INSTITUTO
UNIVERSITÁRIO
DE LISBOA

Machine Learning for Precise Water Leaks Detection

João Miguel de Jesus Alves Coelho

Master in Telecommunications and Computer Engineering

Supervisor:
Prof. Doctor Pedro Joaquim Amaro Sebastião, Assistant Professor,
ISCTE-IUL

Co-Supervisor:
Master André Filipe Xavier da Glória, Researcher at Instituto de Telecomunicações,
ISCTE-IUL

November, 2020

Acknowledgment

First and foremost, I would like to thank my parents, Miguel and Maria, and my sister Rita, for all the support and motivation through the years and always encouraged me for following my goals.

To André Glória, the biggest thank you for all the support and patience in this last months, helping me overcome the various problems in the development in this dissertation. For that I am forever gratefull.

To Professor Pedro Sebastião, for the orientation and motivation given during the development of the dissertation.

I would also like to thanks to my colleagues, who accompanied me on this journey and especially to Maria Inês Pires for the love, support and motivation during the process of development of this dissertation.

Resumo

A Internet das Coisas surgiu para revolucionar o mundo tecnológico e o nosso dia a dia. Com base na capacidade de conexão de dispositivos, capazes de controlar e monitorizar ambientes inteligentes de modo a reduzir a acção humana, estes dispositivos, sendo de baixo consumo permitem tornar os sistemas mais ecológicos e amigos do ambiente, reduzindo os gastos de energia e a pegada de carbono, mas nunca perdendo a capacidade.

Tendo em conta a má gestão e a cada vez maior escassez de recursos, existe uma maior preocupação em acompanhar os sistemas de água de modo a ser possível detectar e localizar fugas de água com a maior brevidade possível para que o desperdício seja o menor possível.

Esta dissertação apresenta uma proposta para um sistema baseado numa rede de sensores sem fios, desenhada para monitorizar sistemas de distribuição de água, como por exemplo sistemas de irrigação, que com a ajuda de um algoritmo de Aprendizagem Automática permite localizar com precisão o local onde se deu a fuga de água. De modo a obter um sistema capaz e de baixo custo, foi feita uma análise a diversos módulos de software e hardware para que o sistema, através de uma aplicação móvel Android, permita ao utilizador visualizar informação, recolhida pelos sensores, e consequentemente, tratar de forma activa o problema da fuga.

A principal vantagem deste sistema e que o distingue de outros é o algortimo de Aprendizagem Automática, que através da informação que os sensores recolhe, vai aprendendo com o sistema e a qualquer variação de valores, alerta o utilizador para onde se encontra o local da fuga.

Palavras Chave: Internet of Things, Tecnologias Verdes, Água, Aprendizagem Automática, Sustentabilidade, Fugas de Água, Eficiência, Deteção, Localização.

Abstract

Internet of Things emerged to revolutionize the technological world and our daily lives. Based on the ability to connect devices, capable of controlling and monitoring intelligent environments in order to reduce human action, these devices, being low consumption, make systems more environmentally friendly, reducing energy costs and footprint carbon, but never putting in jeopardy the capability.

Bearing in mind the poor management and the increasing scarcity of resources, there is a greater concern to monitor water systems in order to be able to detect and locate water leaks as soon as possible so that the waste is as small as possible.

This dissertation presents a proposal for a system based on a wireless sensor network, designed to monitor water distribution systems, such as irrigation systems, which with the help of an Automatic Learning algorithm allows to precisely locate the place where gave the water leak. In order to obtain a capable and low-cost system, an analysis was made of several software and hardware modules so that the system, through an Android mobile application, allows the user to view information, collected by the sensors, and, consequently, deal with active way the problem of escape.

The main advantage of this system and that distinguishes it from others is the Automatic Learning algorithm, which through the information that the sensors collect, learns from the system and any variation of values, alerts the user to where the leak is located.

Keywords: Internet of Things, Green Tech, Water, Machine Learning, Sustainability, Water Leaks, Efficiency, Detection, Location.

Contents

Acknowledgment i

Resumo iii

Abstract v

List of Figures ix

List of Tables xi

List of Acronyms xiii

Chapter 1. Introduction 1

1.1. Motivation and Framework 1

1.2. Goals 2

1.3. Main Contributions 2

1.4. Structure of the Dissertation 2

Chapter 2. State of the Art 5

2.1. Internet of Things 5

2.2. Wireless Sensor Network 6

2.2.1. Sensing 7

2.2.2. Communications 9

2.2.3. Data Analysis 15

2.3. Mobile Application 19

2.4. Related Work 20

Chapter 3. System Architecture 23

3.1. Data Collection 25

3.1.1. Aggregation Node 25

3.1.2. Sensor Node 26

3.1.3. Communications 28

3.2. Data Analysis 29

3.2.1. Storage 29

3.2.2. Application Programming Interface 30

3.2.3. Machine Learning 31

3.3. Data Visualization 31

3.3.1. Application Features 32

3.3.2. Android Activities 32

Chapter 4. Machine Learning Training 35

4.1. Dataset Creation 35

4.2. Model Analysis 38

4.3. Remarks 40

Chapter 5. System Implementation 43

5.1. Laboratorial Tests 43

5.2. Experimental Implementation 46

5.3. Results & Discussion 49

Chapter 6. Conclusions and Future Work 51

6.1. Conclusions 51

6.2. Future Work 52

References 53

Appendices 59

Appendix A. Scientific Paper 59

Appendix B. User & Technical Manual 75

Appendix C. Detailed Image 89

List of Figures

2.1 The Elements of IoT 6

2.2 Wireless Sensor Network Architecture 7

2.3 Arduino Uno board 8

2.4 ESP32 Board 9

2.5 Topology of a Bluetooth Network 10

2.6 ZigBee Topologies 11

2.7 LoRaWAN Topology 12

2.8 Configuration of Wi-Fi Networks 13

2.9 MQTT - Publish/Subscribe Process 15

2.10Support Vector Machine Architecture 16

2.11Decision Trees Architecture 17

2.12Random Forest Architecture 18

2.13Neural Network Architecture 18

2.14Extreme Gradient Boosting architecture 19

3.1 System Implementation 23

3.2 System Architecture 24

3.3 System Logic 24

3.4 Agreggation Node Diagram 25

3.5 RFM95W Transceiver 26

3.6 Sensor Node Diagram 27

3.7 YF-B2 Water Flow Sensor 27

3.8 Database Relational Model 30

3.9 Mobile Application Icon 31

3.10Android Screens 32

4.1 System Prototype 35

4.2 Clustered Dataset 38

4.3 Standard Dataset 38

4.4 Standard Dataset Feature Analysis 39

4.5 Clustered Dataset Feature Analysis 39

4.6 Test Results 41

5.1 Sensor Node 43

5.2 HiveMQTT messages received 44

5.3 Stored values Sample 45

5.4 Jupyter Environment 45

5.5 Script Sample - Data Pre-Processing 46

5.6 Machine Learning Test Sample 47

5.7 Test Results 48

5.8 Experimental Implementation Values Sample 48

5.9 Data on Application 49

5.10Test Reults 50

List of Tables

2.1 Intra-node Communications Protocols Characteristics 10

2.2 Server Side Communications Protocols Characteristics 12

4.1 Intended Outputs 36

4.2 Standard Dataset 37

4.3 Clustered Dataset 37

4.4 Test Scenarios 40

4.5 Tests Results 40

5.1 Results Matrix Analysis 50

List of Acronyms

ADC	Analogic Digital Converters
AES	Advanced Encryption Standard
AP	Access Point
API	Application Programming Interface
BBS	Basic Service Set
BLE	Bluetooth Low Energy
CSS	Chirp Spread Spectrum
DT	Decision Trees
DSSS	Direct Sequence Spread Spectrum
FFD	Full Function Devices
FSK	Frequency-Shift Keying
HTTP	Hypertext Transfer Protocol
IBBS	Independent Basic Service Set
IDE	Integrated Development Environment
IoT	Internet of Things
IP	Internet Protocol address
LAN	Local Area Network
LR-WPAN	Low-Rate Wireless Personal Area Network
MAC	Media Access Control
ML	Machine Learning
MLP	Multilayer Perceptron
MQTT	Message Queue Telemetry Transport
NB-IoT	Narrowband IoT
NOOBS	New Out Of the Box Software
NN	Neural Network

OS	Operating System
PAN	Personal Area Network
RF	Random Forest
RFD	Reduce Function Devices
SD	Secure Digital
SoC	System on Chip
SVC	Linear Support Vector Clustering
SVM	Support Vector Machine
URL	Uniform Resource Locator
USA	United States of America
Wi-Fi	Wireless Fidelity
WLAN	Wireless Local Area Network
WPAN	Wireless Personal Area Network
WSN	Wireless Sensor Network
XGB	Extreme Gradient Boosting

CHAPTER 1

Introduction

1.1. Motivation and Framework

Nowadays, technology takes a major role in our life and we are always searching for better solutions to solve everyday problems. There is always a need for improvement to make it easy to do repetitive tasks, but with less effort.

With this evolution in technology and the increase of devices being used every day, the concept of the Internet of Things (IoT) arise [**1**]. IoT consists on a network of devices that are capable of being connect to the Internet and communicate between themselves to enact on a common objective. This technology aims for a fast exchange and update of data between devices to obtain an optimum performance [**2**].

When referring to IoT, it is important to also mentioned the idea of a Wireless Sensor Network (WSN), an area of the technological world that is essential for the development of intelligent systems. In recent times, IoT and WSN have been a reliable features on the development of monitoring and control systems [**3**]. The wherewithal of, as mentioned, monitoring and control, extends to a ever so growing range of applications, such as water management.

Water management depends on how well the use of water is maximize and the water losses minimized [**4**]. This type of management requires precise analysis which may be to complex for humans to perform correctly. Thus, the implementation of a sensor network is paramount to be able to execute those analysis with precision. To improve and optimize the daily management and control of possible leaks, comes up the proposed system, with the main goal of identifying leak points of water systems in an autonomous way.

With the increase in human population, sustainability and efficient water usage assumes an important role. As water is a scarce resource, detecting problems with the supply and distribution of water as fast as possible can be achieved throughout a sensor network, leading to minimal to no waste in activities such as agricultural irrigation.

1.2. Goals

This dissertation aims to create a system that can control and monitor water leaks. The system collects data through a low cost sensor network in order to evaluate possible leak points in the water system. The data obtained from the sensors is stored and treated by a Machine Learning algorithm that will allow the user to be notified if the water distribution system is starting to have leaks, the potential size and the location of those leaks. These will be presented to all users and can be controlled by them with administration permissions, such has farm owners or building maintenance, through a mobile app. When a lower water flow is detected, the user is notified and can act upon and take care of the water leak.

This work proposes a new system to monitor and control the water flow through a water distribution pipeline, having in mind the reduction of water wasted and a monetary saving by the final user.

1.3. Main Contributions

This dissertation has the following main contributions:

- Design and implementation of a WSN for the monitoring of a water distribution system;
- Study and training of a Machine Learning capable of predicting leaks in the pipeline;
- Development of a mobile application for data visualization and control.

The work obtain through the development of this dissertation, besides this document, resulted in the following article, submitted for publication:

- Alves Coelho J., Glória A. and Sebastião P., ”Precise Water Leaks Detection using Machine Learning and Real-Time Sensor Data” in IoT(ISSN 2624-831X) journal by MPDI.

1.4. Structure of the Dissertation

After this chapter, Chapter 2 presents the State of the Art that summarizes all the topics related to the development of this work, outlining important definitions and comparisons that were performed to ensure the best outcome for the system, such as software and hardware. It also reviews related works and describes the improvements of the proposed approach. Chapter 3 focus on the system architecture, with a more detailed description of the software, hardware and communication protocols used in the system and how they best fit in this solution. Chapter 4 focus on the Machine Learning, with a study and a comparison of different algorithms to understand which one will work better on the intended system. Chapter 5 presents the implementation in experimental and real environments and the corresponding discussion of the results. Lastly, Chapter 6 presents the conclusions obtained after the complete development of the system and future work that could be implemented through what was presented in this dissertation.

CHAPTER 2

State of the Art

In this chapter, we scrutiny the different types of approaches studied to create the proposed system. In section 2.1 we start by introducing the topic of IoT and how it is structured. In section 2.2 we define what a sensor network is and in what way we can make use of it. In section 2.2.1 all hardware components to build the system are presented. This section compares all the hardware with similar features, in order to choose the one that fits the system the best. Section 2.2.2 we analyze the communication protocols used in IoT, with finishing remarks regarding the most suitable protocol to use in our system. Section 2.2.3 we introduce the ML algorithms that will studied to see the on that fits the system the best. In section 2.3 we describe what is a mobile application and the operating system that will be used in the development of the mobile application for the system. Lastly, in section 2.4, we refer some related works already developed and on what our differs from them.

2.1. Internet of Things

Internet of Things was first proposed in 1997, by Kevin Ashton and it was presented as the next big thing that could revolutionize the world of technologies [**5**]. Since then, it has gone from just an idea to a reality and it allowed people to be connected anytime, everywhere. It allows sensors and devices to be connected harmoniously within a smart system that provides intelligent services to humans [**6**]. Nowadays, IoT is being fused to almost every service or product. From Post Offices to Smart Homes, IoT is improving efficiency and making jobs and activities easier that before where mostly done manually and could induce error.

With its focus on low power consumption, IoT is directly connected to sustainability. And because of this, its closely related to Green Tech. Green Technologies appear to create a more efficient way for energy consumption while reducing the carbon footprint and waste all in order to improve sustainability [**7**]. To achieve this and make technology greener, the hardware needs to consume less power but still be efficient and the software use better algorithms to achieve the intended goal using less resources [**6**].

To better understand IoT we can divide it in six different elements [**8**], as shown in Figure 2.1, specifically, Identification, Sensing, Communication, Computing, Services and Semantics.

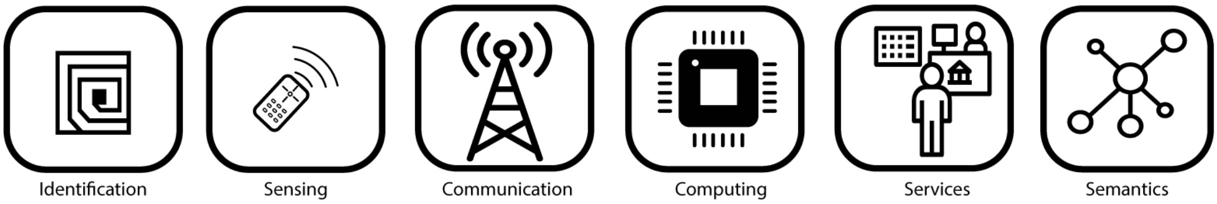


Figure 2.1. The Elements of IoT

This six elements together give us a better vision into the functionality and the real meaning of IoT.

Identification is pivotal in IoT to name and match services with there specific demand. Addressing IoT is important to differentiate between the object ID and its address.

Sensing is the capability of gathering data from the difference sensors that make the network and sending it back to the cloud or database to be analyzed. The data gathered may vary from smart sensors, such as motion-activated light bulbs, to wearable sensors, like heartbeat rate measure.

Communication connects the objects to exchange data and performs specifics types of services. for IoT, the most usual communications are made through Wireless communication protocols, but in some cases, Wired communication protocols are also used.

Computing represents the motor that makes everything run by using hardware, such as microcontrollers and microprocessors, and different software platforms to perform tasks.

Services is divided in four different tiers: Identity-Related services, Information Aggregation services, Collaborative-Aware services and Ubiquitous services.

Semantics represents the ability to round up the knowledge in an intelligent way to provide the required services. This knowledge can be recognition and analyzes of the data or modeling information.

2.2. Wireless Sensor Network

A Wireless Sensor Network is a network with sensors distributed through it, as seen in Figure 2.2, that gave the ability to collected, analyse and distribute the data to control certain circumstances, such as water flow or temperature. Whilom, this types of networks used to be wired and mostly intended to connect computers to the Internet [**9**]. Presently, a WSN is able to arrange several devices by using only wireless technologies.

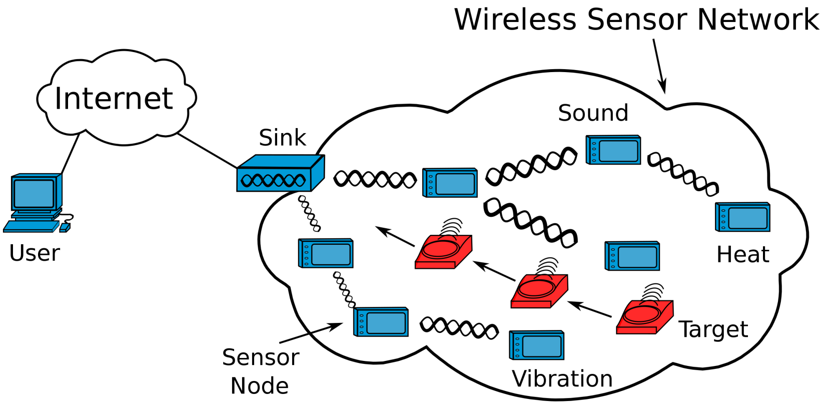


FIGURE 2.2. Wireless Sensor Network Architecture

2.2.1. Sensing

Sensor Networks are made up of different type of sensors, connected through multi-hop. In a Sensor Network there are three different types of sensing nodes [**10**].

Sensor node: its the lowest level of a Sensor Network. Just a sensor and communication mechanism and are used just to pass the data to another node in the network.

Data node: used to store data from the network but can also send data to another node. This nodes can be used to create autonomous sensor networks that record data for later to archive.

Aggregation node: these nodes use a communication device, a getaway and no sensors. They are used to gather data from one or more sensors and send it to the server.

