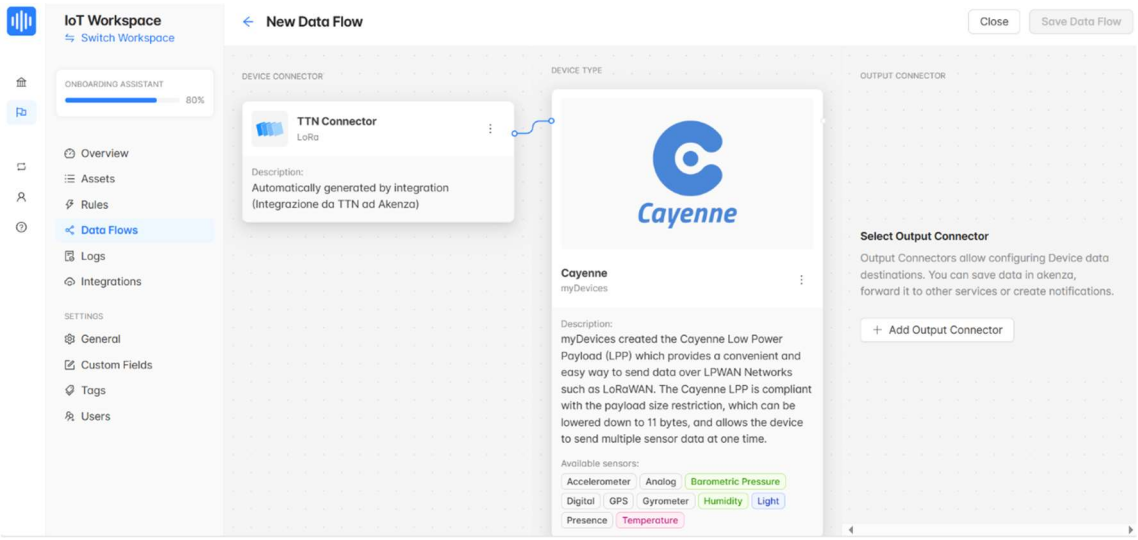


Figure 106 – Akenza, choosing the Device Type.

Akenza Database was chosen as Output Connector (Figure 107).

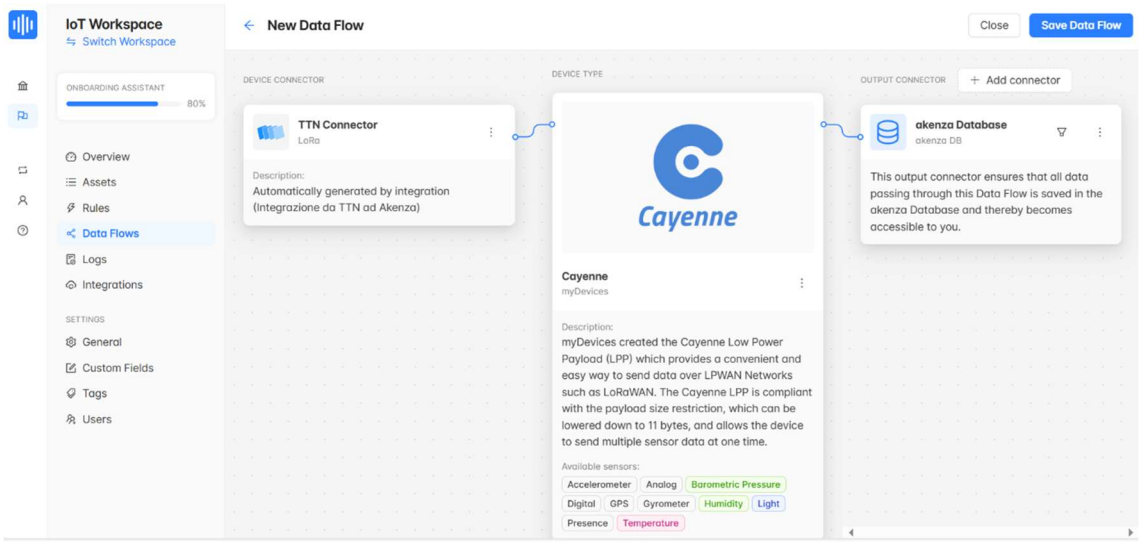


Figure 107 – Akenza, choosing the Output Connector.

For this new Data Flow, the data source is the integration with TTN, the data coming from TTN is in the Cayenne LPP format and has to be processed as such and the data is, then, stored in Akenza's database.

The subsequent step was to associate the LoRaWAN Data Flow created to the LoRa32 device (Figure 108), so in Assets → LoRa32 → Edit Device the LoRaWAN Data Flow was assigned to it (Figure 109).

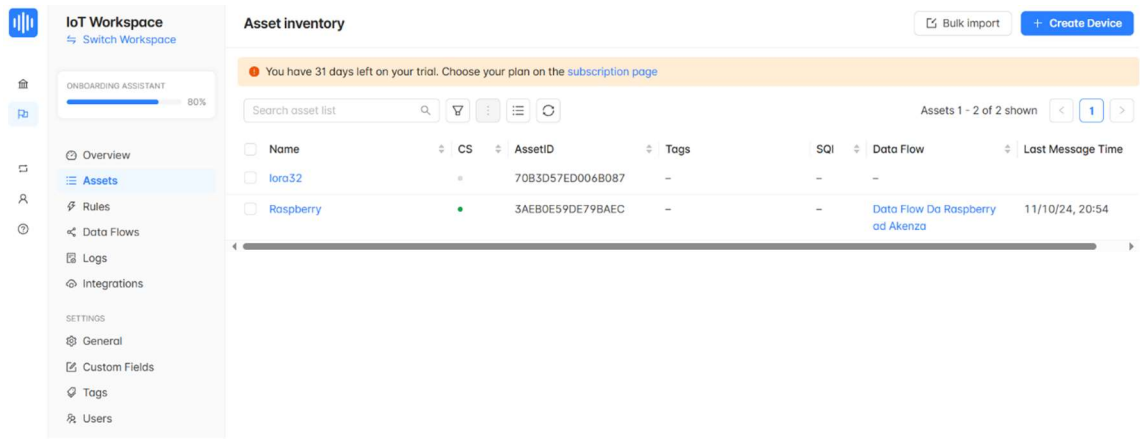


Figure 108 – Akenza, updated list of Devices in Assets.

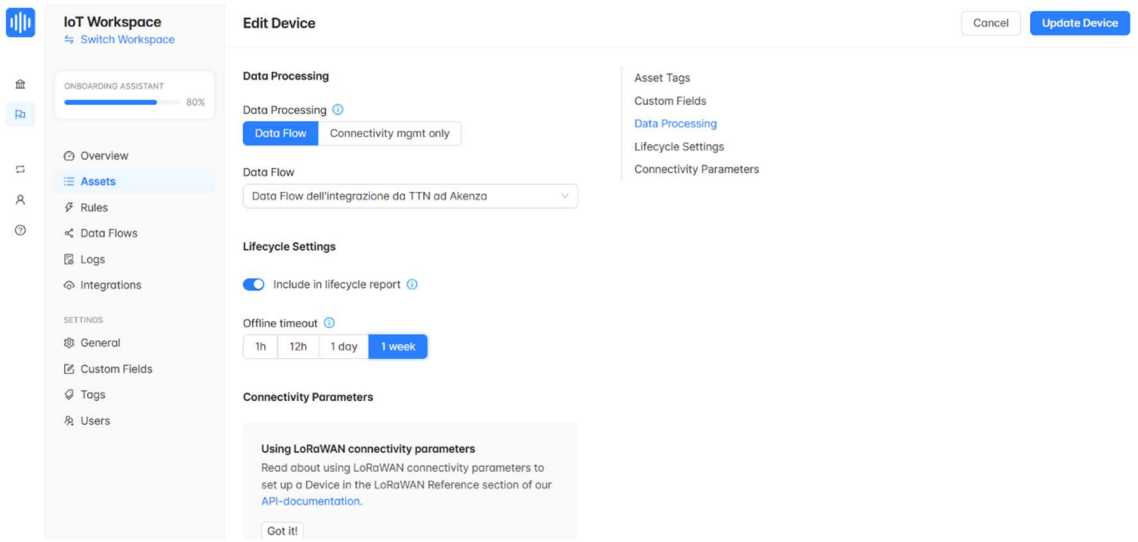


Figure 109 – Akenza, assigning the LoRaWAN Data Flow to the LoRa32 device.

4.6 How to Install Tesseract and OpenCV on a Raspberry Pi

In order to install the required packages for OpenCV on the Raspberry, the first step was to type the following command in its terminal:

```
sudo apt install libatlas3-base libsz2 libharfbuzz0b libtiff5 libjasper1 libilmbase12 libopenexr22 libilmbase12 libgstreamer1.0-0 libavcodec57 libavformat57 libavutil55 libswscale4 libqtgui4 libqt4-test libqtcore4
```

Next, this command was typed to install OpenCV for Python 3:

```
sudo pip3 install opencv-contrib-python libwebp6
```

Pip3 means that OpenCV will get installed for Python 3.

Next, the Tesseract library and the command line Tesseract tool were installed by typing:

```
sudo apt install tesseract-ocr
sudo apt install libtesseract-dev
```

Finally, Python wrapper for Tesseract was installed by typing:

```
sudo pip install pytesseract
```

4.7 Configuring the Tesseract OCR for improved results

Pytesseract allows to configure the Tesseract OCR engine by setting the flags. This changes the way of searching the characters in the image. The three main flags used in configuring a Tesseract OCR are language (*-l*), OCR Engine Mode (*--oem*) and Page Segmentation Mode (*- -psm*).

Along with the default English language, Tesseract supports many other languages and is also able to recognise two or more different languages from the same image. The language is set by the flag *–l*. In order to set a certain language, a code has to be used along with the flag. For example, to set the English language, it would be *–l eng*, where eng is the code for English.

The flag OCR Engine Mode has four different modes and each of them uses a different algorithm to recognise the characters from the image. This flag is indicated by *–oem* so, to set it to mode 1, it would be *–oem 1*. The four different Engine modes are illustrated in Figure 110.

```
OCR Engine modes:
  0    Legacy engine only.
  1    Neural nets LSTM engine only.
  2    Legacy + LSTM engines.
  3    Default, based on what is available.
```

Figure 110 – OCR Engine modes.

The page segmentation mode flag is very useful when the image has many background details along with the characters, or when the characters are written in different orientations or sizes. The flag is indicated by *–psm* so, to set the mode 11, it would be *–psm 11*. There are 14 different page segmentation modes, as shown in Figure 111.

```
Page segmentation modes:
  0    Orientation and script detection (OSD) only.
  1    Automatic page segmentation with OSD.
  2    Automatic page segmentation, but no OSD, or OCR.
  3    Fully automatic page segmentation, but no OSD. (Default)
  4    Assume a single column of text of variable sizes.
  5    Assume a single uniform block of vertically aligned text.
  6    Assume a single uniform block of text.
  7    Treat the image as a single text line.
  8    Treat the image as a single word.
  9    Treat the image as a single word in a circle.
 10    Treat the image as a single character.
 11    Sparse text. Find as much text as possible in no particular order.
 12    Sparse text with OSD.
 13    Raw line. Treat the image as a single text line,
       bypassing hacks that are Tesseract-specific.
```

Figure 111 – Page segmentation modes.

4.8 Tesseract training

Even if the counter digits are not rotating, Tesseract is not able to recognise the character used by the counter if it has not previously been trained. A further problem occurs when the digits are rotating and, therefore, are only partially visible. To ensure that Tesseract also recognised these digits (both whole and rotating), it was mandatory to carry out Tesseract trainings. Two different trainings were needed:

- An initial training was necessary to read the first 6 digits starting from the left and made use of whole digits.
- On the other hand, a second training was necessary to read the seventh digit and it made use of both whole digits and partial digits. It is important to observe that these are single digits seen partially, not images in which there are two digits both seen partially.

The process is explained in detail below.

4.8.1 Image collection

Firstly, it was necessary to collect hundreds of acquired images of whole digits and of all the intermediate positions of all the digits. The meter must be well lit and the camera must be always established in the same position. Thus, in all images, the digits are always illuminated in the same way, and are always in the same position. By proceeding this way, it was possible to work on only one image and then to use the same parameters for all the other images. For this purpose, the camera was inserted into the prototype and positioned very stably on top of the water meter. Figure 112 shows the prototype placed on the water meter. Outside the box, it is possible to see the power bank that was used to power the camera.

Figure 112 – Prototype placed on the water meter.

Figure 113 shows the inside of the box, where the camera was sealed with tape, so that the camera could not move.

Figure 113 – Camera fixed inside the prototype.

Therefore, a code was written to acquire an image every 15 seconds and to save it as pictureN.jpg, where N is a variable that increments after each shot.

4.8.2 Pre-processing of the images

If the image acquired by the camera is given in its normal state as input to Tesseract, this would not produce any results. Furthermore, it is not even possible to use it to train Tesseract. Instead, it was necessary to firstly carry out a comprehensive pre-processing of the image, and a second code was written. First of all, it was necessary to identify the regions of interest, which in this case were the single digits taken individually. Figure 114 shows the most significant digit taken individually.

Figure 114 – Digit 1 of an example image used for the training.

The next step was to convert the colour space of the image. OpenCV reads images in BGR (Blue, Green, Red) colour space by default. The cv2.cvtColor function, with the colour space conversion code cv2.COLOR_BGR2GRAY, converts the image to grayscale, as shown in Figure 115.

Figure 115 – Digit 1 in grayscale.

Blurring an image improves the result of Tesseract. A 5X5 kernel was used for the blurring (Figure 116).

$$\begin{pmatrix} 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \end{pmatrix}$$

Figure 116 – 5X5 kernel used for blurring.

This means that with the blurring operation (cv2.blur), the central element of this kernel passes over each pixel of the previous image, calculates the average of all the pixels under the kernel area and replaces that pixel with the average. The end result is Figure 117.

Figure 117 – digit 1 after blur.

At this point there is the main operation: the binarization. More specifically, it is the inverted binarization, since in Tesseract it is better to have black characters on a white background. For binarization, a threshold is chosen, so that all the pixels of the image having a value below the threshold become black (0), while all those having a value above the threshold become white (255). Since the binarization used is inverted, then all the pixels of the image having a value below the threshold became white (255), while all those having a value above the threshold became black (0). Figure 118 shows how the character was, at this point, clearer.

Figure 118 – Digit 1 after the inverted binarization.

At this point a morphological operation was performed, more specifically a closing operation (dilation + erosion). For dilation, a kernel was chosen and the centre of the kernel slid over each pixel. If there is a white pixel in the area covered by the kernel, then the pixel under examination becomes white. What dilation does, therefore, is to increase the white area to the detriment of the black area and is, therefore, useful for removing black disturbances such as the black stripe at the base of Figure 118. As regards erosion, however, the situation is the opposite. If there is a black pixel around the pixel under examination, in the area covered by the kernel, then the pixel under examination becomes black. In this way, the black digit which became thinner with dilation, with erosion returns to having the right thickness. Figure 119 shows the effects of closing applied on the digit.

Figure 119 – Digit 1 after closing.

Tesseract performs optimally when capital letters are around 32 px tall, so the last step was to resize the image as shown in Figure 120.

Figure 120 – Digit 1 after the resize.

The same procedure was used for all the digits of all the images. For the purpose of being used for the training, each single digit was saved as:

[langname].[fontname].[expN].[file-extension]

In this specific case, the images were saved as cifra.digits.expN.jpg, where N is a number that starts from 1 and increases for each image of the digits. Specifically, since each image has 7 digits, and this is digit 1 of image 11, then the image was called cifra.digits.exp71.jpg.

From the entire collection of images acquired, those with whole digits were initially chosen to do the first training.

4.8.3 Creation of the boxes

A software module was written for the purpose of creating the boxes. In the software, with the instruction *os.system*, it was possible to execute commands which should have been written in the prompt. The command to create the boxes is:

tesseract /path/cifra.digits.exp71.jpg /path/cifra.digits.exp71 --psm 10 makebox

Page segmentation modes (psm) was used in this command. Figure 111 contains all the possible values of psm.

By choosing psm 10, Tesseract knows that in its input images there are single characters, so that it can create more fitting boxes.

After creating the box, there were two files: the image file cifra.digits.exp71.jpg and the box file cifra.digits.exp71.box. This second file contains both the coordinates of

the box that outlines the digit and the value that Tesseract, without any training yet, associated with it. The jTessBoxEditor software (Figure 121) allows to graphically see the box and to possibly correct it, in case Tesseract was not able to do it well.

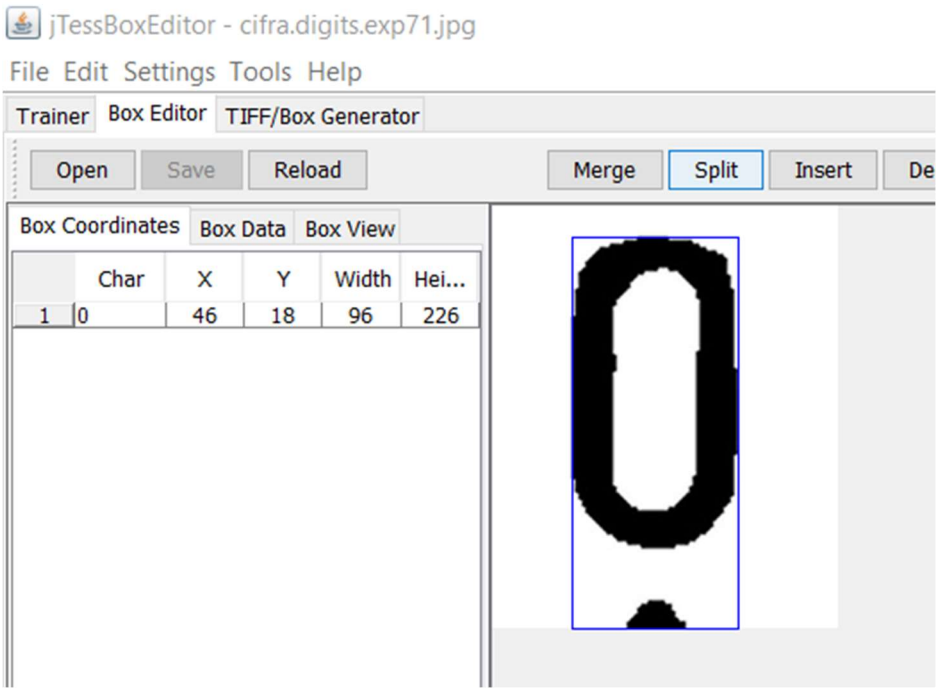


Figure 121 – jTessBoxEditor.

Furthermore, jTessBoxEditor also allows to possibly correct the value of the digit, in the event that Tesseract was not able to read it correctly.

4.8.4 Training

To do the training, firstly it was necessary to create the three folders, as illustrated in Figure 122, and to use the training.py script.

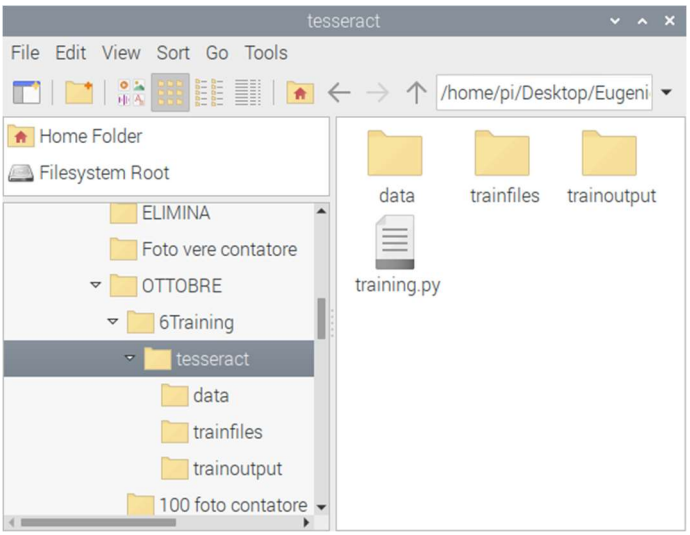


Figure 122 – Folders necessary to do the Training.

In the "data" folder there are all the images of the individual digits and their corresponding box file. The software training.py was written to automatically perform the operations, instead of writing them individually in the prompt. The *os.system()* function was used for this purpose.

The first operation of this software was to compound the jpg image with the box file, through the instruction:

tesseract cifra.digits.exp71.jpg cifra.digits.exp71 box.train

The output of this operation was the file cifra.digits.exp71.tr, located in the trainfiles folder.

The second step in training.py was to use unicharset_extractor tool to read the box files and to extract the unicharset, which is a set of characters that Tesseract can recognise. This is the command used:

unicharset_extractor cifra.digits.exp71.box

The output of this instruction was the unicharset file, which is in the trainfiles folder.

The third step was to create, again in the trainfiles folder, the font_properties file that contains the following text:

[fontname] [italic (0 or 1)] [bold (0 or 1)] [monospace (0 or 1)] [serif (0 or 1)] [fraktur (0 or 1)]

Therefore, in this specific case, it became digits 0 0 0 0 0.

The fourth step consisted in using all the files created so far to do the training, by using the instructions:

mftraining -F font_properties -U unicharset -O [langname].unicharset [langname].[fontname].[expN].tr
cntraining [langname].[fontname].[expN].tr

So, in this specific case:

mftraining -F font_properties -U unicharset -O cifra.unicharset cifra.digits.exp71.tr
cntraining cifra.digits.exp71.tr

The output of these two instructions were the shapetable, inttemp, pffmtable, normproto files in the trainoutput folder.

The fifth step of the code was to rename these files as:

[langname].shapetable	[langname].inttemp	[langname].pffmtable	[langname].normproto

Therefore, in the specific case, they became:

cifra.shapetable	cifra.inttemp	cifra.pffmtable	cifra.normproto

The sixth and final step of the code consisted in executing the combine_tessdata [langname] instruction, to create the [langname].traineddata file. So, the instruction used is:

combine_tessdata cifra

The cifra.traineddata file was obtained in the trainoutput folder.

All this procedure that was done on cifra.digits.exp71, had to obviously be done on all the other images as well. The software written performed these operations automatically for all the images used for training.

The file cifra.traineddata had to be copied and pasted into:

/usr/share/tesseract-ocr/4.00/tessdata

In the Raspberry, it was not possible to put anything in the tesseract-ocr folder, because the owner was *root*. To make the owner become *pi*, this line was inserted into the prompt:

```
sudo chown -R $USER:$USER /usr/share/tesseract-ocr
```

From this moment onward, it was possible to use the *cifra* language to read the first six digits through Tesseract. To make the training for the seventh digit, the procedure was identical so far. However, it was necessary to use also the images of the rotating digits, to which a further pre-processing step was added before creating the boxes.

4.8.5 Further pre-processing of the images for the last digit

From this second training, carried out using both the whole and the rotating digits, the result was the *cifra7* language.

Figure 123 shows an example of the seventh digit while its value is rotating from 0 to 1.

Figure 123 – Rotation of the last digit from the value 0 to the value 1.

In this situation, since the 0 has a greater height than the 1, the recognition produces a better result if the reading is made considering only the 0. First of all, it was necessary to search for all the edges of all the digits present in the image and to choose the edge of the digit with the greatest height. Therefore, in this case it was 0, as indicated in Figure 124.

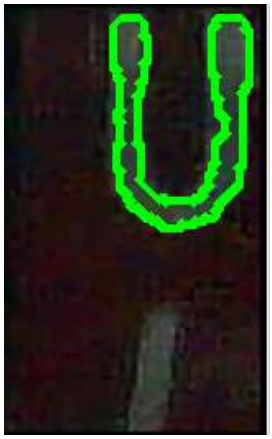


Figure 124 – Contour of the highest shape in the image.

By knowing which figure has the greatest height, the piece that had not to be read was removed, thus obtaining Figure 125.

Figure 125 – Last digit after cutting.

At this point the procedure, described in the previous subsection, was carried out, using this cut image to create the boxes and to perform the training.

4.9 Post-processing

The seventh digit (the least significant digit) rotates very quickly during the usage of the water. For this digit it was not possible to apply any post-processing algorithm. To read the seventh digit, it was necessary to use the language *cifra7*, which was obtained from the training with both the whole and the partial digits. From the first to the sixth digit the rotation is much slower. In this case there is no need to read the rotation of the digits. Therefore, the language *cifra* obtained from the training with only the fully visible digits was used in these cases. Between two consecutive images one of following three situations can occur:

- The current digit is the same as the previous one;
- The current digit is the same as the previous one + 1;
- The digit is rotating.

In the first two cases the number is clearly visible, and Tesseract has no problems in recognising the number. The third case is more critical because, in the event if Tesseract reads a value different from both the previous digit and the previous digit+1, then this digit must maintain the value of the previous reading.

In the developed software, for digits 1 to 6, there are three variables:

- *digitNprecedente*
- *digitNletto*
- *digitNconfermato*

Figure 126 is the flowchart of the algorithm used for each digit from 1 to 6 (in the example the sixth digit).