Sensor Networks are no longer the expensive industrial constructs with big energy consumption that used to be. Nowadays, Sensor Networks are energy efficient, because the network can be located in zones where energy is a hard resource to get, and scalable, due to the high number of nodes the network can have.

A smart IoT system consists of several hardware components, like controlling platforms and radio modules that provide a communication between the nodes of the system and the sensors network.

The choice of hardware in an IoT system is crucial for its efficiency and to achieve the principal objective of the system. For this, it is important to consider the characteristics and compatibility of the modules and if these adhere to the communication protocols used by the system.

In this topic we will analyse which are the best hardware devices to create the desired IoT system.

2.2.1.1. *Controlling Platforms.* Controlling Platforms are electronic devices that function as the brain controlling the entire system on account of being able to retrieve, store and analyze data and also being able to incorporate a functional communication protocol scheme.

The most common solutions include microcontrolers and System on Chip (SoC) [**11**]. SoC are based on an electronic chip or integrated circuit that encompass the whole operating system. This innovation led to a decrease of production costs of computer systems and making them cheaper to buy for the final consumer.

Nowadays, there are several solutions for projects like this each one with its advantages and disadvantages. For the IoT system we intend to create its pivotal to chose a board that is simple but with a high performance and that can gather the information from the sensors as well as transmit, furthermore having the capacity to perform on different environments.

With this in mind, the boards that meet these requirements are the Arduino Uno and the ESP32.

Arduino - Is an open source computing platform based on a simple Input/Output board (Figure 2.3) and a development environment that uses Processing language [**12**]. Arduino has their own Integrated Development Environment (IDE) and the users can develop sketches using C++ or an already developed library.

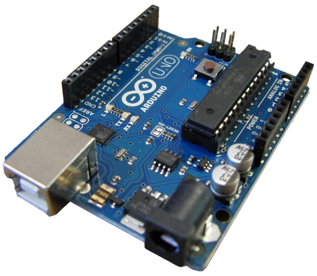


FIGURE 2.3. Arduino Uno board

There are multiple varieties, each with its own functionalities being Arduino UNO, showed in Figure 2.3 the most common. The board is composed by a ATMega328 microcontroller with 32 kB flash memory and an USB port that allows the charging and programming conferring it an extensive and felxible use.

The Arduino board is unique in its nature due to their cross-platform capability (Windows, MacOS and Linux), low cost, open source with extensible software and hardware.

Nonetheless it comes with some disadvantages such has the inability to run multiple tasks at once and having a low memory [**12**].

ESP32 - The ESP32 is a dual core chip (Figure 2.4) which includes Wi-Fi and BLE. It also comes with 32 GPIO ports, that can be assigned different programming functions, and 12 analog ports. This in combination with the dual core chip allows for this microcontroller to have an optimal performance [**13**].

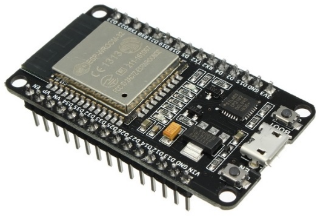


FIGURE 2.4. ESP32 Board

The reason that makes this microcontroller ideal for IoT systems is due to the low power capabilities and low cost. In its low power capability resides the ability to enter into a deep sleep state. This feature allows for the ESP32 to only be awoken when sending information, reducing its power consumption [**14**].

2.2.2. Communications

As mention in Section 2.1, communication is an important element of IoT. It is pivotal that the devices are able to trade data between them, so that IoT projects works as wanted.

The Internet is part of IoT, but that doesn't mean that all situations need to depend on it, so there are other types of protocols that can be used. With rise in the use of Sensor Networks and applications being used in the most varied types of environments, IoT uses diverse types of protocols to cope with the specific need of each situation.

IoT, like every new technology, wants to use the minimum space possible. One of the best methods to achieve this is wireless communication standards, because it can cover for all the necessities with the improvement of wireless low space environment. Wireless communications provide all that is needed to transfer data between devices and being a better solution to wired communications.

2.2.2.1. *Intra-node Communications.* Intra-node communication, as the name indicates, refers to the communication between sensor nodes within the WSN.

The most common intra-node communication protocols are Bluetooth and ZigBee but with new technologies, like LoRa and ESP NOW, starting to get more used.

In Table 2.1 [**15**] are some key features for intra-node communications protocols.

TABLE 2.1. Intra-node Communications Protocols Characteristics

Parameter	**ESPNOW**	**Blueetooth**	**ZigBee**	**LoRa**
Data rate [Mbps]	1	1	0.25	0.11
Frequency [GHz]	2.4	2.4	2.4	0.433/0.868
Range [m]	480	10 - 100	10 - 100	2000
Security	128bit-AES	128bit	128bit	128bit-AES
Nodes	32	7	65540	15000

LoRa - Stands for Long Range and is a physical layer that uses CSS modulation allowing for a longer communication range without a rise in terms of power consumption [**16**]. It uses a star topology which allows for a lower power consumption and a reduction in the network complexity and capacity.

Advantages of LoRa include long communication range, low power consumption and a 128-bit AES encryption provides a secure connection with the biggest disadvantages being its limited package size of only 555 bytes.

IEEE 802.15.1 (Bluetooth) - Firstly introduced in 1994 by Ericsson Mobile Communications, IEEE 802.15.1, mostly known by Bluetooth, is the standard for a wireless communication technique for short range radio system, formally known as Wireless Personal Area Network (WPAN) [**17**].

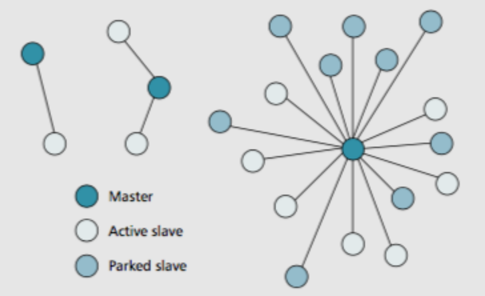


FIGURE 2.5. Topology of a Bluetooth Network [**17**]

IEEE 802.15.1 is currently in its version 5.0 and with each version having its advantages. The most important breakthrough occurred in version 4.0 or Bluetooth Low Energy (BLE), a short range radio with minimal power and with the ability to operate for long periods of time, when compared with other versions [**8**]. Its range is also an advantage, able to reach distances of up to 100 meter, which is 10 times bigger than older versions, making BLE a great tool for IoT.

Advantages of Bluetooth consist of one master node that controls all other nodes, a simple way of adding or removing nodes and being universally supported with the disadvantages maximum limit of 8 nodes, low communication range and only working with a star topology, seen in Figure 2.5.

IEEE 802.15.4 (ZigBee) - Made known in 2002, ZigBee is based in protocol 802.15.4 and was created for Low-Rate Wireless Personal Area Network (LR-WPAN). Due to its low power consumption, low data rate and low cost makes it one of the most used communication protocol in IoT. Also provides an high security, both in encryption and authentication services, can work with different platforms and can handle a large number of nodes (up to 65000 nodes) [**8, 18**].

This standard supports two network node types (Figure 2.6), Full Function Devices (FFD) and Reduce Function Devices (RFD). The FFD can operate as a Personal Area Network (PAN) coordinator, coordinator or as just a normal node in the network. A coordinator node is also responsible for the creation, maintenance and control of the network. The RFD are the simple nodes of the network and only have the ability to communicate with the coordinator [**8, 19**].

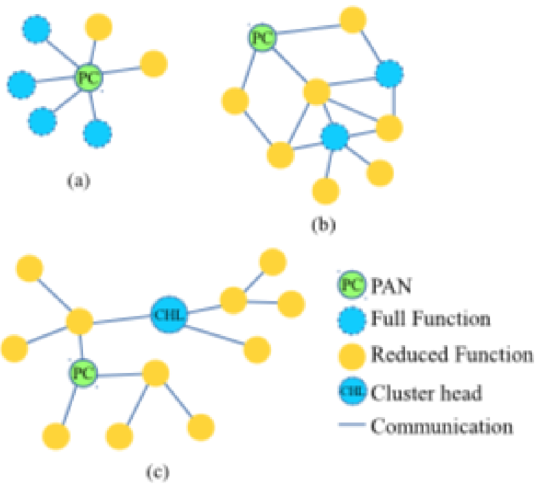


FIGURE 2.6. ZigBee Topologies: (a) Star, (b) Peer-to-Peer, (c) Cluster-tree [**8**]

As other protocols, ZigBee has advantages, such has support for many nodes, low power consumption and disadvantages like requiring additional hardware and not compatible with other communication protocols.

ESPNOW - Is a fast communication technology, developed by Espressif, with short transmission. It enables multiple devices to communicate between them, with a safe connection, without the need of Wi-Fi.

It has some similarities with Wi-Fi has it needs the devices to be paired before being able to communicate. which makes it ideal for communication between sensors within the same WSN [**20**].

As all communications protocols, ESPNOW comes with some advantages, like encrypted and unencrypted unicast communication and peer devices and a callback function to inform of success or failure. Some of the disadvantages are the limitation of encrypted peers, limited payload and broadcast not being supported.

2.2.2.2. *Server Side Communication.* Server side communication is how the data leaves the aggregation node and the WSN onto the cloud where it can be stored and analyzed.

The most used server-side communication protocols include Wi-Fi but with new technologies on the rise, like LoRaWAN, NB-IoT and SigFox.

In Table 2.2 are some key features for server-side communications protocols.

TABLE 2.2. Server Side Communications Protocols Characteristics

Parameter	**Wi-Fi**	**NB-IoT**	**SigFox**	**LoRaWAN**
Data rate [Mbps]	1	0.25	1	0.11
Frequency [GHz]	2.4	0.686	0.433/0.868	0.433/0.868
Range [m]	10 - 100	10 - 100	1 - 10	2000
Security	WPA/WPA2	128bit	128bit	128bit-AES
Nodes	32	-	-	15000

LoRaWAN - Is a bidirectional communication protocol that makes use of the physical layer of LoRa to provide long range communications at low power [**15**]. To do this, LoRaWAN uses a Chirp Spread Spectrum (CSS) modulation, that uses the same low power characteristics of the Frequency-Shift Keying (FSK) modulation (which is part of many other communication protocols) but with an increase in range. With LoRaWAN it is possible to cover hundreds of square kilometers [**21**].

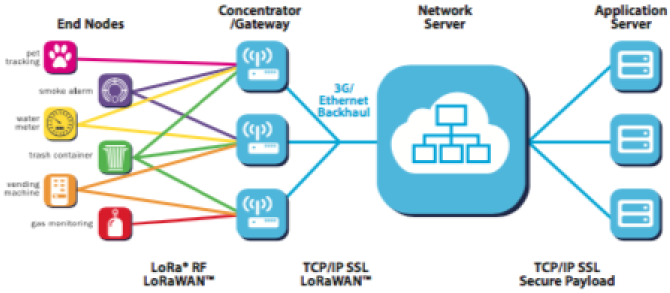


FIGURE 2.7. LoRaWAN Topology [**21**]

In Figure 2.7 its possible to see the different topology of LoRaWAN, when compared to other communication protocols (for example, Figure 2.8).

In LoRaWAN the nodes are not affiliated with a specific gateway and the data transmitted by the node is receive by multiple gateways that will forward it to the cloud-based server.

Advantages, similar to LoRa, consist of a long communication range, low power consumption and a 128-bit AES encryption provides a secure connection with the biggest disadvantages being its limited package size of only 555 bytes.

IEEE 802.11 (Wi-Fi) - IEEE 802.11, commonly known as Wi-Fi, is a communication standard for Wireless Local Area Network (WLAN) [**22**]. It uses a bidirectional radio frequency to connect the devices within the WLAN. Released in 1999, as since come a long way from its initial signal rate of 1Mbps.

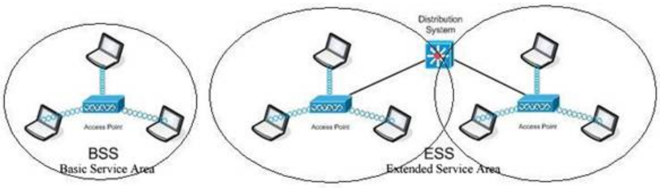


Figure 2.8. Configuration of Wi-Fi Networks

The architecture of IEEE 802.11 is made up by distinct components that cooperate, to provide a WLAN that can support station mobility [**19**]. The key cell of IEEE 802.11 network is called Basic Service Set (BBS) which is a set of fixed or mobile stations. If a station moves out of the BBS, it can no longer communicate with other devices from the BBS.

Based on the BBS, IEEE 802.11 can also implement an Independent Basic Service Set (IBBS), where the devices can connect directly, ,don't need to connect to Access Point (AP). This action is called ad-hoc network, seeing that the Local Area Network (LAN) is pre-planned and only remains active during the time its needed.

Communication over Wi-Fi inside a WLAN is done by TCP/IP packets and each device is identified by its IP and MAC address.

Advantages of Wi-Fi are the ability to penetrate objects, the ease with which you can add or remove nodes and the 128-bit AES encryption that provides a secure connection. The disadvantages are its power consumption, complexity and the fact that radio waves in the network can interfere with the equipment.

Narrowband IoT - Is a standard based LR-WPAN system, developed to allow devices to connect through mobile phone signals [**23**]. It uses devices with low data rate that take advantage of the cellular network to communicatae between each other but with the convenience of consuming less energy.

The main focus of NB-IoT is improving indoors coverage with low cost devices, battery life for up to 10 years and high connectivity.

Some advantages of NB-IoT are the long battery life, penetration capability and data transmission over long range for low bit rates. Disadvantages are the low data rate and no voice transmission.

SigFox - Its a low power solution to connect sensors and devices [**24**]. It focus on using low energy consumption which allows for a longer battery life and cost-efficient.

Mainly applied in networks where there is little information to send but the range of operation is great [**25**].

Because of its simplicity in terms of configuration and low cost its a great protocol to second other networks.

Less overhead, wide coverage area and a lightweight protocol that handle smaller messages efficiently are some advantages of SigFox and the disadvantages being the impossibility of implementation in high data applications, a narrow band spectrum and working better in a fixed location.

Message Queue Telemetry Transport - Message Queue Telemetry Transport (MQTT) is a communication protocol built on TCP protocol, with a low complexity and with the main goal of connecting devices to embedded networks. Is a many-to-many type of communication and it consists of three components: publisher, broker and subscriber [**26**]. MQTT differs from the other protocols mentioned above has it is not a communication protocol but instead is a protocol method commonly used in IoT, that works side by side with other protocols like Wi-Fi and NB-IoT.

The publisher, to communicate via MQTT, sends a message with two components: a topic and a message. The broker receives the message and distributes it between multiple devices. This devices, registers as the subscribers for the specific topics of a publisher, in order to gather the data [**27**].

In this publish/subscribe model, the subscriber can only receive messages from the topic they have subscribed to. This is ensured by the broker that also makes available the different topics. Contrary to HTTP, with this type of publish/subscribe model, is possible for several publishers to communicate at the same time, so that one node does not need to end its connection with the server for other to make a new connection [**28**]. Since it provides a simple implementation and provides routing for low power, low cost and low memory devices in low throw low bandwidth networks [**29**], is one of the most suitable connection protocols for IoT.

This message exchange process is illustrated in Figure 2.9.

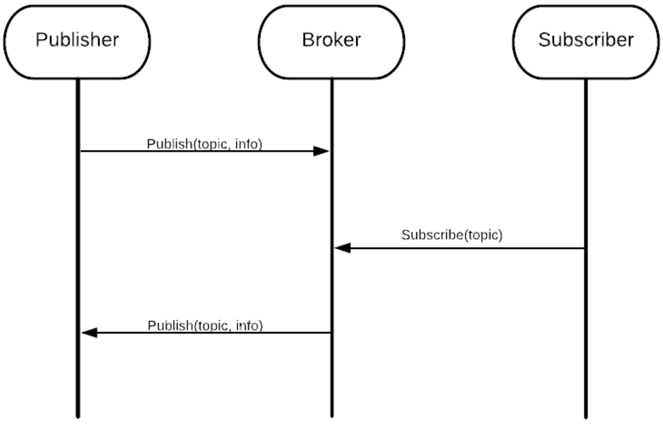


FIGURE 2.9. MQTT - Publish/Subscribe Process

2.2.3. Data Analysis

Machine Learning (ML) is an area of artificial intelligence that allows for the development of very precise software applications to predict results [**30**]. Machine Learning systems can learn from the data, identify patterns and make decisions with minimum human intervention. It is stated by the author of [**31**] that Machine Learning is a method of data analysis that automates the construction of analytical models.

The reason Machine Learning algorithms can predict outcomes is because access to historical data is provided to the algorithm, generating a dataset, making the algorithm learn, that is, iteratively adjusting the information model representation to enhance its efficiency and performance. After this learning process, the algorithm can make precise predictions for future situations that are related to the historical patterns [**32**].

Machine Learning is influential by virtue of allowing to build precise models, considering its conceivable to produce, fairly quick and automatic models with extensive and complex sets of data. Machine Learning can be divided in many subsections but we will focus our attention on only one and that is Classification.

Classification, in Machine Learning, is a predictive model that approximates a mapping function (f) from input variables (x) to identify discrete output variables (y) [**33**], represented in Equation 2.1. The mapping function of the algorithms is responsible for predicting the class output of the given input variables. In simpler words, Classification categorizes a set of data into different classes.

$$y(f : x \to y) \tag{2.1}$$

An algorithm that implements Classification is known as a classifier. There are multiple Machine Learning classification algorithms with their effectiveness depending on the type of dataset that will be analyzed. For this system we intend to implement, the algorithms with the best efficiency for the design are as follows:

Support Vector Machine (SVM) - Used mostly for classification, it classifies the data by building n dimensions between two classes and finding an optimal hyperplane to categorize the data, seen in Figure 2.10, using the distance between the neighboring points and differentiating between the classes with minimum error margin [**34**]. In a more simple explanation, given training data, the algorithm outputs the best hyperplane that classifies new examples [**35**].