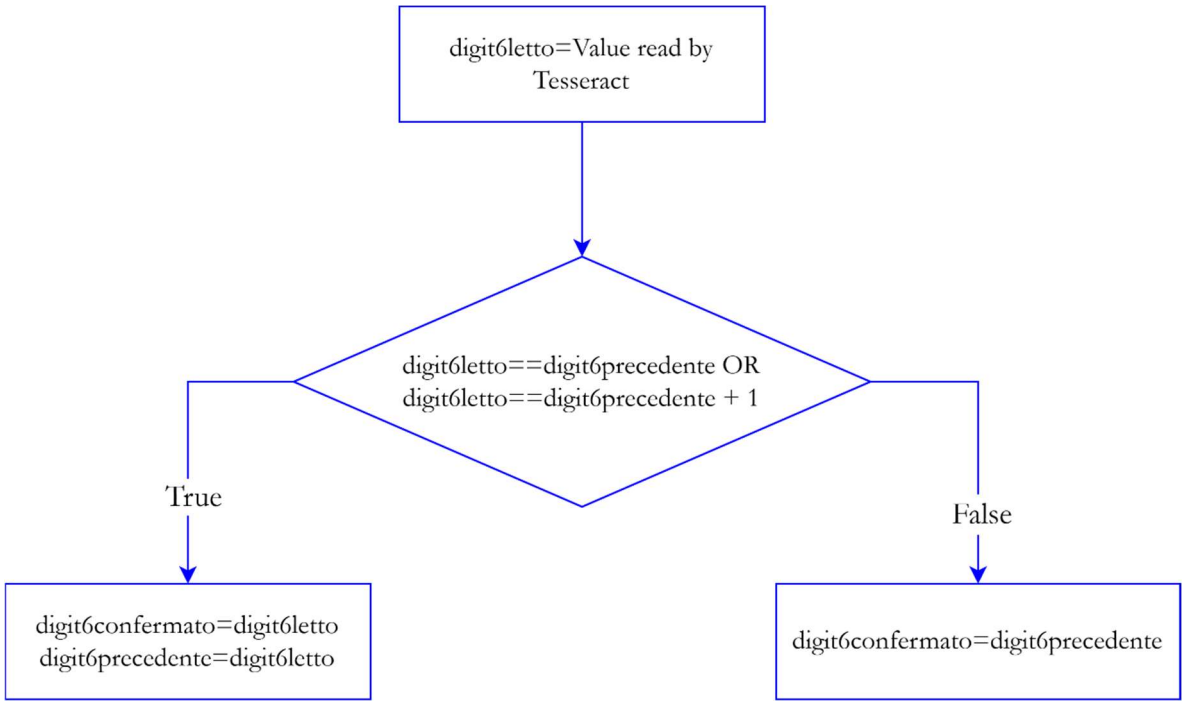


Figure 126 – Algorithm used for correctly read from digit 1 to digit 6.

The value *digit6precedente* is the last value that Tesseract read in the previous image. Tesseract, then, reads the current image and saves the newly read value in *digit6letto*. If *digit6letto* is equal to *digit6precedente* or to *digit6precedente*+1, then everything is ok, so *digit6confermato*= *digit6letto* and *digit6precedente*=*digit6letto*.

If, instead, Tesseract reads an unwanted digit, then this comparison is not true, so the new reading is not accepted and *digit6confermato*= *digit6precedente*. This situation can occur if the digit is rotating and, in this case, it is preferable to maintain the previous value. Then, when the rotation is completed, the new digit will be clearly visible and will be read correctly.

Solved Problem

If *digit6precedente*=9 then *digit6letto* can be 9 or 0. Since the algorithm, as simplified in the flowchart, expects *digit6letto* to be equal to *digit6precedente* (9) or equal to *digit6precedente*+1 (9+1=10), in this situation the algorithm would fail. This issue was solved creating the string *stringa*:

stringa=string(*digit6precedente*+1)

So now *stringa*=10, where 1 and 0 are char so, in other words, *stringa*[0]=1 and *stringa*[1]=0.

Now, knowing that len(*stringa*)=2 , this is the comparison that has to be made:

digit6letto==int(*stringa*[len(*stringa*)-1]), instead of *digit6letto*==*digit6precedente*+1

4.10 Description of the Software in the ESP-EYE

The software written for the ESP-EYE is mainly made up of two parts: the setup and the loop. The setup is represented by the flowchart in Figure 127.

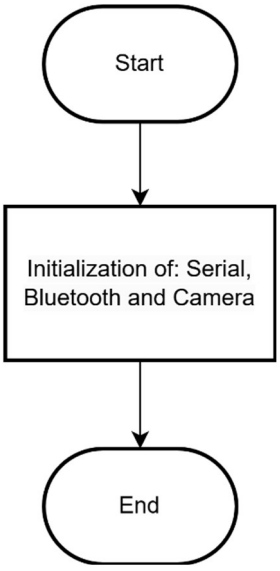


Figure 127 – ESP-EYE Software: *setup()* function.

Basically, in *setup()*, the serial communication is initialised and the *initBT()* and *initCamera()* functions are called.

The *loop()* function (Figure 128) is an empty infinite loop.

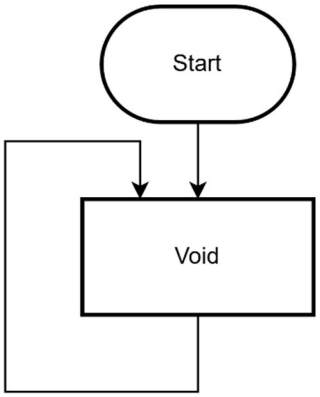


Figure 128 – ESP-EYE Software: *loop()* function.

After running *setup()*, the ESP-EYE enters this infinite loop and everything it does will be driven by Bluetooth events. All the other functions present in the code are described in detail below, starting from the functions related to Bluetooth.

4.10.1 The *initBT()* function

The flowchart in Figure 129 represents the *initBT()* function.

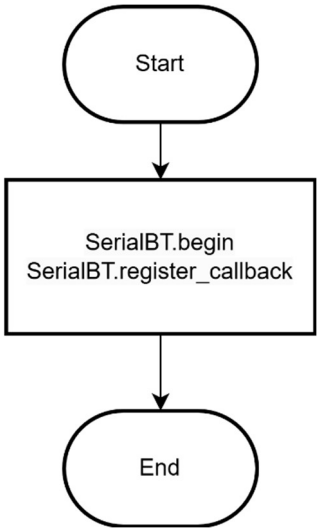


Figure 129 – ESP-EYE Software: *initBT()* function.

The *initBT()* function ensures that the ESP-EYE's Bluetooth is initialized correctly using the *SerialBT.begin* function. It also defines, by using the *SerialBT.register_callback* function, that whenever there is an interrupt from a Bluetooth event, the *btCallback* callback function should be called.

4.10.2 The *btCallback()* function

The *btCallback* function is able to distinguish four Bluetooth events:

1. Connection by a Bluetooth Client Event;
2. Writing Event;
3. Congestion Event;
4. Data Reception Event.

In case of reception of data, this function takes the received value and, subsequently, calls the *setCameraParam* function and gives it this parameter as an argument. The system is designed so that this parameter is sent from the Raspberry to the ESP-EYE to choose the image quality settings that the latter should use.

4.10.3 The *writeSerialBT()* function

The *writeSerialBT()* function is the last Bluetooth-related function. It has as its argument the image acquired by the camera and its sole purpose is to send it to the Raspberry, which is the one who made the request.

The following subsections describe the functions related to the camera.

4.10.4 The *initCamera()* function

The *initCamera()* function starts by assigning the GPIOs and the camera frequency and then chooses the image format. In this project the JPEG format was chosen for the image. The other possible choices were: YUV422, GRAYSCALE and RGB565.

The next step is to check if the ESP-EYE module is provided with an external PSRAM. The used module features an external 8 MB PSRAM chip, but it is only possible to use the lowest 4 MB. Since the PSRAM is present, then the choice was to use it. Moreover, since the memory is sufficiently high, then the best frame size and the best jpeg quality values were chosen. The frame size could be selected among the options represented in Table 11.

Table 11 – Frame size for the acquired images.

Frame Sizes
FRAMESIZE_VGA (640 x 480)
FRAMESIZE_SVGA (800 x 600)
FRAMESIZE_XGA (1024 x 768)
FRAMESIZE_UXGA (1600 x 1200)

The image quality (jpeg_quality) can be a number between 0 and 63. A lower number means a higher quality.

4.10.5 The *setCameraParam()* function

As aforementioned, when a parameter from the Raspberry arrives via Bluetooth to the ESP-EYE, then the *setCameraParam()* function is called. This function first sets brightness, contrast, saturation, white balance and exposure. Then, based on the parameter received, it chooses the frame size, which may be different from the one previously selected. The correspondence between this parameter and the frame size is represented in Table 12.

Table 12 – Correspondence between the parameter and the size frame.

Parameter	Function
0	FRAMESIZE_VGA (640 x 480)
1	FRAMESIZE_SVGA (800 x 600)
2	FRAMESIZE_XGA (1024 x 768)
3	FRAMESIZE_SXGA (1280 x 1024)
4	FRAMESIZE_UXGA (1600 x 1200)
5	Sleep

Finally the *capture()* function is called.

4.10.6 The *capture()* function

The purpose of this function is to acquire the image and to send it to the Raspberry. Two problems were encountered in this function, which were subsequently resolved.

1. If this function is called immediately, the images do not have the correct exposure. This happens because the Automatic Exposure Control (AEC) and the Automatic Gain Control (AGC) are highly related to the calculation of the "Luminance Average" YAVG, which is the last parameter written in the registers. In order to have the correct AEC and AGC, some seconds of stabilisation are required. The time to stabilise is related to the resolution (the higher is the resolution, the longer is the time) and to the luminance of the scene (the stronger is the brilliance, the longer is the time). To solve this issue, this function has to start with a delay of 6 seconds.

2. The acquired images were initially damaged, with some parts missing or with extended gray areas. To fix this bug, it was necessary for each image to be acquired twice immediately in succession.

After acquiring the image, this function gives it as an argument to the *writeSerialBT* function, in order to send it to the Raspberry.

4.11 Description of the Software in the Raspberry

The Raspberry is the core of the developed system, and the software inside of it manages all the aspects of the system. Figure 130 represents the flowchart of this software.

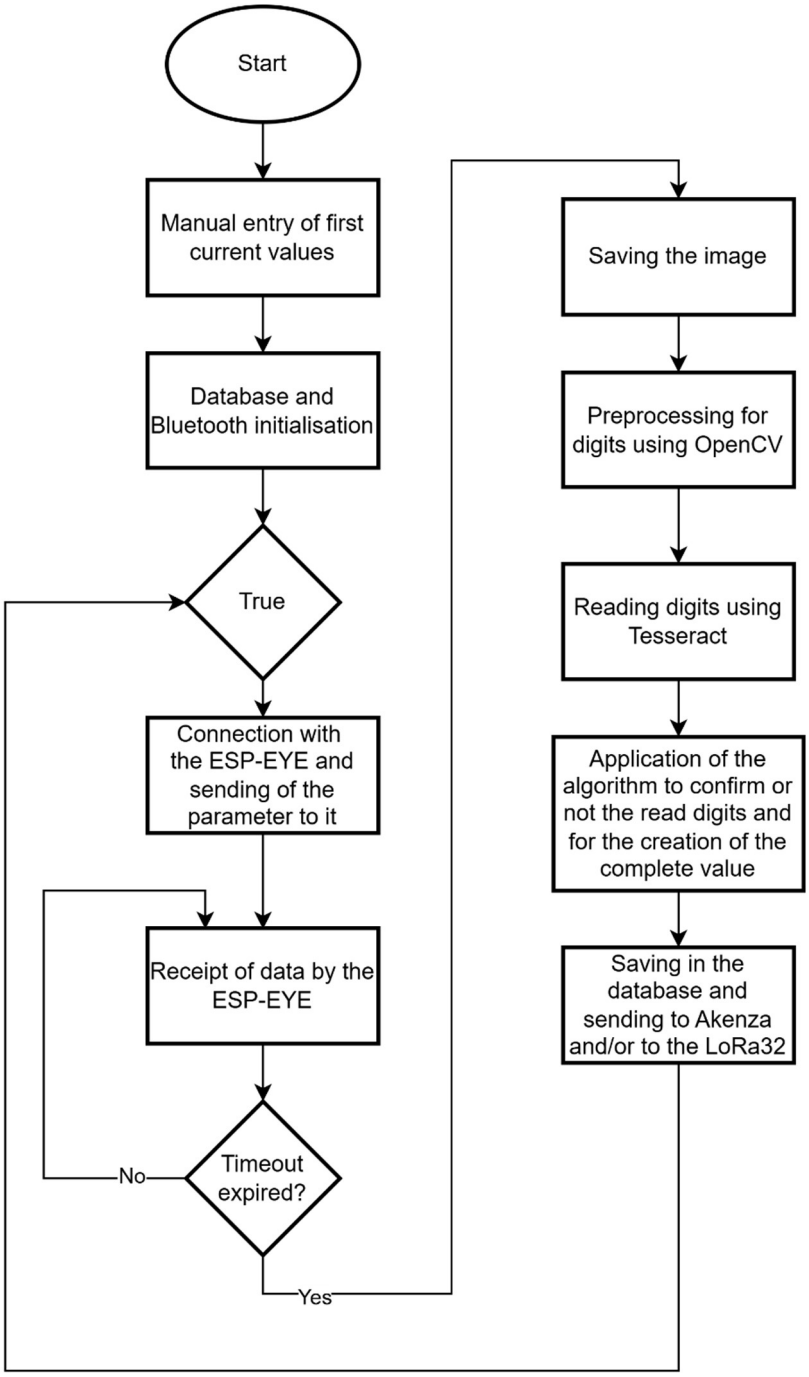


Figure 130 – Flowchart of the Raspberry Software.

After importing the necessary libraries, the current values of the water meter digits are entered manually. In the first reading taken by the system, these values are considered as if they are the values of a previous reading. Next, the database is initialised, and the table cell names are defined. The last initialisation done before the start of the instructions concerns the Bluetooth. In this initialisation, the MAC address of the server (the ESP-EYE), the timeout value, and the parameter to be sent to the ESP-EYE are defined. As already mentioned, the Raspberry, which is the

client in this Bluetooth communication, must send a parameter to the ESP-EYE, to ask for a picture with a certain frame size.

Henceforth, all software is in an infinite loop. Each new cycle starts by establishing the connection with the ESP-EYE and sending it the parameter. Then, the Raspberry starts to receive the image data from the ESP-EYE, using the RFCOMM protocol. The Raspberry continues to receive the data from the camera and to accumulate it into a single variable. When the ESP-EYE finishes sending the data, the Raspberry waits for the timeout to be triggered, after which it considers the transmission to be finished. At this point, the Raspberry saves the image as pictureN.jpg, where N is a variable that is initially 1 and is incremented with each cycle.

At this point, the pre-processing begins on the digits. What is described applies to all digits, from the first to the seventh one. The seventh, however, as explained above, requires further pre-processing. The first operation the software does is to crop from the entire image, only the Region of Interest (RoI), which is different for each digit. The RoI coordinates are entered manually based on the first image, so it is necessary for the prototype to stand still on the water meter to prevent the RoI from changing over time. For each individual digit, the pre-processing described above then begins, namely: conversion to greyscale; blur with a 5x5 matrix; inverted binarization with a different threshold for each digit; a morphological transformation (closing) and finally a resize, to give the digit the minimum size required by Tesseract. All these operations are done using the OpenCV library.

The seventh digit undergoes all the operations listed above, plus what will now be described. Since it is necessary to work with edges, the first operation done was to insert a thin white frame around the entire RoI. In this way, the digits appearing in this RoI are distanced from the edge of the RoI.

OpenCV searches for all edges in the image and considers the outer edge of the RoI as an edge in the image. If there is no space between the digit and the outer edge, sometimes it is possible that the digit is seen by OpenCV as a continuation of the outer edge but now, thanks to the added white frame, this can never happen.

Of all the edges found, the system identifies the highest one, which is the outer edge of the entire RoI, and sets the height of this edge equal to 0. At this point, the search for the highest edge is repeated again and this time the digit with the greatest height is actually found. Then, knowing which is the highest digit, the software cuts the image around this digit, thus eliminating the lowest digit from the picture.

Now that the digits have been correctly preprocessed, Tesseract is used to read the contents of the RoI. The *cifra* language is employed for the first six digits, while the *cifra7* language is used for the seventh digit. In all cases *psm10* is chosen, as each image contains only one character.

The digits are read by Tesseract as strings, so it is necessary to convert them to *int*. These read digits are saved into the variables *digitletto1*, *digitletto2*, ...*digitletto7*.

At this point, for the first six digits, the algorithm intervenes to confirm or to deny the value of the read digits. As already mentioned, between an image and the next one, the digit can, at most, be rotated by one position. If Tesseract reads a different change, it simply means that the digit was rotating at that moment and, therefore, the previous value must be maintained. In this case, the value of the current reading is confirmed to be the value read in the previous reading:

digit1confirmed=digit1precedente, *digit2confirmed=digit2precedente, ... digit7confirmed=digit7precedente.*

If, on the contrary, the increment is within the correct parameters, then:

digit1confirmed=digit1letto, digit2confirmed=digit2letto, ... digit1confirmed=digit1letto.

At this point, the digits are put together, respecting their weights:

1000·Digit1+100·Digit2+10·Digit3+Digit4+0.1·Digit5+0.01·Digit6+0.01·Digit7.

The last step is to save this value in the database and send it to Akenza via Wi-Fi and/or to the LoRa32 via USB, in order to be successively sent to TTN via LoRaWAN and to, finally, reach Akenza via integration. The Raspberry also calculates the water consumption that occurred between two consecutive images and, in addition to the value just read, sends also this value to Akenza.

4.12 Description of the Software in the LoRa32

The software running on the LoRa32 is composed of the setup and the loop. At the first power up of the LoRa32, the code starts with the setup, which calls for the first time the function *do_send()*. Figure 131 shows the flowchart of this function.

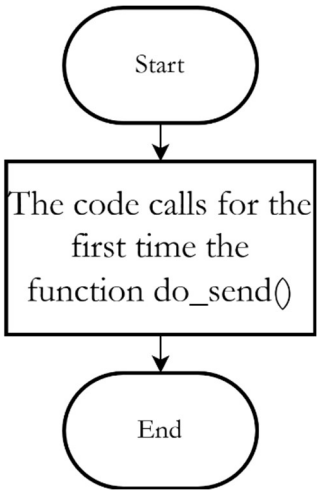


Figure 131 – LoRa32 Software: *setup()* function.

In the function *do_send()*, the LoRa32 waits for data from the Raspberry and prepares the actual data for the transmission to TTN.

Subsequently, the code continues with loop, whose function *os_runloop_once()* manages all the events (Figure 132).

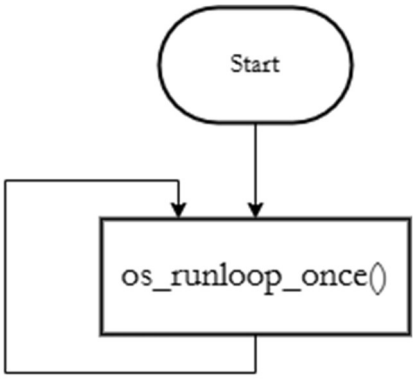


Figure 132 – LoRa32 Software: *loop()* function.

Since it is the first time that the LoRa32 has to send data to TTN, then *os_runloop_once()* calls the joining event (EV_JOINING) in order to create the connection between the end-device and the Join Server of TTN. As soon as the joining is successful, the algorithm continues with the loop, whose function *os_runloop_once()* calls the joined event (EV_JOINED). Subsequently, the transmission event (EV_TXSTART) effectively sends the prepared data to TTN. Thereafter, the next event transmission completed (EV_TXCOMPLETE) calls the *do_send()* function, in order to wait for new data and to prepare it for the next transmission. The events joining and joined only happen in the case of the first transmission to TTN.

From this moment onward, the algorithm will proceed in this order:

- In the function *do_send()*, the LoRa32 waits for data from the Raspberry, and prepares the actual data for the transmission to TTN.
- The event of transmission of the data sends the prepared data to TTN.
- The event of transmission complete calls the *do_send()* function.

These steps will continue to happen in this succession.

The *do_send()* function

Figure 133 illustrates the Flowchart of the *do_send()* function of the LoRa32 software.

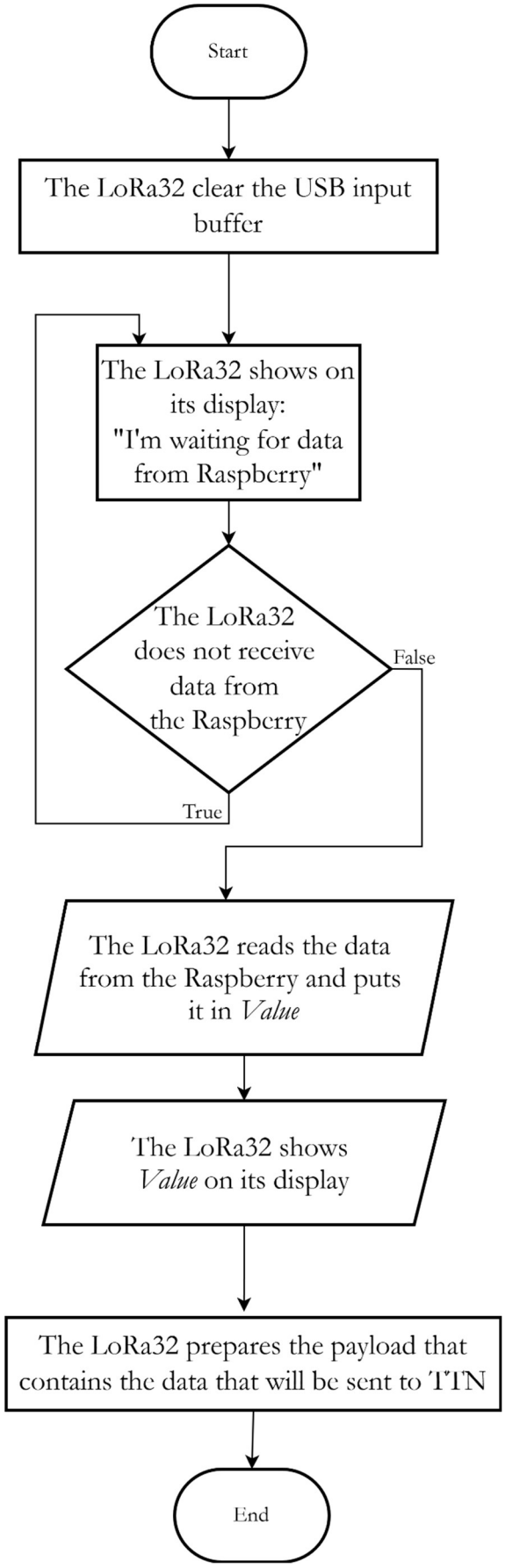


Figure 133 – LoRa32 Software: *do_send()* function.

At the start, the LoRa32 reads what is in its buffer to clean it. However, it is possible that, while the LoRa32 is busy doing other things, the Raspberry continues to send it values that accumulate in its buffer and therefore must be ignored. Then, the LoRa32 shows on its display "I'm waiting for data from Raspberry" and waits until there is data. As soon as this happens, the LoRa32 creates the string *Value* and reads the received data one char at a time and adds them to *Value* and, then, the LoRa32 shows this data on its display.

The array *mydata* is made of 4 bytes:

- *mydata* [0] contains the number of the channel of the Cayenne LPP payload. This number, however, is not used because Akenza does not use the Cayenne LPP channel.
- *mydata* [1] contains the type of data in the Cayenne LPP payload.
- *mydata* [2] contains the MSBs, the first 8 bits of int(*Value*). This variable is an integer and, since an *int* is made of 2 bytes, only the most significant byte are put into *mydata*[2].
- *mydata* [3] contains the LSBs, the last 8 bits of int(*Value*).

At this point, the function *do_send()* ends and the software returns to the *loop()*, where event EV_TXSTART is called to send the data to TTN.

This chapter began with a description of the initial procedures for using the Raspberry, TTN, Akenza, Tesseract and OpenCV. Next, the chapter explored the Tesseract training, with an in-depth analysis of the pre-processing of the images used for the training and of the post-processing algorithms. The chapter concluded with the detailed presentation of the software running in the ESP-EYE, the Raspberry and the LoRa32. The next chapter is going to show the use of the developed device on a real functioning meter.

5 EXPERIMENTAL AND FIELD TEST RESULTS

This chapter initially shows a real experience in which the system was attached to a water meter. Next, there is an analysis of the partial tests, that were made using the same images of the total experience, for the purpose of verifying that all the components of the system work correctly and of pointing out certain situations.

5.1 Field Test Results

Obviously, for the purpose of doing this test, the training procedure was not repeated; instead the *cifra* and *cifra7* languages were used. It was important to establish the prototype in the same exact position used to acquire the images for the training.

Figure 134 is a real image of a home water meter acquired by the ESP-EYE.

Figure 134 – Example of an image acquired by the device.

The ESP-EYE allows manual focusing by rotating the crown around the camera but, since the crown was blocked with glue, it was necessary to remove it by using a cutter.

The experiment began on 25/11/2024 at 08h:18m and ended at 09h:45m. During this experiment, since the meter used was within the Wi-Fi range, both the Wi-Fi network and the LoRaWAN network were used simultaneously. In a normal use, it is possible to use only one network if the other is not present, or it is possible to use the Wi-Fi network as the main network and, in case of absence of the Internet connection, the system automatically switches to the LoRaWAN network.

The developed system sends the read value to Akenza as soon as it is available. Since the camera takes approximately 7 seconds to acquire the image properly (if a shorter time is used, the exposure value is not applied), and the timeout after which the Raspberry considers the Bluetooth transmission complete is 7 seconds, then the values are sent to Akenza approximately every 14 seconds.

Figure 135 indicates the data received from Akenza via MQTT during the experiment.