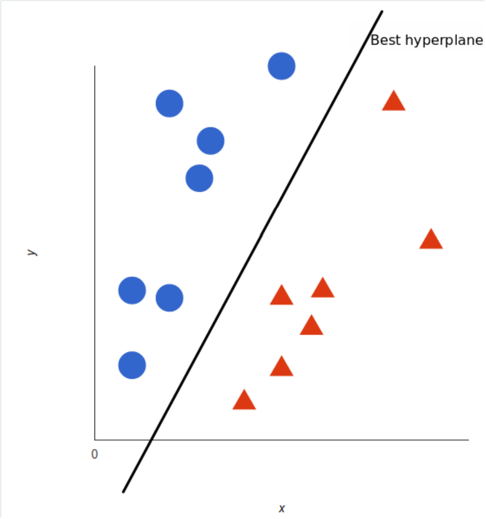


Figure 2.10. Support Vector Machine Architecture

Almost all of the training data becomes superfluous once a boundary is established and a set of points helps to identify said boundary. The points used to attain the boundary are called as support vectors. Support Vector Machine presents the best classification from a provided set of data which means that the model complexity of an Support Vector Machine is not affected by the number of features detect in the training data. For this reason, Support Vector Machine works efficiently to deal with learning tasks where the number of features is large. Applying Support Vector Machine in WSN helps to addressed problems such as localization, connectivity problem and fault detection [**36**].

Decision Trees (DT) - Through an hierarchical partition of training data, a certain feature is used to split the data, with this split being done iteratively till the leaf node contains a number of records that can be used to classify the data [**31, 37**]. However, as described in [**38**] this algorithm faces some limitations because a small change in the training dataset can lead to a substantial change in the tree, making it harder to predict the next values with precision.

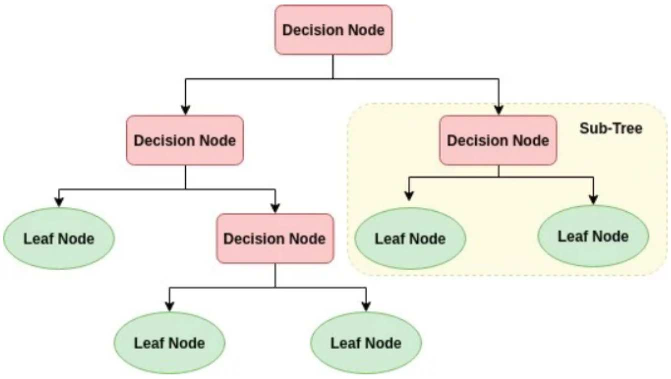


FIGURE 2.11. Decision Trees Architecture

Based on a set of "if-then" rules to increase readability, a Decision Trees contains two types of nodes denominated leaf nodes, the final outcomes of the classification, and decision nodes, the choices between given alternatives, presented in Figure 2.11. Decision tree uses to predict a class or target by creating a training model based on decision rules inferred from training data. Some advantages of a Decision Trees are the easy comprehensive analysis and a reduction of ambiguity in decision making. Decision Trees are adopted to solve many problems in WSN such as connectivity, data aggregation and path selection [**39**].

Random Forest (RF) - Best applied for classification problems, integrates the process of aggregation bagging and Decision Trees, by choosing a subset of features from an individual nodes of the tree and that way it avoids the correlation on the boostrapped set [**37**], working with an assortment of trees, in which each tree gives a classification.

Random Forest algorithm works in two stages, seen in Figure 2.12, the first being the creation of a random forest classifier and after the prediction of results and works efficiently for datasets that are large and have different data. This allows for the algorithm

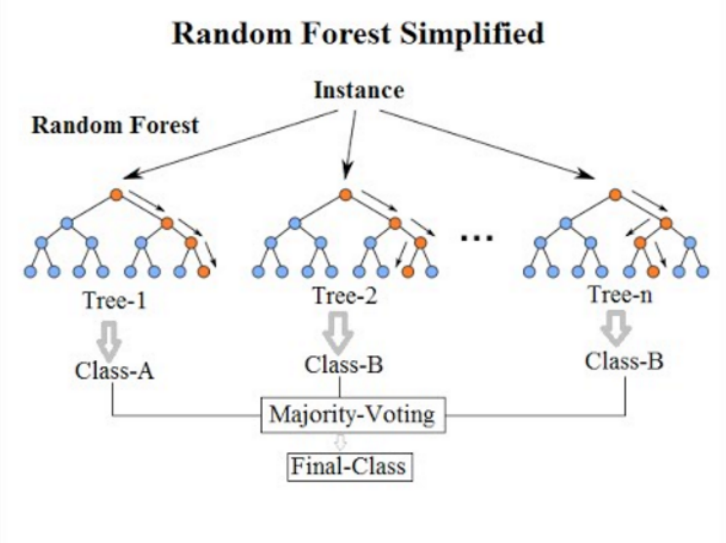


FIGURE 2.12. Random Forest Architecture

to accurately predict the missing values. Random Forest algorithm has been applied to solve various issues in WSN such as coverage and MAC protocol [**40**].

Neural Network (NN) - Functions identical to the nervous system, using neurons in various layers in bio-system like shape. Each of the neurons analyzes parts of the input and sending the information to the next layer and neurons continuously (Figure 2.13), until its able to reach a valid output [**31**].

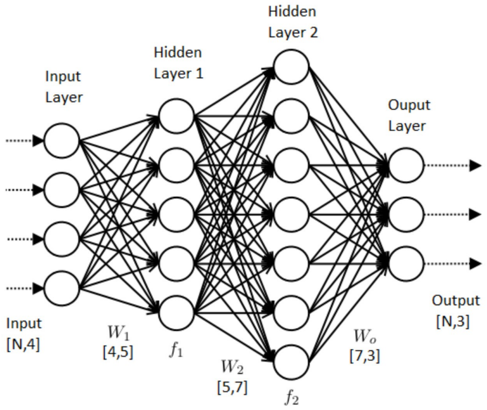


FIGURE 2.13. Neural Network Architecture

Ideally used for non linear and complex problems which requires a big computational power which is not fitting for data from IoT systems [**37**]. In our case, we used Multilayer Perceptron (MLP) a variation of NN algorithm which consists of multiple neurons organized into layers [**41**].

Neural Network usually works in layers, with these layers connected to nodes and each node associated to a function. The basic structure of an Neural Network contains three layers called input layer, one or more hidden layers and lastly the output layers. This allows for complex classification with no restriction for the inputs like other classification methods. Neural Network can be used to help face numerous problems in WSN including localization, data aggregation and congestion control [**42**].

Extreme Gradient Boosting (XGB) - With a similar model to Decision Trees, the objective of this algorithm is, as the name suggests, boosting the performance of the model, presented in Figure 2.14. It creates a sequence of models and rather than training all individually, it models in succession such that these models attempt to correct the mistakes of the models before [**43**]. The first model is built on original dataset, with the second model improving the first model, the third model improving the second, and so on. The models are added sequentially until no further improvements can be made.

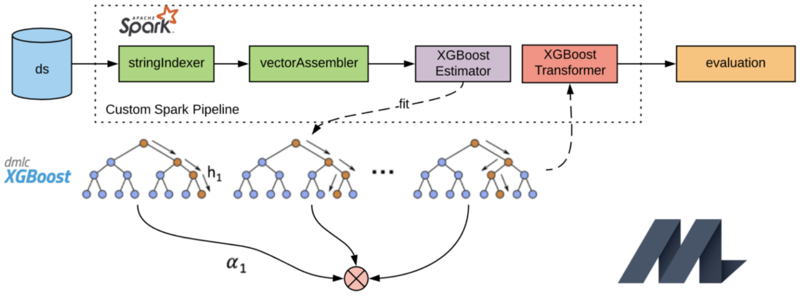


FIGURE 2.14. Extreme Gradient Boosting architecture

2.3. Mobile Application

A mobile application will be the main hub for the retrieve information and where the data will be shown. The data is sent, periodically, from the sensors with current values and a graph to show fluctuations from the values in time so that the user can have a detailed view of the system. With the given information, the user can see if there are leaks in is pipes and that way can act upon it faster and prevent a greater loss of resources.

For this, it is necessary to evaluate which is the best operating system in which this mobile application will be used. Currently there are a vastly amount of mobile operating systems but the most predominant being Android and iOS.

Android is an open source based on Linux and developed by Google with the main focus of its designed being mobile devices such has smartphones and tablets. It uses

Java as the programming language and with its own IDE, AndroidStudio, which can be installed on all computers [**44**].

iOS is an operating system developed by Apple and works solely on Apple devices, more specifically on iPhones and iPads. By Apple standards, it does not allow for their operating system to be used on hardware that does not belong to its brand. The language used to develop mobile applications for iOS is Swift and can only be developed on Apple own software and hardware, which usually means a large investment is required [**45**].

In the global market of mobile devices, Android operating system is by far the most used one, with a market share of around 74%.

With the information, it only makes sense that development of the mobile application, for the proposed system, will be developed using Android operating system.

2.4. Related Work

Water is a crucial natural resource and its widely mishandled. There is an estimate that one third of the world water utilities have a loss of water of around 40% due to leakage [**46**]. Traditional, pipeline leakage detection requires periodical inspection with human involvement, which makes it slow and inefficient for timely and fast detection. This leads to an increase in the development of methods to detect, locate, and fix leaks and its something we can see in numerous different projects.

In [**47**] the authors created a system to inform and control the water usage in households and buildings. The system, a WSN, uses different types of sensors to measure the water flow along the plumbing system. The system can detect water leaks, but only if there is a sudden decrease in the water flow and it cannot identify where the leakage occurred, only that it occurred.

The authors of [**48**] also developed a WSN with thermal sensors to detect water leaks in the pipes. In this case, the temperature of the soil above the leakage will change, so it is possible to detect with some assurance the location of the leak in the pipe. The problem with this system lies with the inevitable difference of soils around the world. It is easier to detect temperature changes of the soil on a sand type of soil than on concrete or brick.

In [**49**] it is presented a system of Narrowband IoT (NB-IoT) using ultrasonic sensors to detect, with precision, the location of leaks in the pipelines. The system uses the sensors to detect the minimal change of water passing by the pipes and when detected, inform the user where the leak happened. However, this system requires expensive sensors which creates a system that it is not accessible to all users.

We can see that there are already various types of WSN solutions for water leak control and detection, but all with some limitations that have prejudice its efficiency in precisely detecting the leaks. By using low cost hardware combined with ML and a communication protocol our proposal aims to solve this problem.

CHAPTER 3

System Architecture

The main objective of this dissertation is to design a control and monitoring system to be applied in water distribution pipelines, in public and private domain. This system proposes to use multiple sensors with real time data collection, mainly water flow parameters to improve the efficiency and the early detection of leaks, with the support of Machine Learning techniques. This system can be seen in Figure 3.1.

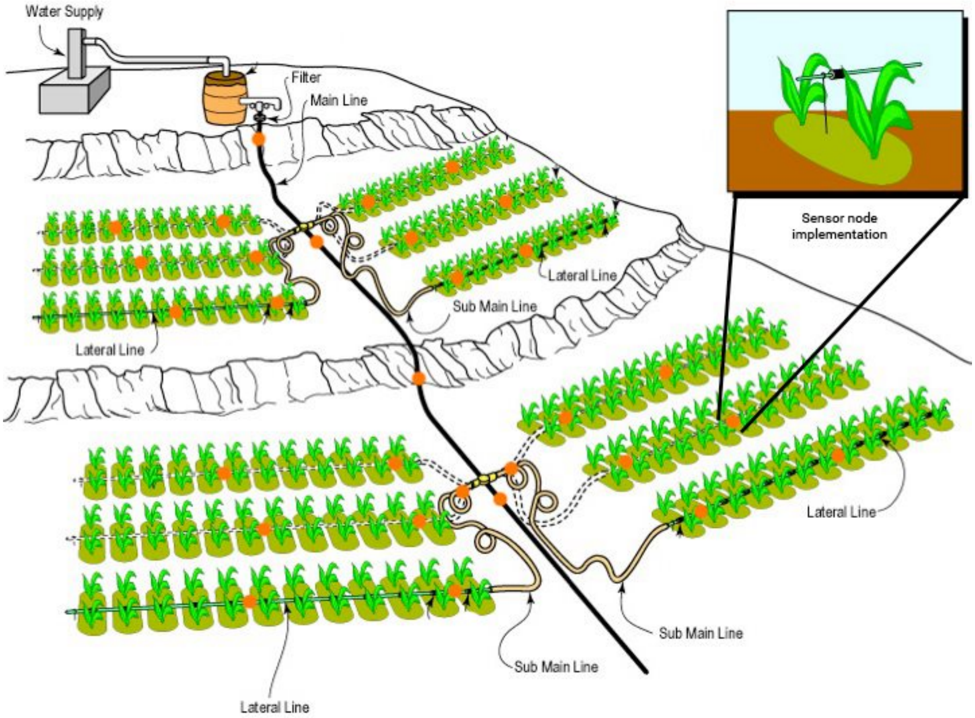


FIGURE 3.1. System Implementation

As mentioned in Chapter 2, there are already identical systems developed, none meet the desire requirements of being able to detect with precision the location of leaks in the pipelines. With this in mind, our new IoT system was developed from scratch and, as can be seen in Figure 3.2, is divided in multiple modules, with both software and hardware parts, each with its individual purpose, from the hardware nodes for data collection, the server for data processing, and finally the user.

The system consists of various water flow measuring sensors, spread throughout the water distribution pipeline that will collect information as water flows by them. This

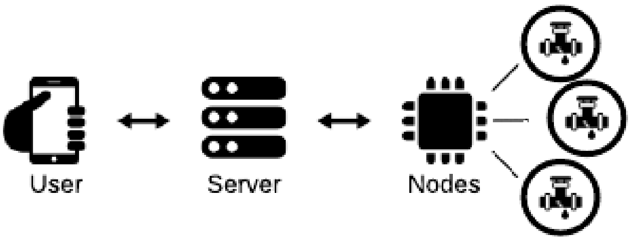


Figure 3.2. System Architecture

information is then sent to the aggregation node, the main hub of our system and that will be responsible to communicate with the server and send to it the information from the sensors. In the server the information will be saved and will also run through the Machine Learning algorithm to be studied and interpreted. From here, depending on the analysis done by the algorithm the information is shown to the user as being all normal within the system or will alert the user for potential locations of water leaks. This logic can be better comprehended in Figure 3.3.

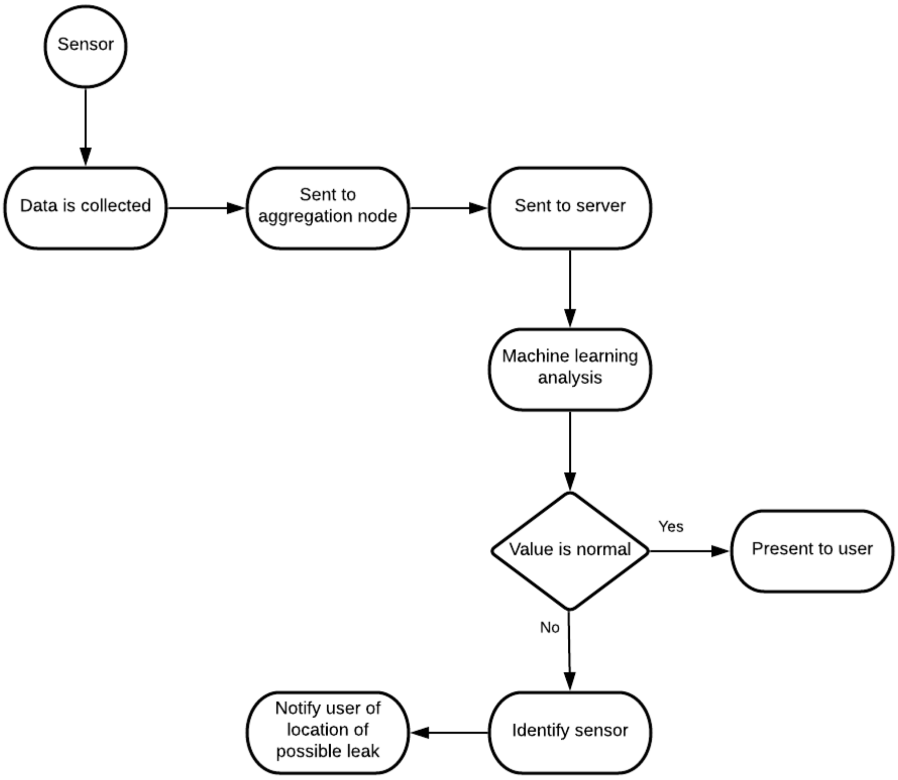


Figure 3.3. System Logic

In this chapter, the architecture will be defined in detail, with all hardware and software components that collectively devise the final system.

3.1. Data Collection

The suggested system confides in a network of sensors, as showed in Figure 3.2, that conjointly create a typical WSN, capable of retrieving data from the sensors and connected to the server.

As mentioned, the system follows a normal guideline to a WSN and is composed by sensors nodes with a communication between the sensor nodes and the server.

Each of the nodes was created in a way to get the maximum efficiency, low cost and low power with the best microcontroller, communication protocols and supplementary hardware being used, as studied in Subsections 2.2.2 and 2.2.1.1.

3.1.1. Aggregation Node

The aggregation node is the central node of the network and the one responsible for keeping the network connected. The aggregation node does not collect any data, just sends the information it receives to the server. In our system it works as a bridge between the sensors, that gather the data, and the server that stores the data.

In Figure 3.4 is possible to see how the aggregation node of our system is built.