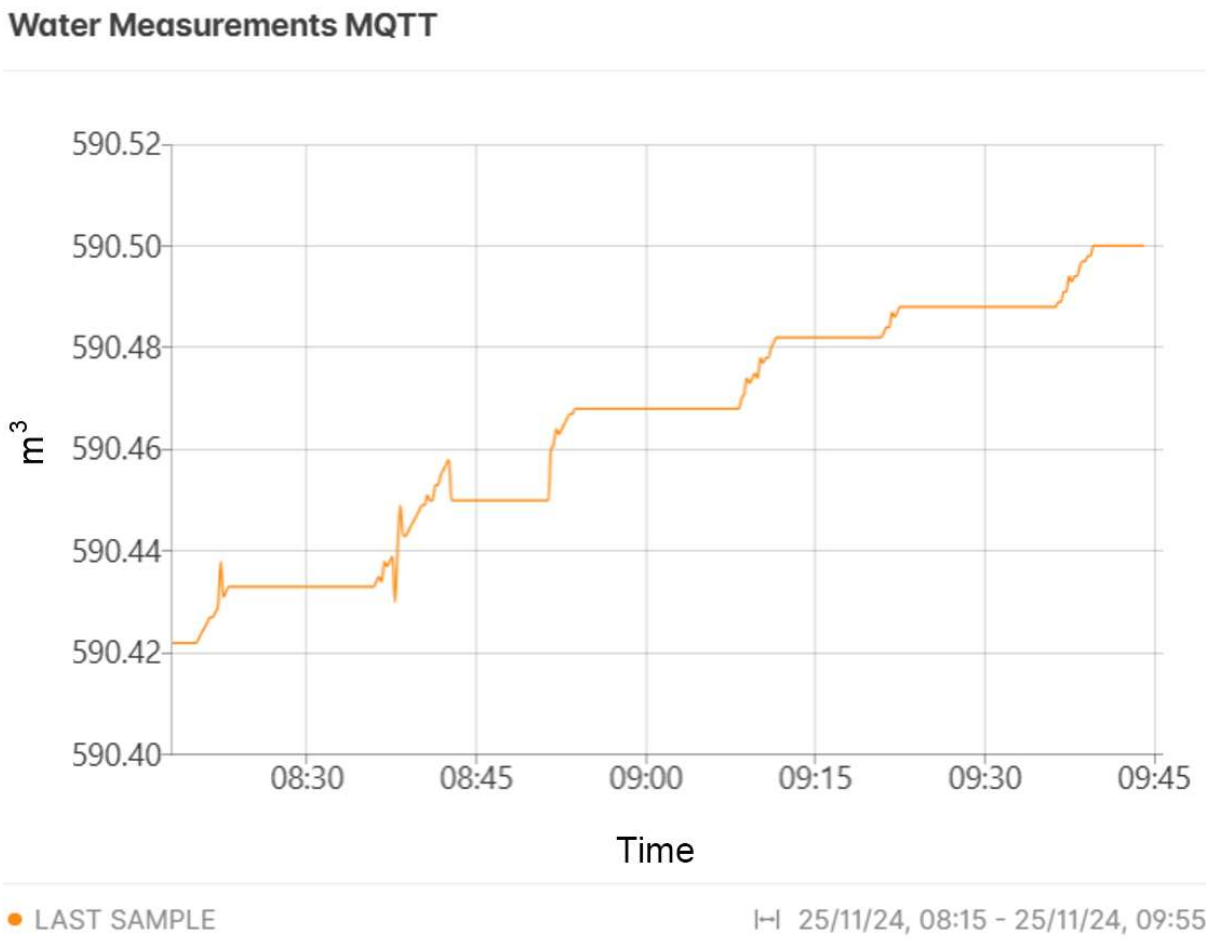


Figure 135 – Water Measurements MQTT.

During the hours of the experiment, the water was used only when necessary, therefore there are periods during which the water is not used and on the graph the value read remains constant. Sometimes the graph presents some peaks. These peaks

seem large due to the scale used but, in reality, they are only small errors that sometimes occur on the third decimal digit (the seventh digit).

Figure 136, instead, indicates the data received from Akenza via LoRaWAN.

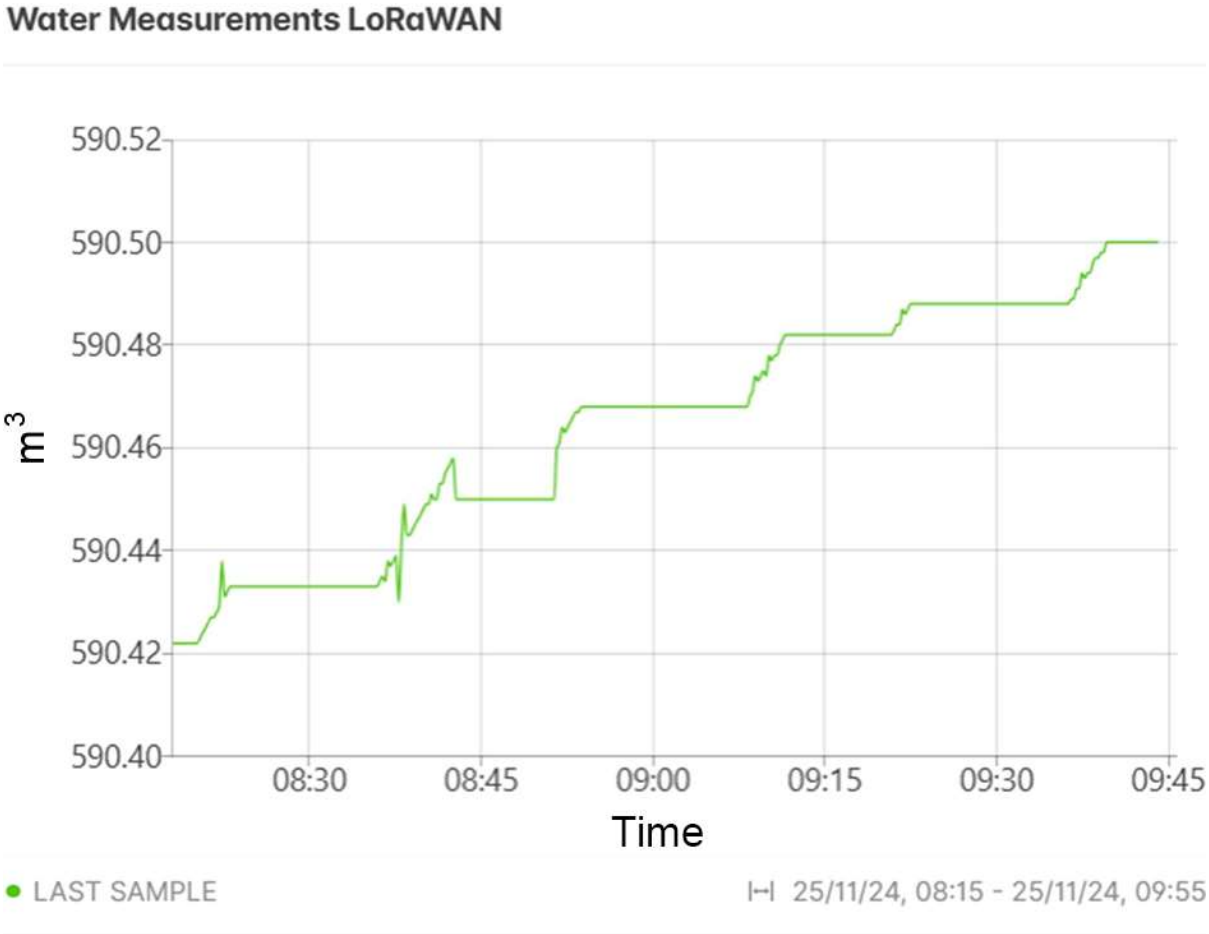


Figure 136 – Water Measurements LoRaWAN.

Since for the experiment the data was sent simultaneously both via Wi-Fi and via LoRaWAN, the two graphs are the same. Figure 137 firstly shows the LoRa32 waiting data from the Raspberry. When LoRa32 receives data for the first time, it tries to make the joining event and, if it has success, on the LoRa32 display the word "JOINED" appears.

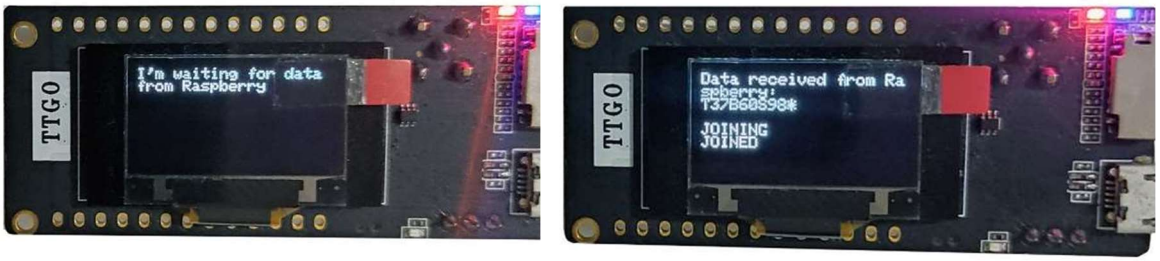


Figure 137 – Joining event on the LoRa32.

As already seen in the subsection concerning the software analysis, the Raspberry, in addition to sending the read value to Akenza, also sends the value of the consumption that occurred between two consecutive images. The Raspberry subtracts the value read in the previous image from the value read in the current image, in order to calculate the water consumption. Akenza can automatically sum the individual values received in the desired time interval and represent them graphically. Figure 138 indicates the water consumption in the last hour.

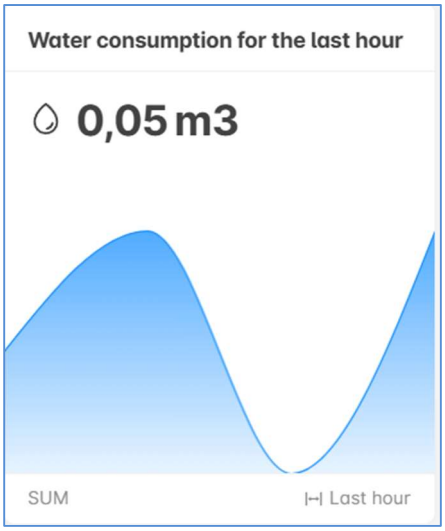


Figure 138 – Water consumption for the last hour.

The first value sent to Akenza at 08h:18m is 590.422 m^3 (Figure 139).

Figure 139 – First value sent to Akenza.

The value read at 08h:44m is 590.45 m^3 (Figure 140) and the one read at 09h:44m is 590.5 m^3 (Figure 140).

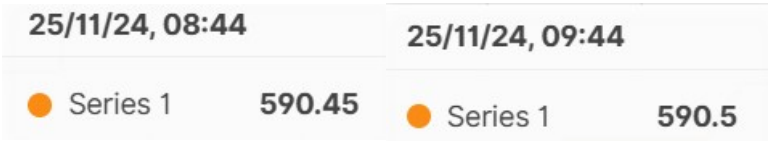


Figure 140 – Values read one hour apart.

The consumption of the last hour is 0.05 m^3 (590.5 m^3 - 590.45 m^3), as indicated by the Akenza graph (Figure 138). All the data was also saved in the database.

5.2 Reading using the English language

Even after applying all the pre-processing described in the previous chapters to make the digits perfectly visible, when using the default English language (Figure 141), the reading carried out by Tesseract is completely wrong.

```
custom_config = r'-l eng--oem 3 --psm 10'
custom_config7 = r'-l eng --oem 3 --psm 10'
```

Figure 141 – Setting the default English language.

Figure 142 shows the result from Tesseract when using the default English language on the preprocessed image.

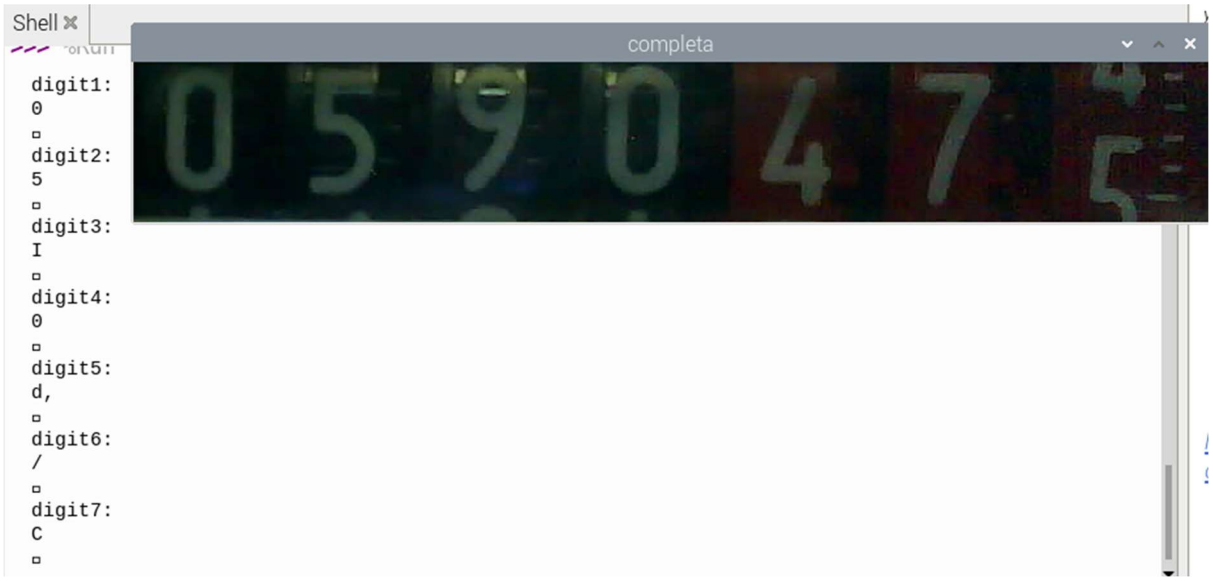


Figure 142 – Reading carried out using the English language.

The number 0590475 is read as "05I0d,/C". This demonstrates that the training is absolutely necessary to read these characters, also when reading whole digits.

5.3 Reading of a partial digit

In this test the system read the seventh digit. After the pre-processing, the rotating digit appeared as represented in Figure 143.