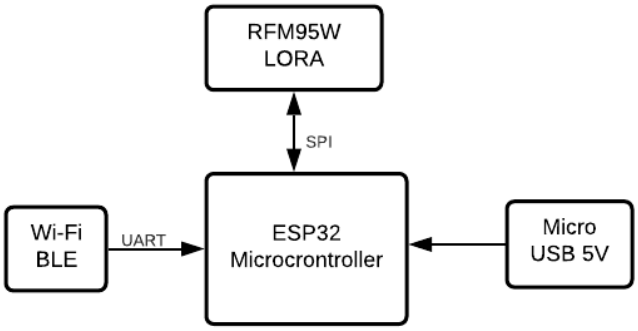


Figure 3.4. Agreggation Node Diagram

The communication protocol between the aggregation node and the sensor nodes was chosen to be the one with the lowest power consumption and biggest range coverage so that the system works all connected in an efficient way. For this, and with what was studied in Section 2.2.2, the best solution for the intra-node communication of our system is LoRa.

LoRa connectivity is achieved using the RFM95W, a LoRa transceiver that grants a long range spectrum communication and immunity to high frequency, all while reducing power consumption. It supports high performance FSK modes for systems and delivers exceptional noise, selectivity, receiver linearity for a much lower current consumption than other devices [**50**].

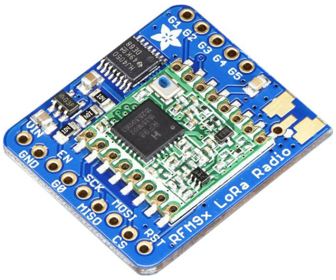


FIGURE 3.5. RFM95W Transceiver

Some of the key feature of this transceiver module are +14 dBm high efficiency power amplifier, high sensitivity, down to -148 dBm and built-in temperature sensor and low battery indicator [**50**].

For the server node communication, MQTT is used, to take advantage of the already integrated Wi-Fi capabilities of the ESP32 microcontroller.

In remote locations, where no Wi-Fi coverage is available, NB-IoT can be used to provide a celular coverage. This can be achieved by attaching the SIM7000E module to the ESP32, providing a 2G and NB-IoT connectivity to the node. MQTT works as well with NB-IoT, being unnecessary to make changes in that part.

3.1.2. Sensor Node

Sensor node is the simplest and the lowest level of a WSN. Their sole purpose is to collect data from the attached sensors and send it to the aggregation node. In order to perform these tasks, the sensor node needs a microcontroller and a set of sensors that can be permuted from node to node, depending on the different needs for the solution.

In Figure 3.6 is represented the sensor node schematics for the system. It shares the same main hub as the aggregation node, the ESP32 controller, with the difference being the communication protocol being used.

The sensor nodes consists of, as mentioned before, an ESP32 microcontroller, a RFM95W module and an array of sensors suited for retrieving information and send it to the aggregation node. To transmit the information collected from the sensors to the aggregation

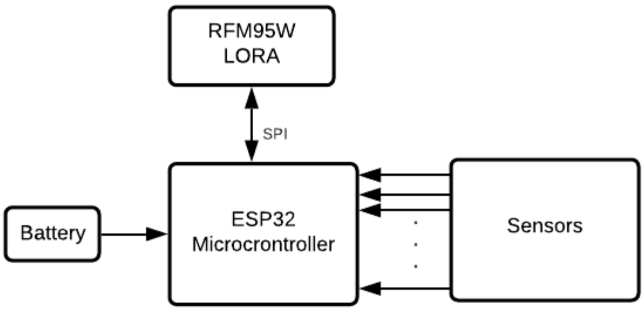


Figure 3.6. Sensor Node Diagram

node, we make use LoRa through the RFM95W module that allows for a much lower power consumption.

Seeing as the node only needs to be awaken when water is flowing by, we can power the sensor node with a battery which allows for a completely wireless node and a longer lifetime. And with the only goal of measuring the water flow of the water pipelines, we just need one type of sensor, distributed through the sensor nodes to retrieve the information from the water distribution pipeline.

Usually, a sensor has a limited capacity and are low cost components of a WSN. In our system, we only need on type of sensor and it is to measure the water flow of the water distribution system. For this, the recommended sensor to perform this task is the YF-B2 [**51**], that can be seen in Figure 3.7.

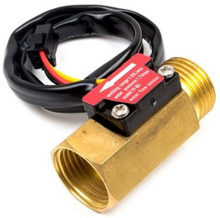


Figure 3.7. YF-B2 Water Flow Sensor

This sensor consists of a water rotor and a hall-effect sensor. When water flows through it the rotor rolls and its speed changes with the different rate of flow.

3.1.3. Communications

As previously mentioned, the communication is the most crucial part of a WSN as it is responsible for transmitting the data collected from the sensors but also to keep the network connected and to assure that the system abides to the tasks that were specified. Communication also helps to improve the area that can be covered, with some protocols allowing the network to expand in terms of area size.

The projected system, composed by two sets of nodes, it will require two types of communication protocols, one for the communication between nodes and the aggregation node and the other for the communication between the aggregation node and the server.

3.1.3.1. ***Node-to-node communication.*** The node-to-node communication is how the information travels inside the WSN. Because sensors can be far from each other and so are power by batteries, a communication protocol capable of long range transmission and low power consumption is essential.

With this in mind and in order to have all the nodes in the network connected LoRa was implemented, using a RMF95W module. Also used was RadioHead, an Arduino library, which enables node addressing, with unique addresses being set for each node, guaranteeing differentiation in the network.

With the RadioHead library, the nodes in the network can communicate in two different ways. By sending a broadcasting message to every node with the destination ID included in the message and by directly messaging the node by using its destination address.

3.1.3.2. ***Node-to-server communication.*** The node-to-server communication is how the information containing the data gathered from the sensors leaves the network. To connected with a server outside of the network, it requires some kind of internet connection and this is were the ESP32 comes to our advantage has it haves Wi-Fi built-in it only required a low power way of using it to transmit the data.

In order to always maintain a connection between the network and the server the MQTT protocol was selected based on its architecture that fits IoT projects and for its lower power consumption's when facing typical HTTP requests, as detailed in Subsection 2.2.2.2.

To implement a MQTT protocol its necessary to defined specific topics for each way of the communication link. making it easier to distinguish the messages intended for the server and the ones intended for the network:

- **"teste/joao/in":** topic to send the messages from to the server and mobile application;
- **"teste/joao/out":** topic to be subscribed by the network to receive messages;

The server and mobile application, by subscribing to the "teste/joao/in" topic, wait for new information from the sensors to save it and to show it to the user.

In the sensor nodes, the sensors needed a efficient and low power way to transmit the data to the microcontroller. To do that, using Arduino IDE and C++, a script was created and implemented directly on the ESP32 with the objective of collecting data from the attached sensors, using the corresponding libraries.

After gathering the data, the next step is sending the information to the server. To send the information to the server, as described in Subsection 2.2.2.2, the protocol used was MQTT, following the publisher/subscriber model. The ESP32 works as the publisher and uses the data collected by the sensors has the body of the message and forwards it to the server, using the topic subscribed by the server. For this, the PubSubClient library was used to implement the MQTT on the microcontroller.

3.2. Data Analysis

For the user to be able to see, monitor and control the system through the real time data available by the mobile application, it was necessary to create and run a few scripts. The main purpose of this scrips is to receive the data gathered by the sensor node and send it to the server and mobile application. The next section will dissect the how, when and where of each of the scripts was implemented.

3.2.1. Storage

The storage is where all the data collected by the sensors handled and analyzed to be correctly saved in the database. It is through the storage that the data passes from the hardware on its way to the user. For the storage to function properly, a Python script was developed and deployed in the background to maintain an open connection from the sensor node to the storage.

This script has some complexity due to the fact this script has a dual purpose, to connect to the aggregation node to collect the information and to send the data to the database.

To receive the information from the sensor node, that is sent via MQTT protocol, we implemented the Paho library in order to allow the script to subscribe to the topic where the messages from the sensor node are being sent. With this, any message created in the WSN is sent and receive in real time by the server.

After a new message is received, it needs to save in the database the corresponding sensor values. By using the MySQLdb library a connection to the database can be established and through SQL commands save the values. The database is needed because we are dealing with a real time monitoring system therefore a large volume of data, so we need to structure the information to be able to access it easier in the future.

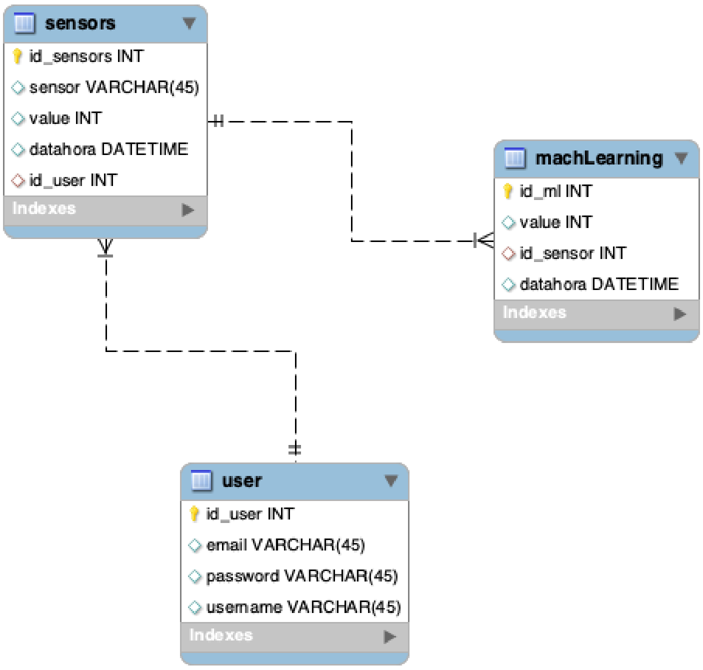


FIGURE 3.8. Database Relational Model

The structure of the database can be seen in Figure 3.8. Each data collecting sensor has an unique id, so that in the message we can identify from which sensor the information comes from.

3.2.2. Application Programming Interface

The mobile application is where the user can interact and see what is occurring throughout the system. To reach the mobile application interface we needed to create another Python script.

When the information arrives at the server, it needs to be redirected to the application via MQTT. For that, the mobile application uses the Paho Java MQTT library, which recognizes the topics that the mobile application subscribes to.

To get the information from the database and show it on the mobile application, it is necessary to execute queries to retrieve the information. For that, we implemented Flask in the Python script. Flask is an Application Programming Interface (API) that allows to build web applications. Because the mobile application needs specific information, it was implemented a HTTP connection in Java to allow the mobile application to retrieve the information from the URL created by Flask.

This script imports diverse libraries to complete its purpose and uses them in the following order:

- **Flask:** generates the URL and web application;
- **MySQL:** creates the connection to the database;

Once the data is collected, it is presented on specific screens of the mobile application.

3.2.3. Machine Learning

The proposed goal of the pretended system is to identify in real time, the locations of possible water leaks and for that we make use of Machine Learning. The first step for the system to be able to predict leaks is to create a dataset and pre-process the data. Afterwards, the chosen algorithm will be implemented and trained to start analysing the system and detect water leaks.

The algorithm receives, as input, the water flow value measured by the sensors and calculates if the pipes before that sensor have any problems. This input is then used to help understand the system and if its everything okay and no water leaks are starting to appear in the water distribution pipeline.

3.3. Data Visualization

To provide the user a way to see the data collected from the sensors of the system, an Android mobile application was developed from scratch. The mobile application is called ”Water Leak Control” and its logo icon is presented in Figure 3.9.

FIGURE 3.9. Mobile Application Icon

The mobile application was developed in Java programming language and using the official IDE for Android development, AndroidStudio. The mobile application was developed in the last version of the OS available.

The mobile application features and the developed Android Activities layouts will be presented ahead.

3.3.1. Application Features

As the name of the mobile applications states, the objective of the mobile application was to give the user a dashboard of the water distribution system, where it is possible to check all the values from the sensors in real time, everywhere.

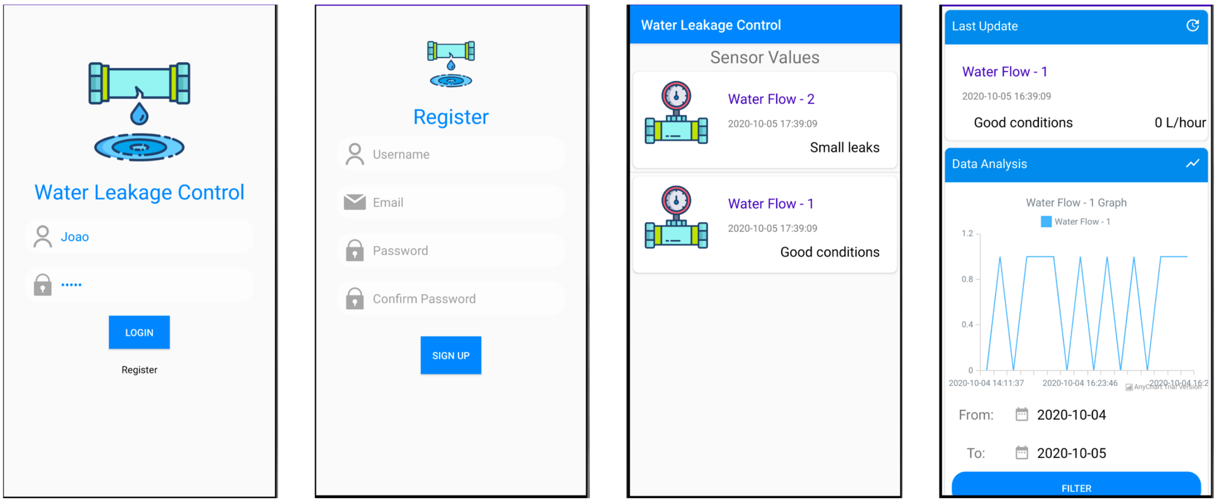
Regarding the values retrieved from the sensor nodes, the application gives the user the opportunity to check in real time, not only the latest values, but also the current conditions of the corresponding pipes, as evaluated by the Machine Learning algorithms, as well as historical data and conditions. For the real time system, the Paho Java MQTT library was used to subscribe to the topic the network publish the information.

If the user does not check the application on a regular basis, it will warn the user via a notification if some problem is encountered by the system.

3.3.2. Android Activities

In order for the user to have a visual interaction with the system and to be able to perform the described functionalities, the developed Android application is composed by a set of screens.

Figure 3.10 presented the developed screens which will be described in detail and individually in the following subsections.



a) b) c) d)

FIGURE 3.10. Android Screens

3.3.2.1. ***Authentication.*** In order to be apart and use the system, the user must register in the application so that he can authenticate himself. This authentication can either be a new registration or a login with an already created account.

When the application is launched, the Login screen, presented in Figure 3.10 a), is shown to the user to authenticate. Username and password are required for the user to authenticate and use the app.

3.3.2.2. ***Register.*** To register a new account the user must provide some information such as e-mail, username and password. This screen, as seen in Figure 3.10 b), is shown when the user clicks ion the ”Register” button within the Login Screen.

3.3.2.3. ***Main dashboard.*** The main screen, Figure 3.10 c), for the mobile application is where some of the sensors information is provided to the user. Is is possible to see the sensors that are part of the system, as well as a short text indicating the status of the pipes. When a sensor is clicked, the user is redirected to the screen containing the sensor details.

3.3.2.4. ***Sensor Details.*** This is the screen where the user can see more detailed information about the sensor that was selected in the main dashboard.

The screen is divided in two sections. The first is a more detailed information tab than the main screen where the value from the sensor is showed and the message about the status of the pipes. Below appears an interactive graph, that shows the last data analyzed by the machine learning algorithm and allows the user to select a specific interval of dates. It is also possible to see a timestamp and sensor value when clicking on a value on the graph.

CHAPTER 4

Machine Learning Training

A central part of the system is the machine learning algorithm, which is used to predict water leaks and inform the user of the leak location. To be able to do this, the algorithm must first be trained to understand the system and the environment so that in case a water leak happens, the system may know what is happening and what to do.

In this chapter will dissect and analyze everything related to the Machine Learning, from the different algorithms tested, to the final training of the best algorithm chosen for the system.

4.1. Dataset Creation

In order to train the algorithm, first we need to create a dataset with similar data to the one that will be used in the future, containing all possible outputs. As if an output never happens in the dataset, the algorithm will not be able to predict it. In the intended system the Machine Learning algorithm will only be used for analysing a specific scenario: water leaks.

For this scenario, and in order to create the needed dataset, a small water distribution system was created in which our system was implemented.

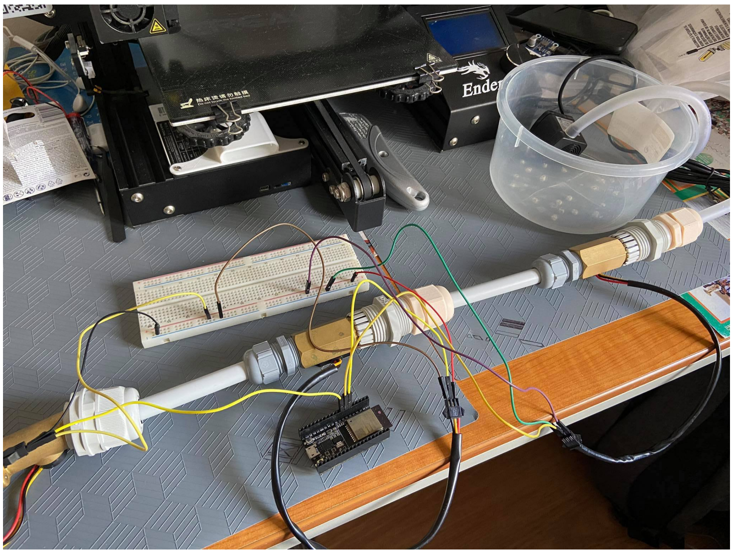


Figure 4.1. System Prototype

The system collected data every 5 minutes, about 12 data samples an hour, with all samples being recorded with a value collected and the corresponding pipe to be able to classify the collected information, for all possible outputs.