Figure 143 – Rotating digit after pre-processing.

The algorithm found all the edges present in this image and only the second highest was taken into consideration (the first was the rectangle that encloses the entire image). The focus was, therefore, shifted only on the number 6 (Figure 144).

Figure 144 – Highlighted edges of the digit to be analysed.

Figure 145 shows that the seventh digit was read correctly even if it was only partially visible.

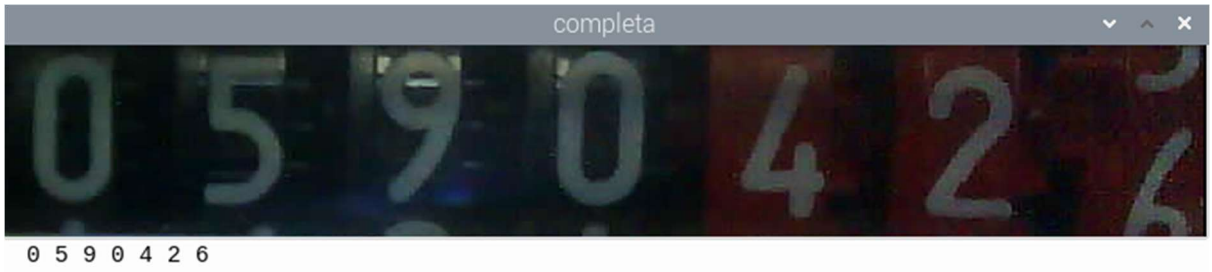


Figure 145 – Correct reading of the rotating digit.

5.4 Incorrect reading of the first digits, resolved by the algorithm

Figure 146 is a real image acquired by the system. Tesseract, in this case, could clearly read 0590449, even if the seventh digit was only partially visible.

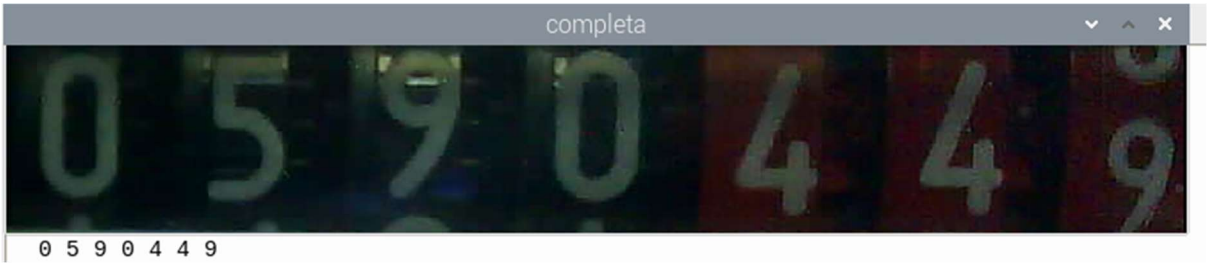


Figure 146 – Correct reading of the value 0590449.

Figure 147 is the image that the system acquired immediately after that in Figure 146. In this case Tesseract was not able to read correctly. It, in fact, read 0590420.

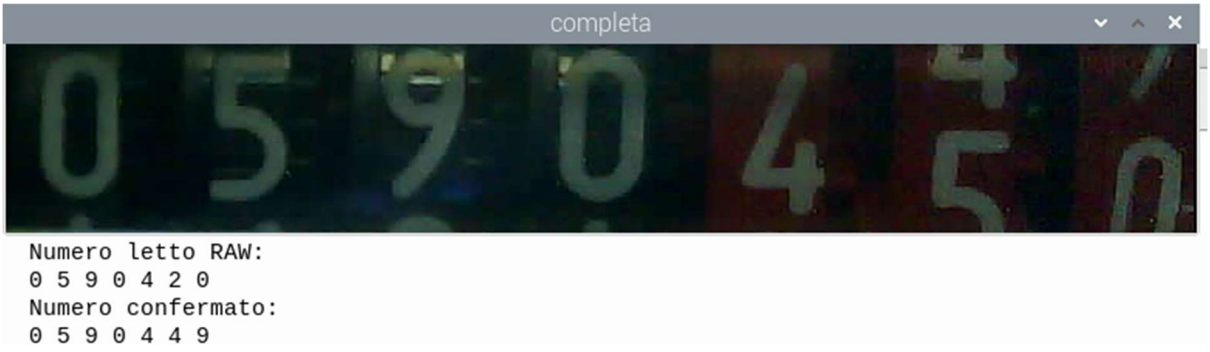


Figure 147 – Algorithm intervention to correct reading error.

Since the sixth digit cannot go from 4 to 2, then both the sixth and the seventh digits remained as they were previously. In this case, thanks to the intervention of the developed algorithm, the confirmed number remained 0590449.

Figure 148 is a photo taken by the system immediately after that in Figure 147. Here the sixth digit was clearly visible, therefore, in this situation, the reading was correctly done as 0590451.

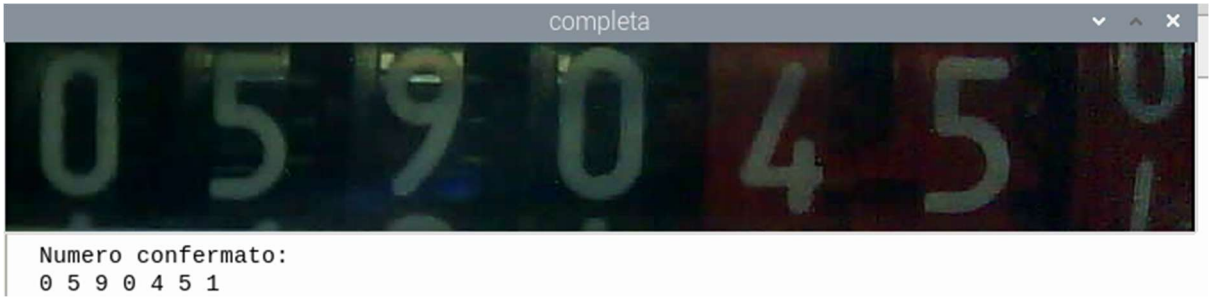


Figure 148 – Correct reading of the value 0590451.

In these three consecutive images the numbers actually followed one another as 49 50 51 and the system read 49 49 51, which is a good result since, without the algorithm, the reading would have wrongly been 49 20 51.

5.5 Inserting a white frame around the RoI of the last digit

As already seen in the description of the Raspberry software, the additional pre-processing of the seventh digit involves adding a thin white frame around the RoI. In this subsection, it is demonstrated why such a border was necessary.

Figure 149 shows the seventh digit without the addition of the white border.

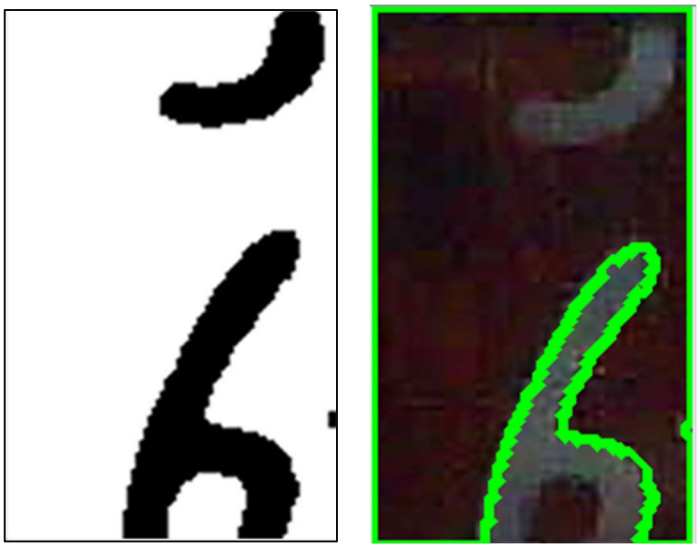


Figure 149 – Seventh digit without white border.

When OpenCV searches for edges contained within this image, the number 6 is considered part of the large edge surrounding the entire image. Figure 150, instead, represents the same digit, but with the addition of a white frame that delimits the content of the image from its outer edge.

Figure 150 – Seventh digit with white border.

In this case, when OpenCV searched for the edges, it located the highest edge in the outer edge of the entire Region of Interest, with the second highest edge being the number 6. This allowed the algorithm to correctly read the seventh digit.

5.6 Database

The values read by Tesseract were also saved in a local Database, as shown in Table 13.

Table 13 – Local Database.

Id=1	Date= 25/11/2024	08:18:23	Value= 590.422
Id=2	Date= 25/11/2024	08:18:38	Value= 590.422
…			
Id=98	Date= 25/11/2024	08:42:20	Value= 590.45
Id=99	Date= 25/11/2024	08:42:37	Value= 590.453
Id=100	Date= 25/11/2024	08:42:50	Value= 590.453

If, for any reason, the data cannot reach Akenza, it would still be present in the Database saved in the Raspberry, so it would still be possible to consult from the database. For each value read, the date and time were also saved.

5.7 Analysis of the collected data

Table 14 illustrates the comparison between the values read by the developed system and the actual values shown by the counter.

Table 14 – Read values, actual values and absolute errors (m^3).

Read n	Read Value	Actual Value	Error	Read n	Read Value	Real Value	Error	Read n	Read Value	Real Value	Error
1	590,422	590,422	0,000	31	590,449	590,449	0,000	61	590,474	590,476	-0,002
2	590,423	590,423	0,000	32	590,449	590,450	-0,001	62	590,478	590,476	0,002
3	590,424	590,424	0,000	33	590,451	590,451	0,000	63	590,477	590,477	0,000
4	590,425	590,425	0,000	34	590,45	590,451	-0,001	64	590,478	590,478	0,000
5	590,426	590,426	0,000	35	590,45	590,452	-0,002	65	590,478	590,479	-0,001
6	590,427	590,427	0,000	36	590,453	590,453	0,000	66	590,48	590,480	0,000
7	590,427	590,427	0,000	37	590,453	590,454	-0,001	67	590,481	590,481	0,000
8	590,428	590,428	0,000	38	590,455	590,455	0,000	68	590,482	590,482	0,000
9	590,429	590,429	0,000	39	590,456	590,456	0,000	69	590,483	590,483	0,000
10	590,438	590,430	0,008	40	590,457	590,457	0,000	70	590,484	590,484	0,000
11	590,431	590,431	0,000	41	590,458	590,458	0,000	71	590,484	590,484	0,000
12	590,432	590,432	0,000	42	590,45	590,459	-0,009	72	590,487	590,485	0,002
13	590,433	590,433	0,000	43	590,45	590,460	-0,010	73	590,486	590,486	0,000
14	590,434	590,434	0,000	44	590,46	590,460	0,000	74	590,487	590,487	0,000
15	590,435	590,435	0,000	45	590,461	590,461	0,000	75	590,488	590,488	0,000
16	590,434	590,436	-0,002	46	590,464	590,462	0,002	76	590,489	590,489	0,000
17	590,438	590,437	0,001	47	590,463	590,463	0,000	77	590,489	590,490	-0,001
18	590,437	590,437	0,000	48	590,464	590,464	0,000	78	590,491	590,491	0,000
19	590,438	590,438	0,000	49	590,465	590,465	0,000	79	590,491	590,492	-0,001
20	590,439	590,439	0,000	50	590,466	590,466	0,000	80	590,494	590,493	0,001
21	590,43	590,440	-0,010	51	590,467	590,467	0,000	81	590,493	590,493	0,000
22	590,441	590,441	0,000	52	590,467	590,467	0,000	82	590,494	590,494	0,000
23	590,449	590,442	0,007	53	590,468	590,468	0,000	83	590,494	590,495	-0,001
24	590,443	590,443	0,000	54	590,468	590,469	-0,001	84	590,496	590,496	0,000
25	590,443	590,443	0,000	55	590,47	590,470	0,000	85	590,497	590,497	0,000
26	590,444	590,444	0,000	56	590,471	590,471	0,000	86	590,497	590,498	-0,001
27	590,445	590,445	0,000	57	590,474	590,472	0,002	87	590,498	590,499	-0,001
28	590,446	590,446	0,000	58	590,473	590,473	0,000	88	590,498	590,500	-0,002
29	590,447	590,447	0,000	59	590,474	590,474	0,000	89	590,5	590,500	0,000
30	590,448	590,448	0,000	60	590,475	590,475	0,000	90	590,5	590,500	0,000

The Absolute error, for each individual measurement, is given by:

$$Absolute\ Error = Read\ Value - Actual\ Value$$

Table 14 shows that the first six most significant digits were almost always correct.

The Relative error, for each individual measurement, is given by:

$$Relative\ Error = \frac{Absolute\ Error}{Actual\ Value} \cdot 100$$

Table 15 shows the value of the relative error in the two cases where the absolute error was maximum.

Table 15 – Relative error.

Read n	Read Value	Actual Value	Absolute Error	Relative Error
21	590,43	590,440	-0,010	- 0,00169 %
43	590,45	590,460	-0,010	- 0,00169 %

The maximum Absolute error observed is -0.01 m^3, but an error on the second decimal place occurred only 2 times out of 89 different images and corresponds to a Relative error of only -0.00169%. The third decimal place, however, is not predictable in any way, therefore it was possible to read it only thanks to a good training.

5.8 Water leakage detection

A second experiment was carried out, this time with the aim of detecting a water leak. As shown in Figure 151, the shower faucet has a constant water leak.

Figure 151 – Constant water leak of the shower faucet.

In this case, the water is clearly visible, but a similar situation could occur with water flowing inside the wall, for example, and therefore not easily detectable. To do this experiment, the device was on for X hours while no one in the house used water. On Akenza it can be seen that in these hours, which should have been without any consumption, there is instead a constant consumption (Figure 152).

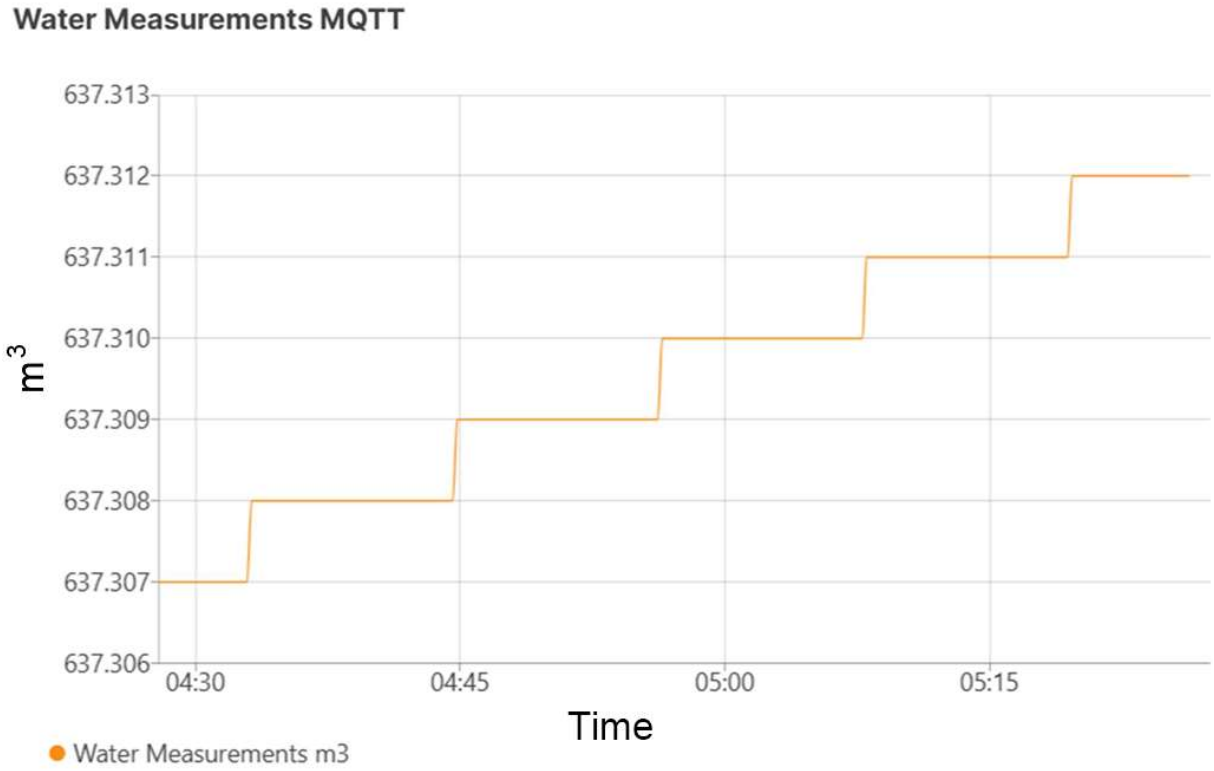


Figure 152 – Water Measurements showed on Akenza.

The images in Figure 153 show the first and last images acquired.

Figure 153 – Consumption due to the loss of water.

In the first image, taken at 04h:26m, the water meter indicates 637,307 m^3. In the second image, taken at 05h:26m, the water meter indicates 637,312 m^3. In 60 minutes, there was a consumption of 0,005 m^3. Obviously, this abnormal consumption is due to the loss of water. This problem induces a loss of 0,12 m^3 of

water per day and 3,6 m^3 per month. According to the World Health Organization (WHO), between 50 and 100 litres (0,05 and 0,1 m^3) of water per person per day are needed to ensure that most basic needs are met, and few health concerns arise [90]. This means that every day, because of this small loss, the amount of water two people need to live is wasted.

Figure 154 shows the tariff schedule for the year 2025 of the water supplier in the city of Coimbra, Águas de Coimbra [91].

Tarifa Variável	Euros/m³ (1000litros)/30 dias
0 a 5m³ (até 5000 litros)	0,5952 €
>5m³ (>5000 litros) a 15m³ (15000 litros)	0,8765 €

Figure 154 – TARIFÁRIO para 2025, Águas de Coimbra [91].

So this small loss of water also leads to a monthly economic loss of between 2,14 € and 3,15 €.

This chapter showed the final experiment, during which the developed device was used in a real situation. Afterwards, special situations that the developed software had to face were also analysed and finally a real water leakage was detected by using the developed system.

6 CONCLUSIONS AND FUTURE WORK

The goal of this project was to build a device and an IoT application capable of turning any water meter into an AMR meter. This was done without replacing the already installed meter. Instead, a device capable of acquiring images of the meter and of extracting the value displayed by the meter was developed.

The camera module of this project was the ESP-EYE, a camera with a very low price, capable of acquiring images up to a resolution of 1600x1200 pixels and equipped with Bluetooth, to be able to send images to the computing unit.

The computing unit for this project was a Raspberry Pi, which is a very low-priced computer. It follows that, with a negligible sum, it is possible to purchase the components to turn any water meter into an AMR meter, without replacing it, which would require a much higher sum. Furthermore, it is possible to use a single Raspberry for many meters, for example in the case of an apartment building. This would even reduce the cost further.

The values read by the device were very similar to the real values. The only few errors were made on the third decimal place, while the other digits were almost always read correctly. This happened due to a good training of Tesseract, which is the OCR used in this project, but also thanks to the algorithms implemented.

The first algorithm takes into account the possible variation that may exist between two subsequent images and is able to filter out incorrect values.

The second algorithm, exclusively developed for the less significant digit, which is the most unpredictable, was able to remove the too small fragments of the rotating digits from the image and to ensure that Tesseract's focus was entirely on the most visible digits.

Finally, by using Akenza it was possible to graphically display the consumption, and the values read by the device. The data is sent to the external server in three different ways, based on the position of the meters:

• Through Wi-Fi, for meters in the range of a Wi-Fi network;

• Through LoRaWAN, for meters out of the range of a Wi-Fi network;

• A hybrid between the two, which includes the Wi-Fi network as the main one and the LoRAWAN network as a substitute in case of problems with the first.

The developed system was trained to recognise the characters used by only one type of water meter. A subsequent development could be to train Tesseract in order to also recognise the character types used by other water meters. Furthermore, the same strategy can be used to read analogue meters that are different from water meters, for example gas or electricity meters.

BIBLIOGRAPHY

[1] N. Pimenta e P. Chaves, «Study and design of a retrofitted smart water meter solution with energy harvesting integration,» *Discover Internet of Things,* vol. 1, n. 1, Mar. 2021.

[2] X. J. Li e P. H. Chong, «Design and implementation of a self-powered smart water meter,» *Sensors,* vol. 19, n. 19, p. 4177, Sep. 2019.

[3] Omega, «Lessons about Positive Displacement Flow Meters,» [Online]. Available: Coursera.org/articles/lstm-neural-network?utm_medium=sem&utm_source=gg&utm_campaign=B2C_EMEA__coursera_FTCOF_career-academy_pmax-multiple-audiences-country-multi-set2&campaignid=20882109092&adgroupid=&device=c&keyword=&matchtype=&network=x&devicemodel=&ad.

[4] Realpars, [Online]. Available: https://www.realpars.com/blog/turbine-flow-meter.

[5] intellimeter, «Difference between Single-Jet and Multi-Jet Water Meters,» 02 October 2019. [Online]. Available: https://blog.intellimeter.com/difference-between-single-jet-and-multi-jet-water-meters.

[6] McCrometer, «Electromagnetic flow meters,» [Online]. Available: https://www.mccrometer.com/electromagnetic-flow-meters-family-page.

[7] Sierra, «How an Ultrasonic Flow Meter Works,» 5 December 2017. [Online]. Available: https://www.sierrainstruments.com/blog/?ultrasonic-flow-meter-works.

[8] yu Energy, «Smart VS AMR Meters: What Is The Difference?,» 6 February 2019. [Online]. Available: https://www.yuenergy.co.uk/smart-vs-amr-meters-what-is-the-difference/.

[9] M. Crainic, «OVERVIEW OF THE CURRENT STATE OF THE ART IN THE DOMAIN OF DOMESTIC WATER METERS PART II WATER METERS FOR SMART METERING SYSTEMS,» in *Installations for Buildings and Ambiental Comfort Conference XX- edition*, Timisoara - ROMANIA, April 2011.

[10] P. Satish, A. Raghul , B. Srinivas e N. Sujaudeen , «Automated Meter Reading System - A Study,» *International Journal of Computer Applications,* vol. 116, n. 18, pp. 39-46, April 2015.

[11] EJP, «Water Meter Solutions,» [Online]. Available: https://www.ejprescott.com/products/water-meter-solutions#undefined1.

[12] y. Energy, «Smart VS AMR Meters: What Is The Difference?,» 6 Feb. 2019. [Online]. Available: https://www.yuenergy.co.uk/smart-vs-amr-meters-what-is-the-difference/.

[13] A.-W. Zainab e M. O. Agyeman, «Challenges and Opportunities of SmartMeters in Smart Homes and Smart Grid,» *Proceedings of the 2nd International Symposium on Computer Science and Intelligent Control,* pp. 1-6, September 2018.

[14] M-Bus, «Wired M-Bus Documentation,» [Online]. Available: https://m-bus.com/documentation.

[15] M-Bus, «Overview of the M-Bus,» [Online]. Available: https://m-bus.com/documentation-wired/03-overview-of-the-m-bus.

[16] M-Bus, «Wired M-Bus Specification - Physical Layer,» [Online]. Available: https://m-bus.com/documentation-wired/04-physical-layer.

[17] H. Sheikh, C. Prins e E. Schrijvers, «Artificial Intelligence: Definition and background,» *Research for Policy,* p. 15–41, 2023.

[18] A. Azmir, «What is Machine Learning?,» 2023.

[19] I. H. Sarker, «Machine learning: Algorithms, real-world applications and Research Directions,» *SN Computer Science,* vol. 2, n. 3, Mar. 2021.

[20] K. W. K. S. K. a. Y. C. Y. S.-H. Han, «Artificial Neural Network: Understanding the basic concepts without mathematics,» *Dementia and Neurocognitive Disorders,* vol. 17, n. 3, p. 83, 2018.

[21] Z. Fountas, «The structure of a real neuron,» [Online]. Available: https://www.researchgate.net/figure/The-structure-of-a-real-neuron_fig1_266485234.

[22] D. Simant, «Expressive Power and Loss Surfaces of Deep Learning Models,» 2021.

[23] L. Aguiar, L. Reis e F. Morgado-Dias, «Neuron Implementation using System Generator,» in *9th Portuguese Conference on Automatic Control*, Coimbra, January 2010.

[24] Y. Liao, D. Ebler, F. Liu e O. Dahlsten, «Quantum Speed-up in global optimization of binary neural nets,» *New Journal of Physics,* vol. 23, n. 6, p. 063013, Jun. 2021.

[25] L. Benoit, M. Sarat e N. Yoni, «The Mathematical Engineering of Deep Learning,» 2021. [Online]. Available: https://deeplearningmath.org/general-fully-connected-neural-networks.html.

[26] L. Zachary, «A Critical Review of Recurrent Neural Networks for Sequence Learning,» 2015.

[27] J. Kumar, R. Goomer e A. K. Singh, «Long short term memory recurrent neural network (LSTM-RNN) based workload forecasting model for cloud datacenters,» *Procedia Computer Science,* vol. 125, p. 676–682.

[28] A. Timalsina, «How to extract data using TESSERACT OCR?,» [Online]. Available: https://www.docsumo.com/blog/tesseract-ocr .

[29] J. Mungalpara, «What is LSTM , peephole LSTM and gru?,» Medium, [Online]. Available: https://medium.com/nerd-for-tech/what-is-lstm-peephole-lstm-and-gru-77470d84954b.

[30] Y. Perwej, . S. A. Hannan, A. Asif e A. Mane, «An Overview and Applications of Optical Character Recognition,» *International Journal of Advance Research In Science And Engineering (IJARSE),* vol. 3, pp. 261-274, 2014.

[31] R. Smith, «An Overview of the Tesseract OCR Engine Ray Smith Google Inc,» *GoogleInc..*

[32] R. Smith, «An overview of the tesseract OCR engine,» *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007),* vol. 2, pp. 629-633, September 2007.

[33] [Online]. Available: https://www.docsumo.com/blog/tesseract-ocr.