To achieve this outputs, in the test scenario, different holes were carved in the pipes, with size variances and with pipes with more holes than others, to simulate a rupture in the system and match to the desire output.

Table 4.1 shows the desired outputs and the meaning of each for the system.

Table 4.1. Intended Outputs

Output	**Meaning**
0	No leaks in the system
1	Micro leaks in the system
2	Minor leaks in the system
3	Major leaks in the system

In order to have a good dataset, that allow for the proposed Machine Learning methodology to work properly, for each of the output possibilities, multiple tests were performed, changing the duration of the test, pressure applied by the pump or length of the pipes, each of these being also performed multiple times to guarantee that small changes in data are recorded.

These tests allow us to obtain 5607 records containing information from three sensors that were spread through the same path of pipe, and which analyse the conditions of the pipes in two different zones. Since multiple situations and scenarios were applied to this data collection, it is possible to assume a good reliability on the data collected and that will train the algorithms, so it can be applied to other scenarios.

Although the collected data shows information from multiple sensors, each individual record only has the timestamp and the data collected from the sensor. So, the first thing that we intend to analyze was how the data that is inputted into the algorithm can affect its accuracy. For that two datasets were created, using the same collected data. One is the data collected according to the sequence of sensors in the system and the other where the data was clustered, to show how the entire path of pipes is working. This was done in order to understand which of the datasets presented the best results for the intended scenario and system.

When creating the dataset, besides the timestamp and collected sensor data, other features were added to each record, based on calculations and pre-processed data from the sensor values.

In Tables 4.2 and 4.3 are presented the fields of each dataset.

TABLE 4.2. Standard Dataset

Parameter	**Meaning**
id	Identification of the data input
sensorID1 / sensorID2 / sensorID3	Id of sensor the information was collected
value1 / value2 / value3	Value collected from the sensor
average1 / average2 / average3	Calculated average of the last 5 values collected from the sensor
diff_ref1 / diff_ref2 / diff_ref3	Calculated value difference from current value and the last value from sensor 1
diff_sen1 / diff_sen2 / diff_sen3	Calculated value difference from current value and the last value from sensor n-1
hasProblems1 / hasProblems2 / hasProblems3	Final output

TABLE 4.3. Clustered Dataset

Parameter	**Meaning**
id	Identification of the data input
sensorID	Id of sensor the information was collected
value	Value collected from the sensor
average	Calculated average of the last 5 values collected from the sensor
diff_ref	Calculated value difference from current value and the last value from sensor 1
diff_sen	Calculated value difference from current value and the last value from sensor n-1
hasProblems	Final output

The average, for the entry *i*, shown in Equation 4.1, is calculated using the last five values of the respective sensor.

$$average_i = \frac{value_i + value_{i-1} + value_{i-2} + value_{i-3} + value_{i-4}}{5} \tag{4.1}$$

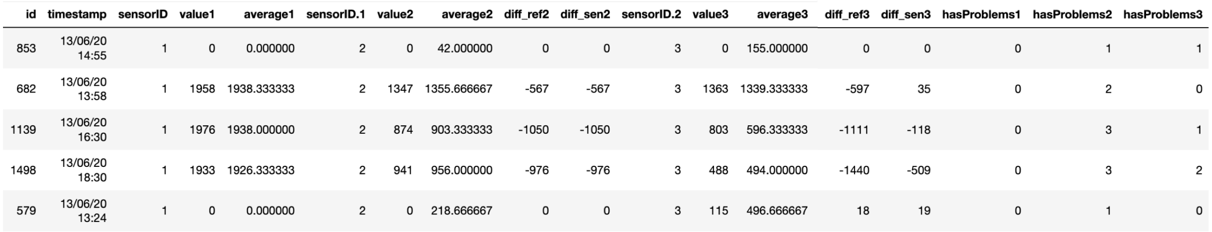
The diff_ref, for the entry *i*, shown in Equation 4.2, represents the difference between the current collected value and the last value collected from sensor number one, that works as the reference sensor for the rest of the system, the one closest to the water supply.

$$diff\_ref_i = currentValue + lastValueFromSensor_1 \tag{4.2}$$

The diff_sen, for the entry *i*, shown in Equation 4.3, represents the difference between the current collected value and the last value collected from sensor n - 1, the one located before the current sensor.

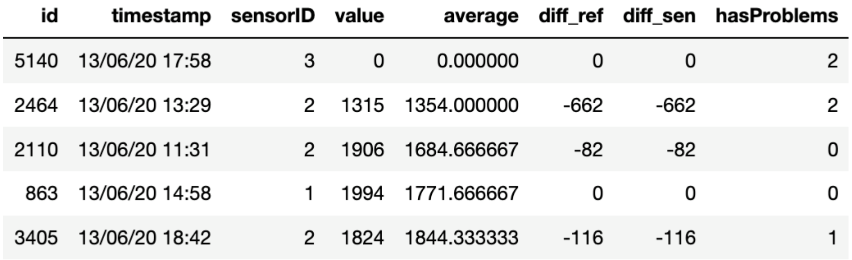
$$diff\_sen_i = currentValue + lastValueFromSensor_{N-1} \tag{4.3}$$

With this, the two dataset created have 5607 and 1869 entries, for the standard and clustered respectively. A sample of each dataset can be seen in Figure 4.2 and 4.3.



id	timestamp	sensorID	value1	average1	sensorID.1	value2	average2	diff_ref2	diff_sen2	sensorID.2	value3	average3	diff_ref3	diff_sen3	hasProblems1	hasProblems2	hasProblems3
853	13/06/20 14:55	1	0	0.000000	2	0	42.000000	0	0	3	0	155.000000	0	0	0	1	1
682	13/06/20 13:58	1	1958	1938.333333	2	1347	1355.666667	-567	-567	3	1363	1339.333333	-597	35	0	2	0
1139	13/06/20 16:30	1	1976	1938.000000	2	874	903.333333	-1050	-1050	3	803	596.333333	-1111	-118	0	3	1
1498	13/06/20 18:30	1	1933	1926.333333	2	941	956.000000	-976	-976	3	488	494.000000	-1440	-509	0	3	2
579	13/06/20 13:24	1	0	0.000000	2	0	218.666667	0	0	3	115	496.666667	18	19	0	1	0

FIGURE 4.2. Clustered Dataset



id	timestamp	sensorID	value	average	diff_ref	diff_sen	hasProblems
5140	13/06/20 17:58	3	0	0.000000	0	0	2
2464	13/06/20 13:29	2	1315	1354.000000	-662	-662	2
2110	13/06/20 11:31	2	1906	1684.666667	-82	-82	0
863	13/06/20 14:58	1	1994	1771.666667	0	0	0
3405	13/06/20 18:42	2	1824	1844.333333	-116	-116	1

FIGURE 4.3. Standard Dataset

In order to reach the goal of detecting water leaks, the algorithms must be trained and tested with both datasets. This guarantees that when used in a real life scenario the results will be precise. And for that, the datasets were divided into a train and test, with 70% and 30%, respectively.

4.2. Model Analysis

For the model analysis, an array of Machine Learning algorithms were used in order to determine which is the best one to use in our system. All the tested algorithms fall under the Classification part of Machine Learning.

Before conducting any individual tests, a feature importance test was conducted on each dataset to guarantee that only the best features would be used. This not only allows for better results, it also eliminates noise from the dataset and improves the computational power, since it reduces the amount of time each algorithm needs to run, because the amount of data being tested is reduced.

As its possible to see in Figures 4.4 and 4.5 the most important feature for both datasets is the diff_sen. When training classification of the datasets, sensorID, value, average, diff_ref e diff_sen were the features used.

As mentioned in Section 2.2.3, we only used Classification algorithms, with the studied algorithms being used to perform the tests. For our scenario we performed a total of 12 tests for each algorithm, each test with different parameters, with the objective to train

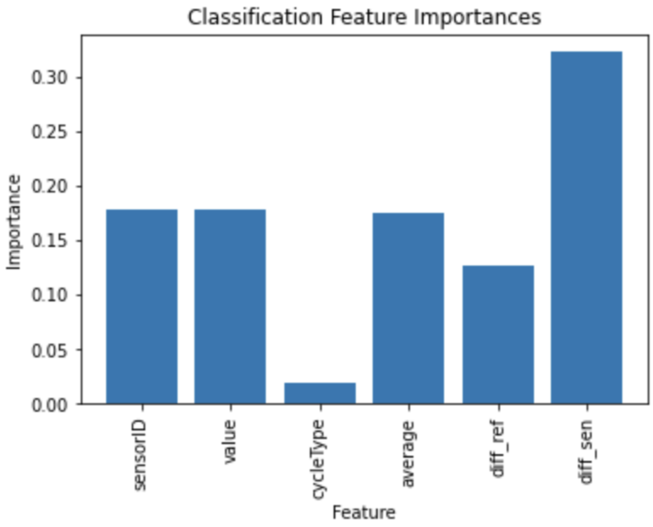


FIGURE 4.4. Standard Dataset Feature Analysis

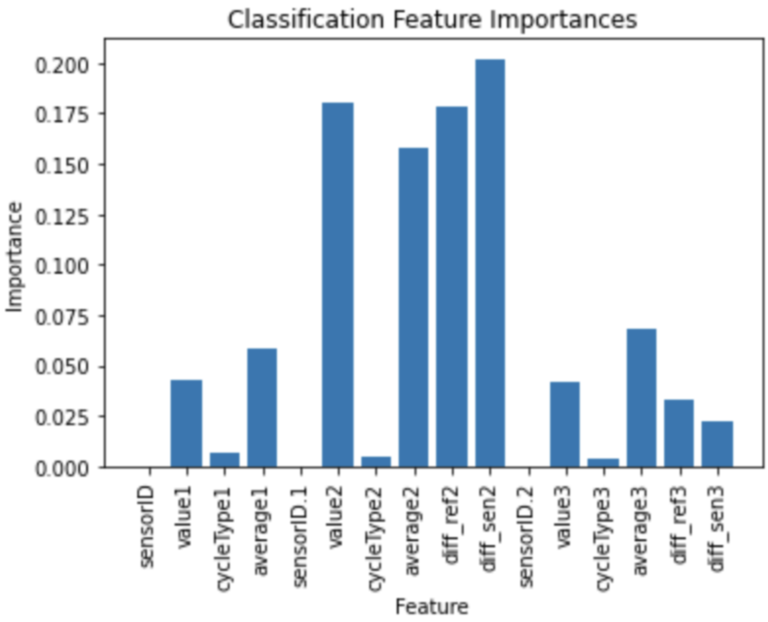


FIGURE 4.5. Clustered Dataset Feature Analysis

the algorithms to understand which algorithm presented the best accuracy so that it could be implemented in our system.

From test to test, some on the input fields were removed from the algorithm to to check if the would work better.

The tests for both datasets were made using different parameters with the goal of training the algorithms for more specific cases. By varying the input variables for each scenario is possible to analyze if the isolated values, i.e., using more or less features, produce better or worst results in terms of accuracy.

Table 4.4 shows all the performed tests, identifying the used dataset, target and unused features.

TABLE 4.4. Test Scenarios

Dataset	**Test Nº**	**Target**	**Unused Features**
Clustered	1	hasProblems2	average, diff_sens, diff_ref
	2	hasProblems3	
	3	hasProblems2	diff_sens, diff_ref
	4	hasProblems3	
	5	hasProblems2	diff_ref
	6	hasProblems3	
	7	hasProblems2	-
	8	hasProblems3	
Standard	9	hasProblems	average, diff_sens, diff_ref
	10		diff_sens, diff_ref
	11		diff_ref
	12		-

In these test cases, the target output wheater a specific sensor has problems or if everything is working correctly in the system, with *”hasProblems2”* or *”hasProblems3”*, for the clustered dataset, indicating a problem on section 2 or 3 respectfully, and as for the standard dataset, *”hasProblems”*, indicating a problem on that specific section.

Each of the tests was conducted using Python, the scikit-learn library and the Jupyter platform. For each of the algorithms a script was developed using the corresponding library for classification, from scikit-learn, ad the default configurations were used. As said, 70% of the dataset was used for training and 30% for testing. It was divided this way to have a good base to train the algorithm and no to test with fewer values.

Table 4.5 presents the results for each test.

TABLE 4.5. Tests Results

Algorithm	**Accuracy [%]**											
	1	2	3	4	5	6	7	8	9	10	11	12
Random Forest	75.40	70.05	80.75	72.73	80.75	73.62	80.57	74.69	74.51	77.00	77.82	84.79
Decision Trees	76.65	70.94	80.21	71.30	80.04	72.19	79.14	70.05	74.03	77.06	77.42	84.02
Neural Network	59.71	60.42	67.74	67.91	75.40	70.23	73.44	72.19	51.69	55.44	74.98	80.68
Support Vector Machine	26.91	26.56	46.17	46.60	51.34	31.02	57.22	41.35	53.77	51.75	29.47	27.09
Extreme Gradient Boosting	76.65	70.94	80.93	75.22	81.46	75.76	81.11	75.04	74.99	77.01	77.78	84.73

Figure 4.6 represents the same results, allowing for a better analysis.

4.3. Remarks

The objective of all of these tests was to determine which had the best accuracy when given inputs related to our water leakage system, as well as check which was the best

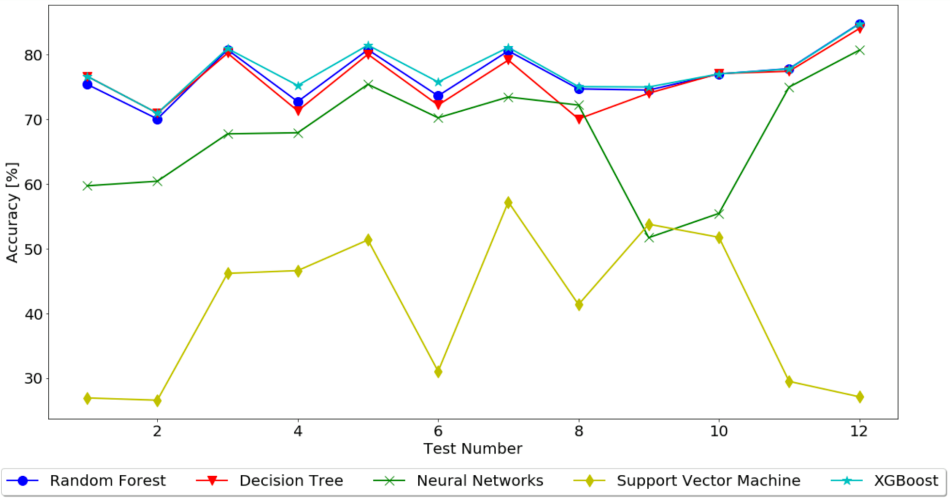


FIGURE 4.6. Test Results

dataset composition. And with what is presented in this Chapter we can take conclusions and define the best algorithm and dataset for our intended system.

As it is possible to attest, the datasets have similar results but with different behaviours, with the accuracy for most of the algorithms being within the same range, of about 70% to 85%, excluding SVM, that for both datasets presented an accuracy within the range of 27% to 53%.

We can also assure that the variables have an important role for the output of each test, always having an influence on the algorithm that is tested. In the clustered dataset, for the algorithms RF and DT, we can see that from the first to the second test for the same target, so test 1 and 3 for *"hasProblems2"* and test 2 and 4 for *"hasProblems3"*, the tendency is for the accuracy to increase with less unused features. This behaviour then changes, and for tests 3 to 5 and 7 for *"hasProblems2"* and test 4 to 6 and 8 for *"hasProblems3"* the accuracy decreases, concluding that this algorithms work better when the number of unused features is bigger. For the algorithms NN, SVM and XGB, the behaviour is the same for all the targets when the unused features decrease, with all the algorithms increasing its accuracy from test to test, thus concluding that this algorithms work better with the lesser unused features. For the standard dataset, all the tests have the accuracy increasing in each test with lesser unused features, with the exception being SVM, where the accuracy decreases when the number of unused features also decreases.

As we can see from the results, SVM showed the worst results in almost all the tests with most results being below 50% accuracy, so it can be discarded as the algorithm to used.

The NN algorithm, even though being better than SVM with no outputs lower than 50% accuracy, it still got final results less than expected and intended.

RF, DT and XGB presented very similar final outputs in terms of accuracy, with no results below 60% accuracy and with a few final outputs above 80%.

Is is also important to mentioned that, for the RF, DT and XGB, the accuracy increases depending on fewer parameters being dropped from the dataset.

So, with what is showed from the results of all the tests is data the best dataset is the standard one, where there is no differentiation from the data, and the best algorithm is the Random Forest, with a best accuracy result of, approximately, 85%, when using all the features on the dataset. This leaves a margin to improve the system, but at the same time it is already an excellent value for the accuracy, compared to the current manual way to do it.

CHAPTER 5

System Implementation

In the previous Chapters, the architecture and the Machine Learning algorithm were studied and discussed. To find out if the proposed system is viable, we need to implement the final system to assure that all objectives, from software and hardware modules, are achieved.

In this chapter, an experimental test, using a small scale system, was conducted to verify that all the modules work harmoniously among themselves, with the communications protocols working as intended and the Machine Learning algorithm being able to detect leaks in the system.

A small case system, simulating a real word environment, was used to apply the system and test

5.1. Laboratorial Tests

In order to assure the proposed system was suited to worked as intended, each module needed to be tested individually and then together. For this, a number of laboratorial tests, in a controlled environment, were performed to understand and attest that they were able to perform their tasks, if the communication protocols where able to maintain communication between them and if the MQTT protocol suited to send the messages from the WSN to the server.