[34] Anishnama, «Understanding Bidirectional LSTM for Sequential Data Processing,» Medium, 18 May 2023. [Online]. Available: https://medium.com/@anishnama20/understanding-bidirectional-lstm-for-sequential-data-processing-b83d6283befc#:~:text=Output%3A%20The%20output%20of%20the,or%20applying%20a%20different%20transformation..

[35] [Online]. Available: https://github.com/tesseract-ocr/docs/blob/main/das_tutorial2016/6ModernizationEfforts.pdf.

[36] G. Furnieles, «Sigmoid and SoftMax Functions in 5 minutes,» Medium, 8 September 2022. [Online]. Available: https://towardsdatascience.com/sigmoid-and-softmax-functions-in-5-minutes-f516c80ea1f9.

[37] OpenCV, «About,» [Online]. Available: https://opencv.org/about/.

[38] S. Kumar, P. Tiwari e M. Zymbler, «Internet of Things is a revolutionary approach for future technology enhancement: a review,» *J Big Data,* vol. 6, p. 111, 2019.

[39] D. Schoder, «Introduction to the internet of things,» *Internet of Things A to Z,* p. 1–50, May 2018.

[40] A. Rghioui e O. Abdelmajid , «Internet of Things: Surveys for Measuring Human Activities from Everywhere,» *International Journal of Electrical and Computer Engineering (IJECE),* vol. 7, n. 5, p. International Journal of Electrical and Computer Engineering (IJECE), October 2017.

[41] A. Calihman, «IOT architectures - common approaches and ways to design IOT at scale,» NetBurner, Inc, [Online]. Available: https://www.netburner.com/learn/architectural-frameworks-in-the-iot-civilization/ .

[42] XiaochaoGONG, «ESP-EYE Getting Started Guide,» Github, [Online]. Available: https://github.com/espressif/esp-who/blob/master/docs/en/get-started/ESP-EYE_Getting_Started_Guide.md.

[43] K. Söderby e J. Hylén, «Getting Started with Arduino IDE 2,» 28 10 2024. [Online]. Available: https://docs.arduino.cc/software/ide-v2/tutorials/getting-started-ide-v2/#overview..

[44] Raspberry Pi, «Raspberry Pi 4,» [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/.

[45] Raspberry Pi Trading Ltd, «Raspberry Pi 4 Computer Model B,» June 2019. [Online]. Available: https://docs.rs-online.com/b1fb/0900766b816fa153.pdf.

[46] «Raspberry Pi Documentation,» [Online]. Available: https://www.raspberrypi.com/documentation/computers/raspberry-pi.html.

[47] G. Bernardo, «La tecnologia LoRa – La scheda di sviluppo TTGO LoRa32 – Parte 1 – introduzione,» SETTOREZERO, [Online]. Available: https://www.settorezero.com/wordpress/lora_lorawan_lilygo_ttgo_lora32_esp32/.

[48] LILYGO, «LoRa32 V2.1_1.6,» [Online]. Available: https://lilygo.cc/products/lora3?srsltid=AfmBOooyHxynFJVjn-Jh85p7wLYcpwJ9hYKpgbHxy_EZzKYRdpxxuc1-.

[49] ESPRESSIF, «ESP32 Series Datasheet Version 4.7,» [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf.

[50] M. Verma , S. Singh e B. Kaur , «An Overview of Bluetooth Technology and its Communication Applications,» *International Journal of Current Engineering and Technology,* vol. 5, pp. 1588-1592, 2015.

[51] C. Bisdikian, «An Overview of the,» 2001. [Online]. Available: https://people.cs.vt.edu/~hamid/Mobile_Computing/papers/bisdikian01bluetooth.pdf.

[52] APFEN, «Comprehensive Guide to Identifying USB Connector Port Types,» 6 September 2023. [Online]. Available: https://www.szapphone.com/blog/usb-connector-types-guide/.

[53] Mockett, «Leading The Charge - USB Type C,» [Online]. Available: https://www.mockett.com/blog/blog-2022-what-is-usb-c.html.

[54] cablexpert, «USB 2.0 extension cable, 10 m, black,» [Online]. Available: https://cablexpert.com/item.aspx?id=8369.

[55] H. Siebeneicher, «Universal Asynchronous Receiver-Transmitter (UART),» 28 10 2024. [Online]. Available: https://docs.arduino.cc/learn/communication/uart/#basic-print-example..

[56] H. H. S. R, «UART-Universal Asynchronous Receiver Transmitter,» 21 05 2023. [Online]. Available: https://www.linkedin.com/pulse/uart-universal-asynchronous-receiver-transmitter-hari-hara-sudhan-r.

[57] A. Tanenbaum, Reti Di Calcolatori, Milano: Pearson, 2023.

[58] geeksforgeeks, «TCP/IP Model,» [Online]. Available: https://www.geeksforgeeks.org/tcp-ip-model/.

[59] ZTE Corporation, «Wi-Fi 6 Technology and,» 2020. [Online]. Available: https://www.zte.com.cn/content/dam/zte-site/res-www-zte-com-cn/mediares/zte/files/pdf/white_book/Wi-Fi_6_Technology_and_Evolution_White_Paper-202009232125.pdf.

[60] B. Havemeier, «Wi-Fi 6: Benefits Beyond Blazing Speed! - The Future of Connectivity!,» CEL, [Online]. Available: https://www.cel.com/blog/wifi-6-benefits/.

[61] Random Nerd Tutorials, «What is MQTT and How It Works,» 16 December 2021. [Online]. Available: https://randomnerdtutorials.com/what-is-mqtt-and-how-it-works/.

[62] Toma, Alexandru, M. Popa e A. Zamfiroiu, «IoT Solution for Smart Cities' Pollution Monitoring and the Security Challenges,» *Sensors,* vol. 19, p. 3401, 2019.

[63] DevAcademy, «MQTT protocol,» [Online]. Available: https://academy.nordicsemi.com/courses/wi-fi-fundamentals/lessons/lesson-4-wifi-fundamentals/topic/mqtt-protocol/.

[64] Semtech, «LoRa (PHY),» [Online]. Available: https://www.semtech.com/lora/what-is-lora.

[65] Sghoslya, «All About LoRa and LoRaWAN,» [Online]. Available: https://www.sghoslya.com/.

[66] Science Direct, «Spread Spectrum,» [Online]. Available: https://www.sciencedirect.com/topics/computer-science/spread-spectrum#:~:text=Spread%20Spectrum%20refers%20to%20a,required%20to%20transmit%20the%20message.

[67] T. Tulka, «LoRa Spreading Factor Explained,» 22 07 2022. [Online]. Available: https://blog.ttulka.com/lora-spreading-factor-explained/.

[68] D. Huang, «Guest Blog: Wi-Fi, LoRaWAN, and IoT Convergenc,» [Online]. Available: https://wballiance.com/guest-blog-wi-fi-lorawan-and-iot-convergence/.

[69] Semtech, «Benefits of LoRa,» [Online]. Available: https://www.semtech.com/lora/why-lora.

[70] LoRa Alliance, «RP002-1.0.3 LoRaWAN® Regional Parameters,» [Online]. Available: https://lora-alliance.org/wp-content/uploads/2021/05/RP002-1.0.3-FINAL-1.pdf.

[71] B. Reynders e S. Pollin, «Chirp Spread Spectrum as a Modulation Technique,» *2016 Symposium on Communications and Vehicular Technologies (SCVT),* Nov. 2016.

[72] «LoRa Physical Layer Packet Format,» The Things Network, [Online]. Available: https://www.thethingsnetwork.org/docs/lorawan/lora-phy-format/.

[73] V. T. e. al., «Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies,» *Lora Backscatter,* vol. 1, n. 3, pp. 1-24, Sep. 2017.

[74] Semtech, «LoRaWAN® Standard,» [Online]. Available: https://www.semtech.com/lora/lorawan-standard.

[75] M. L. Liya e M. Aswathy, «Lora Technology for internet of things(iot):a brief Survey,» *2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC),* Oct. 2020.

[76] J. Santa, R. Sanchez-Iborra, P. Rodriguez-Rey, L. Bernal-Escobedo e A. F. Skarmeta, «LPWAN-based vehicular monitoring platform with a generic IP network interface,» *Sensors,* vol. 19, n. 2, p. 264, Jan. 2019.

[77] LoRa Alliance Technical Committee, «LoRaWAN™ 1.1 Specification,» 11 10 2017. [Online]. Available: https://net868.ru/assets/pdf/LoRaWAN-v1.1.pdf.

[78] LoRa-Alliance, «LoRaWAN Architecture,» [Online]. Available: https://lora-alliance.org/about-lorawan/.

[79] LoRa Alliance Technical Committee, «LoRaWAN® Backend Interfaces,» 19 10 2020. [Online]. Available: https://lora-alliance.org/wp-content/uploads/2020/11/TS002-1.1.0_LoRaWAN_Backend_Interfaces.pdf.

[80] The Things Industries, «End Device Activation,» [Online]. Available: https://www.thethingsnetwork.org/docs/lorawan/end-device-activation/.

[81] The Things Industries, «Architecture,» [Online]. Available: https://www.thethingsindustries.com/docs/the-things-stack/concepts/architecture/.

[82] M. Guay, «What are webhooks?,» Zapier, [Online]. Available: https://zapier.com/blog/what-are-webhooks/.

[83] E. Rescorla, «The Transport Layer Security (TLS) Protocol Version 1.3,» Internet Engineering Task Force (IETF) , August 2018. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc8446#page-6..

[84] akenza.io, «Overview,» [Online]. Available: https://docs.akenza.io/akenza.io. .

[85] akenza.io, «Data flows,» [Online]. Available: https://docs.akenza.io/akenza.io/get-started/your-data-flow. .

[86] akenza.io, «MQTT,» [Online]. Available: https://docs.akenza.io/akenza.io/api-reference/mqtt..

[87] akenza.io, «The things network,» [Online]. Available: https://docs.akenza.io/akenza.io/get-started/your-integration/the-things-network. .

[88] akenza.io, «Dashboard Builder,» [Online]. Available: https://docs.akenza.io/akenza.io/get-started/dashboard-builder. .

[89] myDevices, «Cayenne Low Power Payload,» [Online]. Available: https://docs.mydevices.com/docs/lorawan/cayenne-lpp.

[90] U. Nations, «The Human Right,» [Online]. Available: https://www.un.org/waterforlifedecade/pdf/human_right_to_water_and_sanitation_media_brief.pdf.

[91] Á. d. Coimbra, «TARIFÁRIO para 2025,» [Online]. Available: https://www.aguasdecoimbra.pt/wp-content/uploads/Tarifario_2025_AA.pdf.

[92] K. Yasar, «Transmission Control Protocol (TCP),» [Online]. Available: https://www.techtarget.com/searchnetworking/definition/TCP#:~:text=To%20prevent%20sending%20too%20much,their%20order%20and%20sequence%20number..

[93] «(PDF) a quantitative analysis of a novel network-based intrusion detection system over Raspberry Pi,» [Online]. Available: https://www.researchgate.net/publication/336825947_A_quantitative_analysis_of_a_novel_Network-based_Intrusion_Detection_System_over_Raspberry_Pi.

[94] [Online]. Available: https://mm.digikey.com/Volume0/opasdata/d220001/medias/images/5062/MFG_SC0192%289%29.jpg.

[95] «Raspberry Pi: third best-selling computer of all time and the making of a global community,» University of Cambridge, 3 August 2021. [Online]. Available: https://impactmap.cam.ac.uk/raspberry-pi-third-best-selling-computer-of-all-time-and-the-making-of-a-global-community/.

[96] «TTGO ESP32-Paxcounter LoRa32 V2.1 - 868Mhz,» OPENCIRCUIT, [Online]. Available: https://opencircuit.pt/product/ttgo-esp32-paxcounter-lora32-v2.1-868mhz.

[97] Semtech, «A Brief History of LoRa®: Three Inventors Share Their Personal Story at The Things Conference,» 08 January 2020. [Online]. Available: https://blog.semtech.com/a-brief-history-of-lora-three-inventors-share-their-personal-story-at-the-things-conference.

[98] The Thingd Network, «What are LoRa and LoRaWAN?,» [Online]. Available: https://www.thethingsnetwork.org/docs/lorawan/what-is-lorawan/#lora.

[99] akenza.io, «Rule logic - comparison,» [Online]. Available: https://docs.akenza.io/akenza.io/get-started/rule-engine/rule-logic. .

[100] akenza.io, «Rules,» [Online]. Available: https://docs.akenza.io/akenza.io/get-started/rule-engine/rule-logic. .

[101] Elettronica Conduttori, «Data Cables, Cavo BUS USB,» [Online]. Available: https://www.elettronicaconduttori.com/prodotti/cavo-usb-automotive-per-sistema-infotainment-2/.

[102] W. Siringoringo, «Teaching and Learning Wi-Fi,» *Selected Readings on,* pp. 22-40.

[103] Wi-Fi Alliance, «Who We Are, History,» [Online]. Available: https://www.wi-fi.org/who-we-are/history.

[104] C. Doctorow, «WiFi isn't short for "Wireless Fidelity",» Boing Boing, [Online]. Available: https://boingboing.net/2005/11/08/wifi-isnt-short-for.html.

[105] I. Fatin, «A review of Bluetooth Technology,» 2023.

Instituto Superior
de Engenharia
Politécnico de Coimbra