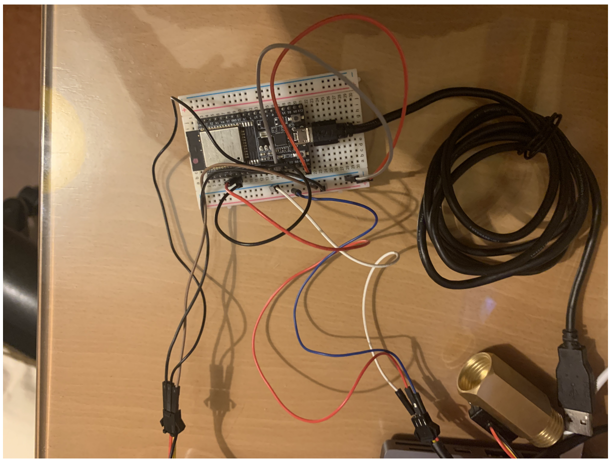
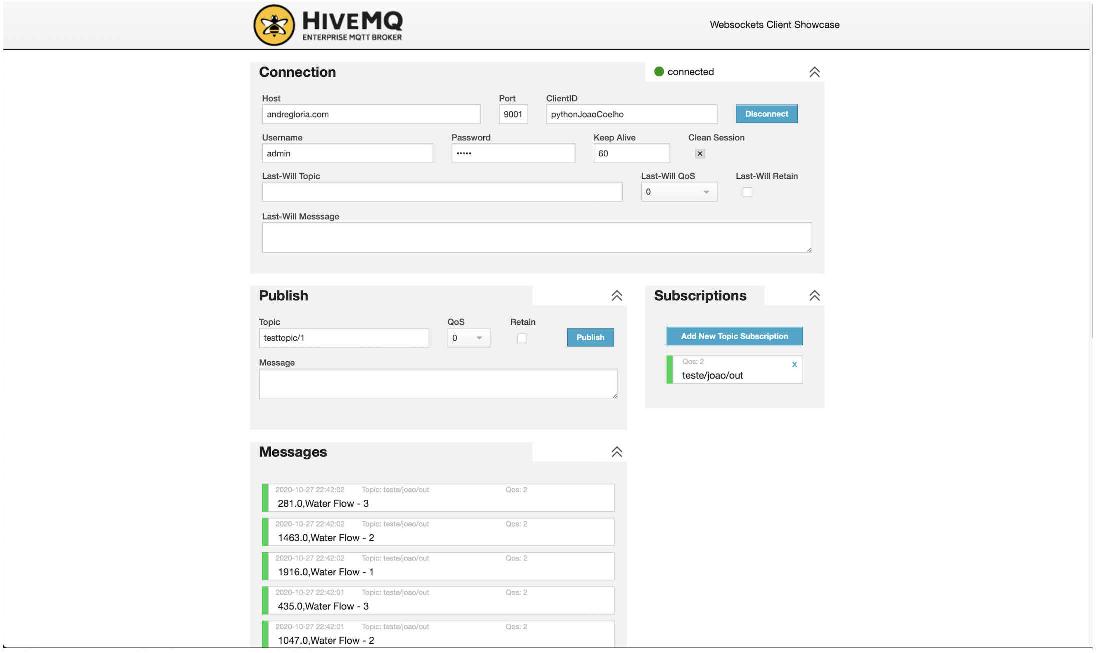


Figure 5.1. Sensor Node

The system, as of now, consists only of a sensor node that concentrates all the functions commonly found in various nodes in a WSN and is presented in Figure 5.1.

By assuring that the sensor node could collected data flawlessly, part of our proposed system and WSN was implemented.

To complete the architecture of the system, MQTT protocol was implemented in the ESP32 so that the information could leave the the WSN and be forwarded to the server. To do that, when a message was received in the ESP32, it would be published onto the corresponding MQTT topic. Using HiveMQTT [**52**], and presented in Figure 5.2, the MQTT logs where is possible to confirm that the implementation was done successfully and the server received the values collected from the sensors. It was also possible to conclude that the network could handle multiple sensors, with intervals between the message being sent to prevent data losses.



Host: andregloria.com
Port: 9001
ClientID: pythonJoaoCoelho
Username: admin
Password: ••••••
Keep Alive: 60
Clean Session: ☒
Last-Will Topic:
Last-Will QoS: 0
Last-Will Retain: ☐
Last-Will Messsage:

Topic: testtopic/1
QoS: 0
Retain: ☐
Message:

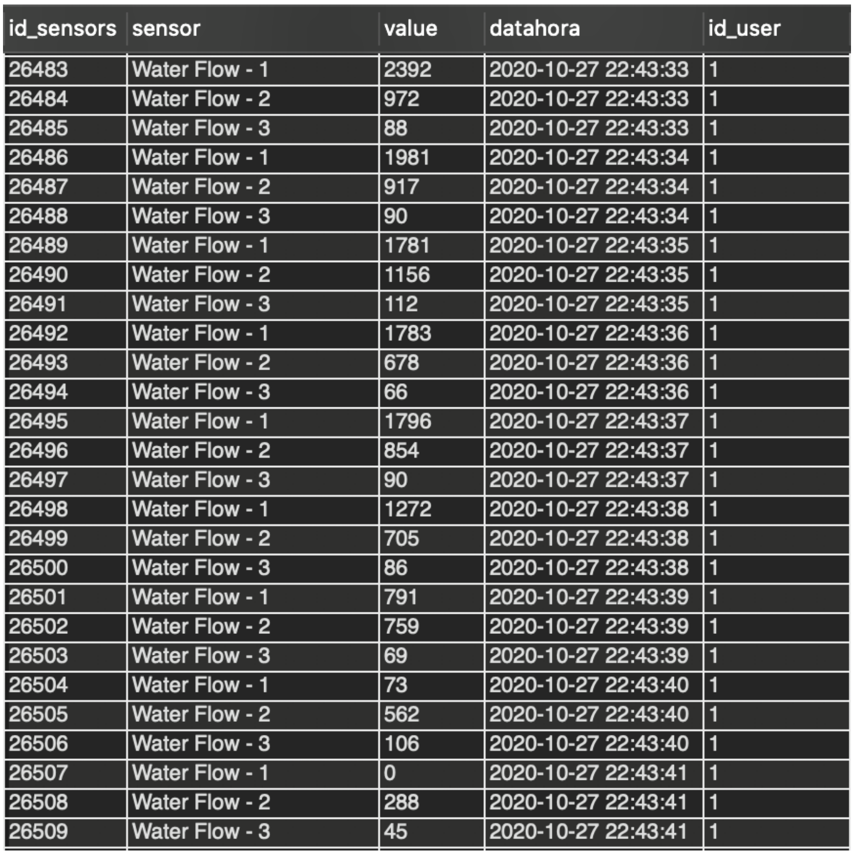
Qos: 2
teste/joao/out

```
2020-10-27 22:42:02   Topic: teste/joao/out   Qos: 2
281.0,Water Flow - 3
2020-10-27 22:42:02   Topic: teste/joao/out   Qos: 2
1463.0,Water Flow - 2
2020-10-27 22:42:02   Topic: teste/joao/out   Qos: 2
1916.0,Water Flow - 1
2020-10-27 22:42:01   Topic: teste/joao/out   Qos: 2
435.0,Water Flow - 3
2020-10-27 22:42:01   Topic: teste/joao/out   Qos: 2
1047.0,Water Flow - 2
```

FIGURE 5.2. HiveMQTT messages received

Guaranteed that the system is able to send data to the cloud using MQTT, the next step was to test if the developed Python scripts were able to receive the MQTT messages, split the information and store them correctly in the database.

For that the node was turned on, alongside the developed scripts, that collected the gathered sensor data in real-time and put them in the corresponding table in the database, as can be seen in Figure 5.3.

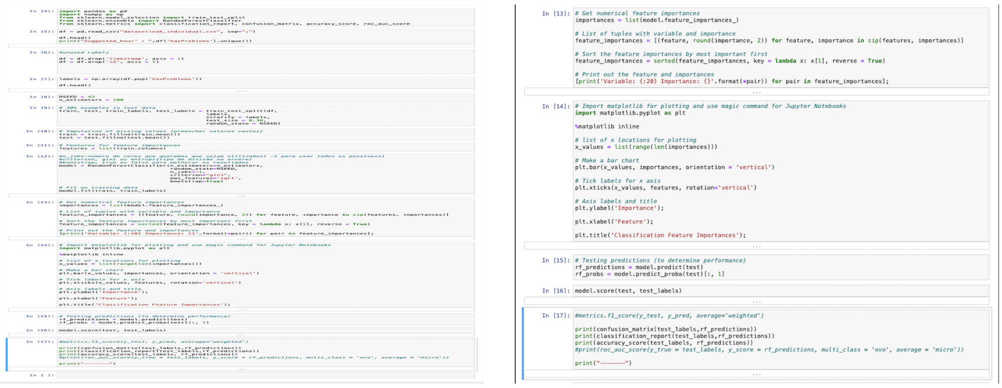


id_sensors	sensor	value	datahora	id_user
26483	Water Flow - 1	2392	2020-10-27 22:43:33	1
26484	Water Flow - 2	972	2020-10-27 22:43:33	1
26485	Water Flow - 3	88	2020-10-27 22:43:33	1
26486	Water Flow - 1	1981	2020-10-27 22:43:34	1
26487	Water Flow - 2	917	2020-10-27 22:43:34	1
26488	Water Flow - 3	90	2020-10-27 22:43:34	1
26489	Water Flow - 1	1781	2020-10-27 22:43:35	1
26490	Water Flow - 2	1156	2020-10-27 22:43:35	1
26491	Water Flow - 3	112	2020-10-27 22:43:35	1
26492	Water Flow - 1	1783	2020-10-27 22:43:36	1
26493	Water Flow - 2	678	2020-10-27 22:43:36	1
26494	Water Flow - 3	66	2020-10-27 22:43:36	1
26495	Water Flow - 1	1796	2020-10-27 22:43:37	1
26496	Water Flow - 2	854	2020-10-27 22:43:37	1
26497	Water Flow - 3	90	2020-10-27 22:43:37	1
26498	Water Flow - 1	1272	2020-10-27 22:43:38	1
26499	Water Flow - 2	705	2020-10-27 22:43:38	1
26500	Water Flow - 3	86	2020-10-27 22:43:38	1
26501	Water Flow - 1	791	2020-10-27 22:43:39	1
26502	Water Flow - 2	759	2020-10-27 22:43:39	1
26503	Water Flow - 3	69	2020-10-27 22:43:39	1
26504	Water Flow - 1	73	2020-10-27 22:43:40	1
26505	Water Flow - 2	562	2020-10-27 22:43:40	1
26506	Water Flow - 3	106	2020-10-27 22:43:40	1
26507	Water Flow - 1	0	2020-10-27 22:43:41	1
26508	Water Flow - 2	288	2020-10-27 22:43:41	1
26509	Water Flow - 3	45	2020-10-27 22:43:41	1

Figure 5.3. Stored values Sample

Assured that the entire system is able to collection, transmission and store information in the envisioned way, the final step was to test the developed processing unit, i.e., the Machine Learning model.

For that, the Random Forest model, in the conditions that obtain the best results, as studied in Chapter 4, was trained and exported using Jupyter, as shown in Figure 5.4. A more visible image of Figure 5.4 is presented in Appendix C.

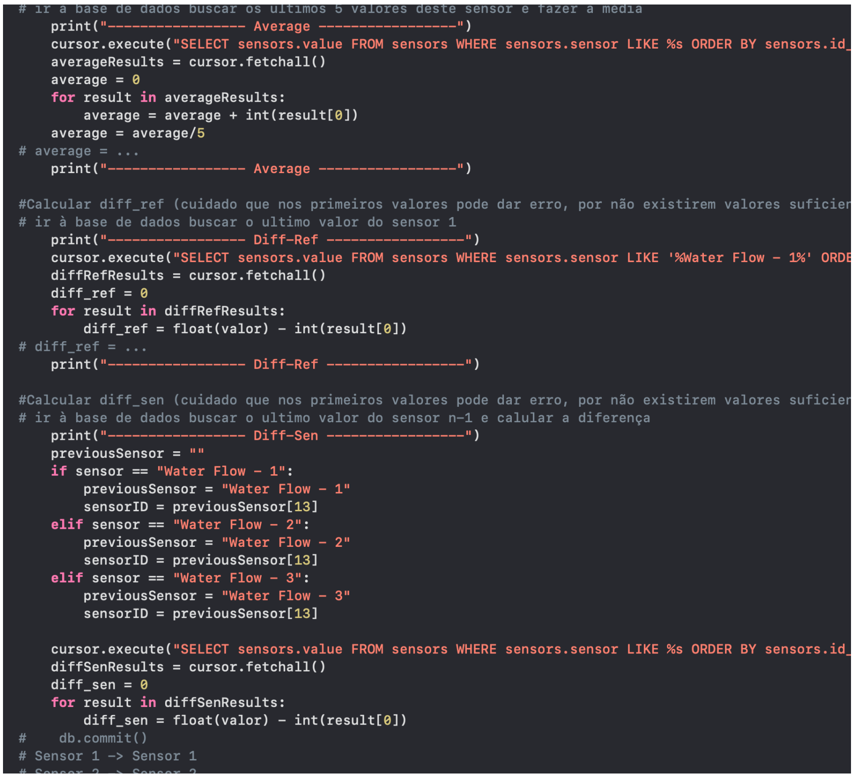


```
In [4]:  import pandas as pd
         import numpy as np
         from sklearn.model_selection import train_test_split
         from sklearn.ensemble import RandomForestClassifier
         from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, roc_auc_score

In [5]:  df = pd.read_csv("datasetleak_individual.csv", sep=";")
         df.head()
         print("Suggested_hour' : ",df['hasProblems'].unique())

In [6]:  #unused Labels
         df = df.drop('timestamp', axis = 1)
         df = df.drop('id', axis = 1)

In [7]:  labels = np.array(df.pop('hasProblems'))
         df.head()

In [8]:  RSEED = 42
         n_estimators = 100

In [9]:  # 30% examples in test data
         train, test, train_labels, test_labels = train_test_split(df,
                                                   labels,
                                                   stratify = labels,
                                                   test_size = 0.30,
                                                   random_state = RSEED)

In [10]: # Imputation of missing values (preencher valores vazios)
         train = train.fillna(train.mean())
         test = test.fillna(test.mean())

In [11]: # Features for feature importances
         features = list(train.columns)

In [12]: #n_jobs=numero de cores que queremos que sejam utilizados( -1 para usar todos os possiveis)
         #criterion, gini ou entrupi(tipo de divisão na arvore)
         #bootstrap, true ou false para melhorar os resultados
         model = RandomForestClassifier(n_estimators=n_estimators,
                                        random_state=RSEED,
                                        n_jobs=-1,
                                        criterion="gini",
                                        max_features='sqrt',
                                        bootstrap=True)

         # Fit on training data
         model.fit(train, train_labels)

In [13]: # Get numerical feature importances
         importances = list(model.feature_importances_)

         # List of tuples with variable and importance
         feature_importances = [(feature, round(importance, 2)) for feature, importance in zip(features, importances)]

         # Sort the feature importances by most important first
         feature_importances = sorted(feature_importances, key = lambda x: x[1], reverse = True)

         # Print out the feature and importances
         [print('Variable: {:20} Importance: {}'.format(*pair)) for pair in feature_importances];

In [14]: # Import matplotlib for plotting and use magic command for Jupyter Notebooks
         import matplotlib.pyplot as plt

         %matplotlib inline

         # list of x locations for plotting
         x_values = list(range(len(importances)))

         # Make a bar chart
         plt.bar(x_values, importances, orientation = 'vertical')

         # Tick labels for x axis
         plt.xticks(x_values, features, rotation='vertical')

         # Axis labels and title
         plt.ylabel('Importance');

         plt.xlabel('Feature');

         plt.title('Classification Feature Importances');

In [15]: # Testing predictions (to determine performance)
         rf_predictions = model.predict(test)
         rf_probs = model.predict_proba(test)[:, 1]

In [16]: model.score(test, test_labels)

In [17]: #metrics.f1_score(y_test, y_pred, average='weighted')

         print(confusion_matrix(test_labels,rf_predictions))
         print(classification_report(test_labels,rf_predictions))
         print(accuracy_score(test_labels, rf_predictions))
         #print(roc_auc_score(y_true = test_labels, y_score = rf_predictions, multi_class = 'ovo', average = 'micro'))

         print("-------")

In [ ]:
```

```
In [13]: # Get numerical feature importances
         importances = list(model.feature_importances_)

         # List of tuples with variable and importance
         feature_importances = [(feature, round(importance, 2)) for feature, importance in zip(features, importances)]

         # Sort the feature importances by most important first
         feature_importances = sorted(feature_importances, key = lambda x: x[1], reverse = True)

         # Print out the feature and importances
         [print('Variable: {:20} Importance: {}'.format(*pair)) for pair in feature_importances];

In [14]: # Import matplotlib for plotting and use magic command for Jupyter Notebooks
         import matplotlib.pyplot as plt

         %matplotlib inline

         # list of x locations for plotting
         x_values = list(range(len(importances)))

         # Make a bar chart
         plt.bar(x_values, importances, orientation = 'vertical')

         # Tick labels for x axis
         plt.xticks(x_values, features, rotation='vertical')

         # Axis labels and title
         plt.ylabel('Importance');

         plt.xlabel('Feature');

         plt.title('Classification Feature Importances');

In [15]: # Testing predictions (to determine performance)
         rf_predictions = model.predict(test)
         rf_probs = model.predict_proba(test)[:, 1]

In [16]: model.score(test, test_labels)

In [17]: #metrics.f1_score(y_test, y_pred, average='weighted')

         print(confusion_matrix(test_labels,rf_predictions))
         print(classification_report(test_labels,rf_predictions))
         print(accuracy_score(test_labels, rf_predictions))
         #print(roc_auc_score(y_true = test_labels, y_score = rf_predictions, multi_class = 'ovo', average = 'micro'))

         print("-------")
```

Figure 5.4. Jupyter Environment

After that, with was imported into the developed Python scripts, in order for them to be able to used the retrieved sensor data form the network to evaluate the system conditions in real time. To allow for that to happen, also the Equations shown in Section 4.1 were included in the scripts, allowing to pre-process the values needed for the algorithm to properly function. Figure 5.5 shows a sample of that script.



```
# ir a base de dados buscar os ultimos 5 valores deste sensor e fazer a media
    print("----------------- Average -----------------")
    cursor.execute("SELECT sensors.value FROM sensors WHERE sensors.sensor LIKE %s ORDER BY sensors.id_
    averageResults = cursor.fetchall()
    average = 0
    for result in averageResults:
        average = average + int(result[0])
    average = average/5
# average = ...
    print("----------------- Average -----------------")

#Calcular diff_ref (cuidado que nos primeiros valores pode dar erro, por não existirem valores suficien
# ir à base de dados buscar o ultimo valor do sensor 1
    print("----------------- Diff-Ref -----------------")
    cursor.execute("SELECT sensors.value FROM sensors WHERE sensors.sensor LIKE '%Water Flow - 1%' ORDE
    diffRefResults = cursor.fetchall()
    diff_ref = 0
    for result in diffRefResults:
        diff_ref = float(valor) - int(result[0])
# diff_ref = ...
    print("----------------- Diff-Ref -----------------")

#Calcular diff_sen (cuidado que nos primeiros valores pode dar erro, por não existirem valores suficien
# ir à base de dados buscar o ultimo valor do sensor n-1 e calular a diferença
    print("----------------- Diff-Sen -----------------")
    previousSensor = ""
    if sensor == "Water Flow - 1":
        previousSensor = "Water Flow - 1"
        sensorID = previousSensor[13]
    elif sensor == "Water Flow - 2":
        previousSensor = "Water Flow - 2"
        sensorID = previousSensor[13]
    elif sensor == "Water Flow - 3":
        previousSensor = "Water Flow - 3"
        sensorID = previousSensor[13]

    cursor.execute("SELECT sensors.value FROM sensors WHERE sensors.sensor LIKE %s ORDER BY sensors.id_
    diffSenResults = cursor.fetchall()
    diff_sen = 0
    for result in diffSenResults:
        diff_sen = float(valor) - int(result[0])
#    db.commit()
# Sensor 1 -> Sensor 1
# Sensor 2 -> Sensor 2
```

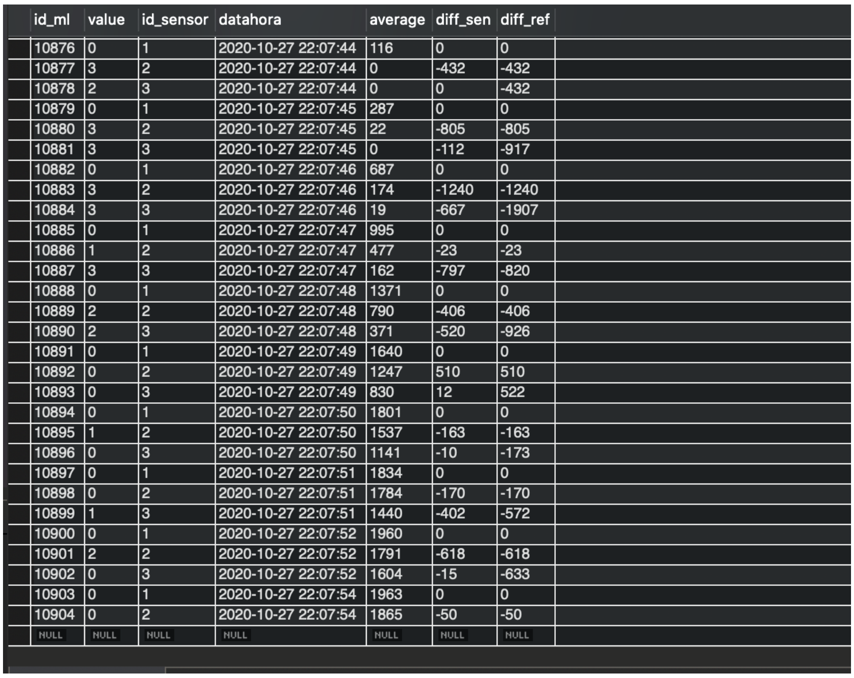
Figure 5.5. Script Sample - Data Pre-Processing

Finally the entire system was tested together, with the machine learning classifying data in real time. Figure 5.6 shows the predicted values, alongside the pre-process data from the equations, as they were stored in the database.

These laboratorial tests allowed us to learn that the planned solution can accomplish its objectives.

5.2. Experimental Implementation

The goal of the proposed developed system was to be fitted on a set of pipelines that supply water in irrigation systems or households, to warn the user when situations such as leaks start to appear, not only to notify the user but also to help prevent this type of situations to evolve to bigger ruptures or other problems, such as water and monetary waste.



id_ml	value	id_sensor	datahora	average	diff_sen	diff_ref
10876	0	1	2020-10-27 22:07:44	116	0	0
10877	3	2	2020-10-27 22:07:44	0	-432	-432
10878	2	3	2020-10-27 22:07:44	0	0	-432
10879	0	1	2020-10-27 22:07:45	287	0	0
10880	3	2	2020-10-27 22:07:45	22	-805	-805
10881	3	3	2020-10-27 22:07:45	0	-112	-917
10882	0	1	2020-10-27 22:07:46	687	0	0
10883	3	2	2020-10-27 22:07:46	174	-1240	-1240
10884	3	3	2020-10-27 22:07:46	19	-667	-1907
10885	0	1	2020-10-27 22:07:47	995	0	0
10886	1	2	2020-10-27 22:07:47	477	-23	-23
10887	3	3	2020-10-27 22:07:47	162	-797	-820
10888	0	1	2020-10-27 22:07:48	1371	0	0
10889	2	2	2020-10-27 22:07:48	790	-406	-406
10890	2	3	2020-10-27 22:07:48	371	-520	-926
10891	0	1	2020-10-27 22:07:49	1640	0	0
10892	0	2	2020-10-27 22:07:49	1247	510	510
10893	0	3	2020-10-27 22:07:49	830	12	522
10894	0	1	2020-10-27 22:07:50	1801	0	0
10895	1	2	2020-10-27 22:07:50	1537	-163	-163
10896	0	3	2020-10-27 22:07:50	1141	-10	-173
10897	0	1	2020-10-27 22:07:51	1834	0	0
10898	0	2	2020-10-27 22:07:51	1784	-170	-170
10899	1	3	2020-10-27 22:07:51	1440	-402	-572
10900	0	1	2020-10-27 22:07:52	1960	0	0
10901	2	2	2020-10-27 22:07:52	1791	-618	-618
10902	0	3	2020-10-27 22:07:52	1604	-15	-633
10903	0	1	2020-10-27 22:07:54	1963	0	0
10904	0	2	2020-10-27 22:07:54	1865	-50	-50
NULL	NULL	NULL	NULL	NULL	NULL	NULL

FIGURE 5.6. Machine Learning Test Sample

Due to the current pandemic situation, the planned real case implementation was not possible, and as such, an experimental implementation was conducted, simulating the real environment.

For that the same scenario that was used for the Machine Learning testing, in Chapter 4 was used for the real environment case. Figure 5.7 shows the system used to mimic the real case scenario, where the sensor node was implemented already using the Machine Learning algorithm to attest the system could locate the water leaks.

The system is composed of:

- **1, 2 and 3:** Water flow sensors;
- **4:** ESP32 board, where all sensors are connected and the Machine Learning algorithm is implemented;
- **5:** Water pump to simulate water flowing.

As in the previous data collection test, the system was left running or several hours, collecting data form the individual sensors as water flows through them. The scenario started with all the pipes in perfect conditions and over time some holes where made in them to start simulating water leaks and see if the algorithm works accordingly with its purpose.

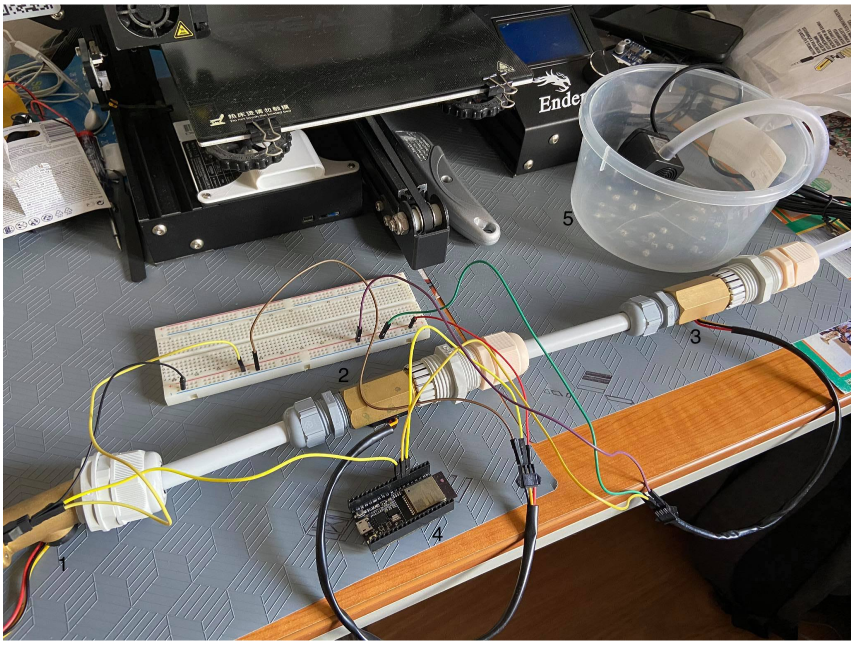
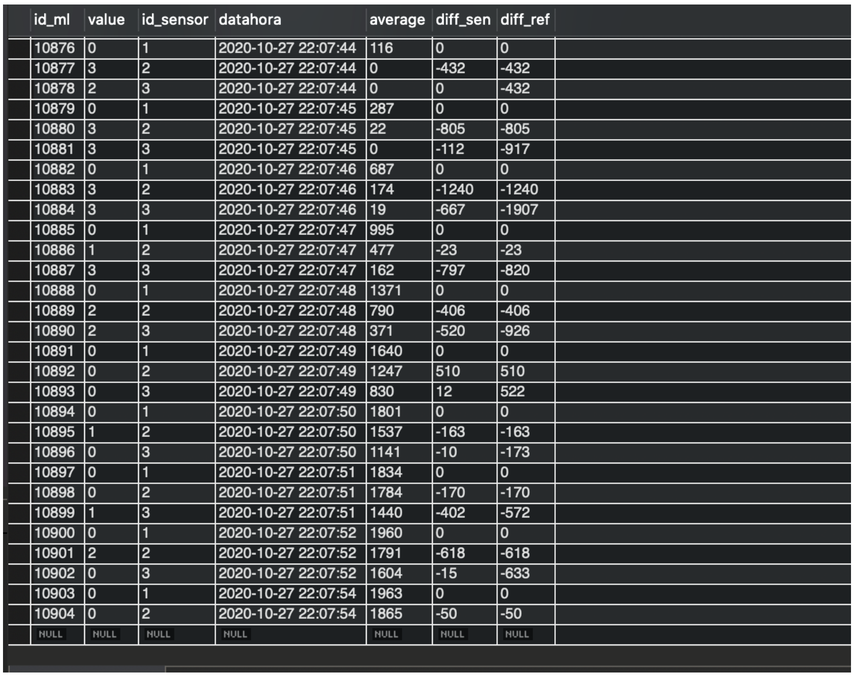


FIGURE 5.7. Test Results

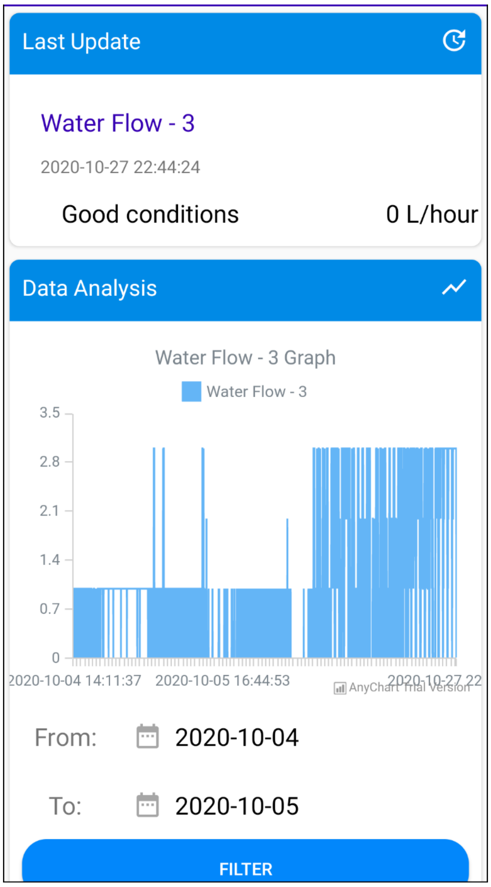
When the data reached the server it was stored and analysed with the machine learning algorithms, giving information in real time about the condition of the pipes. Figure 5.8 shows a sample of the collected data and corresponding data analysis.



id_ml	value	id_sensor	datahora	average	diff_sen	diff_ref
10876	0	1	2020-10-27 22:07:44	116	0	0
10877	3	2	2020-10-27 22:07:44	0	-432	-432
10878	2	3	2020-10-27 22:07:44	0	0	-432
10879	0	1	2020-10-27 22:07:45	287	0	0
10880	3	2	2020-10-27 22:07:45	22	-805	-805
10881	3	3	2020-10-27 22:07:45	0	-112	-917
10882	0	1	2020-10-27 22:07:46	687	0	0
10883	3	2	2020-10-27 22:07:46	174	-1240	-1240
10884	3	3	2020-10-27 22:07:46	19	-667	-1907
10885	0	1	2020-10-27 22:07:47	995	0	0
10886	1	2	2020-10-27 22:07:47	477	-23	-23
10887	3	3	2020-10-27 22:07:47	162	-797	-820
10888	0	1	2020-10-27 22:07:48	1371	0	0
10889	2	2	2020-10-27 22:07:48	790	-406	-406
10890	2	3	2020-10-27 22:07:48	371	-520	-926
10891	0	1	2020-10-27 22:07:49	1640	0	0
10892	0	2	2020-10-27 22:07:49	1247	510	510
10893	0	3	2020-10-27 22:07:49	830	12	522
10894	0	1	2020-10-27 22:07:50	1801	0	0
10895	1	2	2020-10-27 22:07:50	1537	-163	-163
10896	0	3	2020-10-27 22:07:50	1141	-10	-173
10897	0	1	2020-10-27 22:07:51	1834	0	0
10898	0	2	2020-10-27 22:07:51	1784	-170	-170
10899	1	3	2020-10-27 22:07:51	1440	-402	-572
10900	0	1	2020-10-27 22:07:52	1960	0	0
10901	2	2	2020-10-27 22:07:52	1791	-618	-618
10902	0	3	2020-10-27 22:07:52	1604	-15	-633
10903	0	1	2020-10-27 22:07:54	1963	0	0
10904	0	2	2020-10-27 22:07:54	1865	-50	-50
NULL	NULL	NULL	NULL	NULL	NULL	NULL

FIGURE 5.8. Experimental Implementation Values Sample

Figure 5.9 shows the mobile application with the collected data.



Last Update

Water Flow - 3

2020-10-27 22:44:24

Good conditions 0 L/hour

Data Analysis

From: 2020-10-04

To: 2020-10-05

Figure 5.9. Data on Application

5.3. Results & Discussion

After the implementation, the system prototype was put to test, working for several hours during a day, collecting a total of 3703 data entries from three different sensors. The actual test, in a real environment, would extend throughout a longer period of hours and days since irrigation in fields is done at specific times of each day, but as mentioned before, this was not possible so a smaller system was used.

As said, with the implementation done and the Machine Learning algorithm ready, the data was analyzed as soon as it was sent from the ESP32 to the server.

Figure 5.10 presents the collected sensor data with the predicted value for problems.

To evaluate the results obtained, when holes were done in the pipes, the timestamp was annotated, to later be able to compare the values obtained with the reality.

Table 5.1 shows the matrix analysis from the obtained results.

As is possible to see, looking for the diagonal in the matrix, the system is able to detect more correct situations than wrong ones. There are some mistakes between the 0 and 1 values, indicating that for detecting minor leaks the system still needs to be improved. Also some situation were the output was 2 were identified as 3. The more concerning

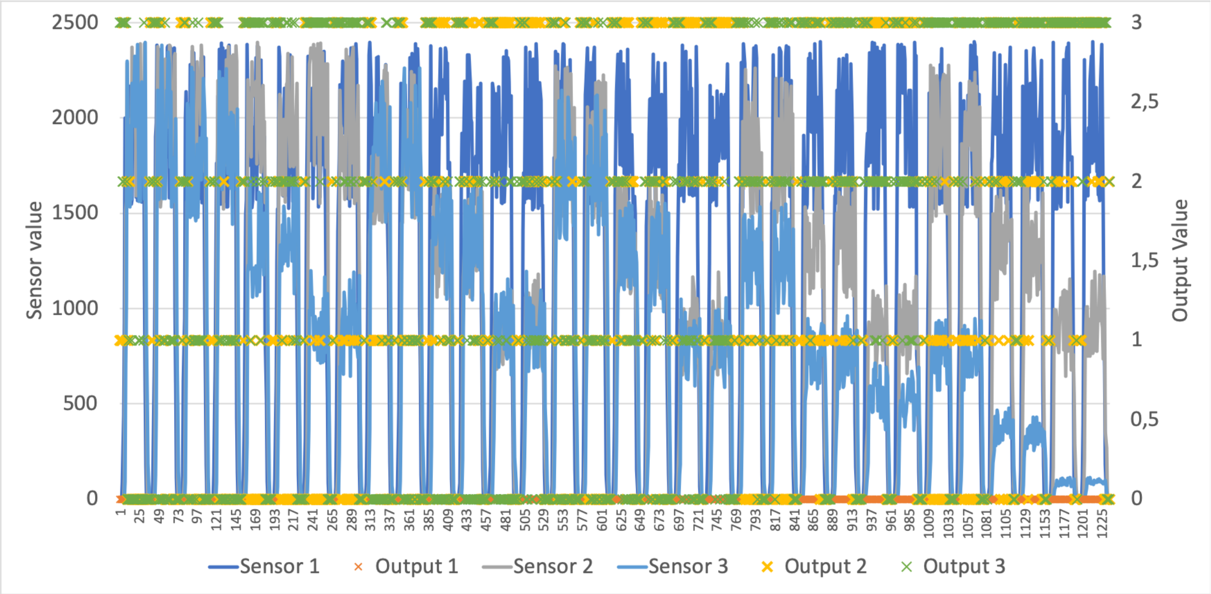


Figure 5.10. Test Reults

Table 5.1. Results Matrix Analysis

		Real Value			
		0	1	2	3
Predicted Value	0	1584	268	123	83
	1	113	179	90	36
	2	63	75	186	74
	3	88	99	224	428

situation is the major leaks that were identified as no problems in 83 cases and situations were no leak is present are 88 times identified as majors leaks.

Overall, the system obtained a 75% accuracy when detecting leaks in the experimental system, which is a good result. Although it is 10% lower, in terms of accuracy, when comparing to the trained algorithm, that can be justified by the small dataset used to train the model, that can lack some of the specific solutions that the system can encounter, or for some environment specification that differ both tests.

Besides this difference, in conclusion a 75% accuracy is still a great result than can help improve the early detection of leaks.

CHAPTER 6

Conclusions and Future Work

6.1. Conclusions

In this dissertation, an IoT system was developed to monitor water distribution systems and to find and locate, with precision, water leaks by using low cost sensors and collecting data in real time. The main goals of the system were achieved and the system proved to be efficient and reliable. Every part of the system was previously and properly tested before reaching the final prototype and its implementation.

Through the implementation made, mostly in a testing environment, it was possible to verify that the communications between sensors were in accordance with the objective. Even though the system was not fully tested with all its components and real case scenario, with what was tested we could verify the system in this initial phase is still reliable. Regarding the server side communication, MQTT proved to be a very reliable and efficient protocol, due to its architecture based in a publisher/subscriber model.

In terms of hardware, the components chosen met the requirements of the system we intended to create. In the microcontrollers, the ESP32 prove to be the right choice with its support for multiple sensors and the sleep mode capability.

Trough the Machine Learning test scenarios it was possible to study and implement an algorithm in the final system, in our case Random Forest, that provided a great accuracy result for the training of the system to predict and locate water leaks.

A mobile application was developed, which allowed the user to interact with the system and analyze in real time the results obtained from the sensors, as well as a graph with the latest value variations. Through the mobile application and the rest of the system, it was possible to give the user all the tools necessary to monitor his water distribution system.

As a final note, it is possible to verify that the developed system meets all the conditions essential for a complete and functional intelligent system and has well that the system is at the level of the other systems already in the market and academic world, with the benefit of being a low cost version but with high quality, efficiency and reliability, which means that in the future it can be adjusted and introduced as a new market solution.

6.2. Future Work

Although all the objectives planned for this system were met, there are always changes that can me made and implemented in order to optimize and take the most out of the system in future work as:

- **Mobile Application:** Develop the app for iOS devices, which would allow for a wider specter of users that can adhere to the system;
- **Sensor types:** Increasing the sensor type, including sensors to measure the water pH for example, that could provide users with other parameters that could be relevant;
- **Actuator nodes:** Add a new node type to the system that could communicate with the other nodes in a bidirectional manner and take actions, such as turning the sensor valves ON or OFF to prevent the pipes with leaks to keep wasting water;
- **Real case scenario:** Test the developed system in a real case scenario to guarantee that the final system is able to work in the real world.

References

[1] A. Al-fuqaha, S. Member, M. Guizani, M. Mohammadi, and S. Member, “Internet of Things : A Survey on Enabling,” vol. 17, no. 4, pp. 2347–2376, 2015. [Online]. Available: http://ieeexplore.ieee.org.proxy.queensu.ca/document/7123563/

[2] H. N. Saha, A. Mandal, and A. Sinha, “Recent trends in the Internet of Things,” in *2017 IEEE 7th Annual Computing and Communication Workshop and Conference, CCWC 2017*, 2017.

[3] S. M. Rabeek, H. Beibei, and K. T. Chai, “Design of wireless iot sensor node platform for water pipeline leak detection,” *Asia-Pacific Microwave Conference Proceedings, APMC*, vol. 2019-Decem, pp. 1328–1330, 2019.

[4] C. Rajurkar, S. R. S. Prabaharan, and S. Muthulakshmi, “IoT based water management,” *Proc. Int. Conf. Nextgen Electronic Technologies: Silicon to Software (ICNETS2)*, pp. 255–259, 2017.

[5] K. Asthon, “That ’ Internet of Things ’ Thing,” *RFID Journal*, 2010.

[6] M. A. Albreem, A. A. El-Saleh, M. Isa, W. Salah, M. Jusoh, M. M. Azizan, and A. Ali, “Green internet of things (IoT): An overview,” in *2017 IEEE International Conference on Smart Instrumentation, Measurement and Applications, ICSIMA 2017*, 2018.

[7] M. Nasiri, N. Tura, and V. Ojanen, “Developing disruptive innovations for sustainability: A review on Impact of Internet of Things (IOT),” in *PICMET 2017 - Portland International Conference on Management of Engineering and Technology: Technology Management for the Interconnected World, Proceedings*, 2017.

[8] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, “Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications,” *Abakos*, 2015.

[9] D. G. Reina, S. L. Toral, F. Barrero, N. Bessis, and E. Asimakopoulou, “The role of ad hoc networks in the Internet of Things: A case scenario for smart environments,” *Studies in Computational Intelligence*, 2013.

[10] C. Bell, *Beginning sensor networks with Arduino and Raspberry Pi*, 2013, vol. 9781430258.

[11] C. Doukas, “The Internet of Things,” in *Building the Internet of Things with the Arduino*, 2012, pp. 8–24.

[12] A. Nayyar and V. Puri, “A review of Arduino board’s, Lilypad’s & Arduino shields,” *2016 International Conference on Computing for Sustainable Global Development (INDIACom)*, pp. 1485–1492, 2016.

[13] A. Maier, A. Sharp, and Y. Vagapov, “Comparative analysis and practical implementation of the ESP32 microcontroller module for the internet of things,” in *2017 Internet Technologies and Applications, ITA 2017 - Proceedings of the 7th International Conference*, 2017.

[14] E. Systems, “ESP32 Series Datasheet,” Tech. Rep., 2019.

[15] A. F. X. d. Glória, "The use of Sensor Networks to create smart environments," 2017. [Online]. Available: https://repositorio.iscte-iul.pt/handle/10071/14628

[16] J. M. Marais, R. Malekian, and A. M. Abu-Mahfouz, "LoRa and LoRaWAN testbeds: A review," in *2017 IEEE AFRICON: Science, Technology and Innovation for Africa, AFRICON 2017*, 2017.

[17] E. Ferro and F. Potortì, "Bluetooth and Wi-Fi wireless protocols: A survey and a comparison," *IEEE Wireless Communications*, vol. 12, no. 1, pp. 12–26, 2005.

[18] A. Hafeez, N. H. Kandil, B. Al-Omar, T. Landolsi, and A. R. Al-Ali, "Smart home area networks protocols within the smart grid context," *Journal of Communications*, vol. 9, no. 9, pp. 665–671, 2014.

[19] J.-s. Lee, Y.-w. Su, and C.-c. Shen, "A Comparative Study of Wireless Protocols :," *IECON Proceedings (Industrial Electronics Conference)*, pp. 46–51, 2007.

[20] E. Systems, "ESPNOW," (visited 05/10/2020).

[21] T. M. Workgroup, "What is it?" no. November, 2015.

[22] N. Chhabra, "Comparative Analysis of Different Wireless Technologies," *International Journal Of Scientific Research In Network Security & Communications*, vol. 1, no. 5, 2013.

[23] W. Manatarinat, S. Poomrittigul, and P. Tantatsanawong, "Narrowband-internet of things (NB-IoT) system for elderly healthcare services," in *Proceeding - 5th International Conference on Engineering, Applied Sciences and Technology, ICEAST 2019*, 2019.

[24] A. Lavric, A. I. Petrariu, and V. Popa, "SigFox Communication Protocol: The New Era of IoT?" in *2019 International Conference on Sensing and Instrumentation in IoT Era, ISSI 2019*, 2019.

[25] ——, "Long Range SigFox Communication Protocol Scalability Analysis under Large-Scale, High-Density Conditions," *IEEE Access*, 2019.

[26] D. Ghosh, A. Agrawal, N. Prakash, and P. Goyal, "Smart saline level monitoring system using ESP32 and MQTT-S," in *2018 IEEE 20th International Conference on e-Health Networking, Applications and Services, Healthcom 2018*, 2018.

[27] Bharati Wukkadada ; Kirti Wankhede ; Ramith Nambiar ; Amala Nair, "Comparison with HTTP and MQTT In Internet of Things (IoT) - IEEE Conference Publication," *2018 International Conference on Inventive Research in Computing Applications (ICIRCA 2018)*, 2018.

[28] N. Tantitharanukul, K. Osathanunkul, K. Hantrakul, P. Pramokchon, and P. Khoenkaw, "MQTT-Topics Management System for sharing of Open Data," in *2nd Joint International Conference on Digital Arts, Media and Technology 2017: Digital Economy for Sustainable Growth, ICDAMT 2017*, 2017.

[29] P. Alqinsi, I. J. Matheus Edward, N. Ismail, and W. Darmalaksana, "IoT-Based UPS Monitoring System Using MQTT Protocols," *Proceeding of 2018 4th International Conference on Wireless and Telematics, ICWT 2018*, pp. 1–5, 2018.

[30] Y. Tan and G. J. Zhang, "The application of machine learning algorithm in underwriting process," in *2005 International Conference on Machine Learning and Cybernetics, ICMLC 2005*, 2005.

[31] R. Saravanan and P. Sujatha, “A State of Art Techniques on Machine Learning Algorithms: A Perspective of Supervised Learning Approaches in Data Classification,” in *Proceedings of the 2nd International Conference on Intelligent Computing and Control Systems, ICICCS 2018*, 2019.

[32] S. Y. M. Mahmoud, “Uav-Cloud : a Platform for Uav Resources and Services on the Cloud,” p. 105, 2015. [Online]. Available: http://scholarworks.uaeu.ac.ae/cgi/viewcontent.cgi?article=1039context=all$_t$*heses*

[33] D. Praveen Kumar, T. Amgoth, and C. S. R. Annavarapu, “Machine learning algorithms for wireless sensor networks: A survey,” *Information Fusion*, 2019.

[34] Q. Tang, X. Ge, and Y. C. Liu, “Performance analysis of two different SVM-based field-oriented control schemes for eight-switch three-phase inverter-fed induction motor drives,” in *2016 IEEE 8th International Power Electronics and Motion Control Conference, IPEMC-ECCE Asia 2016*, 2016.

[35] M. Ross, C. A. Graves, J. W. Campbell, and J. H. Kim, “Using support vector machines to classify student attentiveness for the development of personalized learning systems,” in *Proceedings - 2013 12th International Conference on Machine Learning and Applications, ICMLA 2013*, 2013.

[36] J. Kang, G. S. Member, Y.-j. Park, J. Lee, S.-h. Wang, and D.-s. Eom, “Novel Leakage Detection by Ensemble,” vol. 65, no. 5, pp. 4279–4289, 2018.

[37] M. Mamdouh, M. A. Elrukhsi, and A. Khattab, “Securing the Internet of Things and Wireless Sensor Networks via Machine Learning: A Survey,” in *2018 International Conference on Computer and Applications, ICCA 2018*, 2018.

[38] L. Li, W. Chou, W. Zhou, and M. Luo, “Design Patterns and Extensibility of REST API for Networking Applications,” *IEEE Transactions on Network and Service Management*, 2016.

[39] S. Kim and D. Y. Kim, “Efficient data-forwarding method in delay-tolerant P2P networking for IoT services,” *Peer-to-Peer Networking and Applications*, 2018.

[40] B. Alotaibi and K. Elleithy, “A new MAC address spoofing detection technique based on random forests,” *Sensors (Switzerland)*, 2016.

[41] T. Hastie, R. Tibshirani, and J. Friedman, *Springer Series in Statistics The Elements of Statistical Learning - Data Mining, Inference, and Prediction*, 2009.

[42] A. A. Rezaee and F. Pasandideh, “A Fuzzy Congestion Control Protocol Based on Active Queue Management in Wireless Sensor Networks with Medical Applications,” *Wireless Personal Communications*, 2018.

[43] G. Seif, “A Beginner’s guide to XGBoost,” (visited 15/10/2020). [Online]. Available: https://towardsdatascience.com/a-beginners-guide-to-xgboost-87f5d4c30ed7

[44] Android, “Android,” (visited 14/09/2020). [Online]. Available: https://www.android.com

[45] Apple, “Apple,” (visited 14/09/2020). [Online]. Available: https://www.apple.com

[46] G. Gupta, “Monitoring Water Distribution Network using Machine Learning,” p. 66, 2017.

[47] J. Fikejz and J. Rolecek, “Proposal of a smart water meter for detecting sudden water leakage,” *12th International Conference ELEKTRO 2018, 2018 ELEKTRO Conference Proceedings*, pp. 1–4, 2018.

[48] A. Awwad, M. Yahya, L. Albasha, M. M. Mortula, and T. Ali, “Remote Thermal Water Leakage Sensor With a Laser Communication System,” *IEEE Access*, vol. 8, pp. 163 784–163 796, 2020.

[49] H. Lin, H. Lin, X. Fang, M. Wang, and L. Huang, "Intelligent Pipeline Leak Detection and Analysis System," no. Iccse, pp. 206–210, 2020.

[50] Adafruit, "RFM95W," (visited 12/10/2020). [Online]. Available: https://www.adafruit.com/product/3072

[51] B. n. Roll, "Water Flow Sensor YF-B2," (visited 13/10/2020). [Online]. Available: https://www.botnroll.com/en/biometrics/2543-water-flow-sensor-yf-b2.html

[52] H. M. Server, "HiveMQTT - MQTT Websocket Client." [Online]. Available: http://www.hivemq.com/demos/websocket-client/

Appendices

APPENDIX A

Scientific Paper

Article

Precise Water Leaks Detection using Machine Learning and Real-Time Sensor Data

João Alves Coelho [1,†], André Glória [1,2,†*] and Pedro Sebastião [1,2]

[1] ISCTE - Instituto Universitário de Lisboa, 1649-026 Lisbon, Portugal
[2] Instituto de Telecomunições, 1049-001 Lisbon, Portugal
* Correspondence: afxga@iscte-iul.pt
† These authors contributed equally to this work.

Version November 1, 2020 submitted to IoT

Abstract: This paper presents a proposal for a system based on a wireless sensor network, designed to monitor water distribution systems, such as irrigation systems, which with the help of an autonomous learning algorithm allows to precisely locate the place where the water leak. Also a study to discover the best Machine Learning algorithm to fit the leak detection is presented. The system is described in detail as well as the found results from the machine learning test. Finally a real case implementation is shown to be able to detect leaks with a 75% accuracy.

Keywords: Internet of Things, Green Tech, Water Management, Machine Learning, Sustainability, Water Leaks, Efficiency

1. Introduction

Nowadays, technology takes a major role in our life and we are always searching for better solutions to solve everyday problems. There is always a need for improvement to make it easy to do repetitive tasks, but with less effort.

With this evolution in technology and the increase of devices being used every day, the concept of the Internet of Things (IoT) arise [1]. IoT consists on a network of devices that are capable of being connect to the Internet and communicate between themselves to enact on a common objective. This technology aims for a fast exchange and update of data between devices to obtain an optimum performance [2].

When referring to IoT, it is important to also mentioned the idea of a Wireless Sensor Networks (WSN), an area of the technological world that is essential for the development of intelligent systems. In recent times, IoT and WSN have been a reliable features on the development of monitoring and control systems [3]. The wherewithal of, as mentioned, monitoring and control, extends to a ever so growing range of applications, such as water management.

Water management depends on how well the use of water is maximize and the water losses minimized [4]. This type of management requires precise analysis which may be to complex for humans to perform correctly. With the increase in human population, sustainability and efficient water usage assumes an important role. As water is a scarce resource, detecting problems with the supply and distribution of water as fast as possible can be achieved throughout a sensor network, leading to minimal to no waste in activities such as agricultural irrigation.

Thus, the implementation of a sensor network is paramount to be able to execute those analysis with precision. To improve and optimize the daily management and control of possible leaks, comes up the proposed system, with the main goal of identifying leak points of water systems in an autonomous way.

This paper aims to create a system that can control and monitor water leaks. The system collects data through a low cost sensor network in order to evaluate possible leak points in the water system. The data obtained from the sensors is stored and treated by a Machine Learning (ML) algorithm that will allow the user to be notified if the water distribution system is starting to have leaks, the potential size and the location of those leaks. When a lower water flow is detected, the user is notified and can act upon and take care of the water leak through a mobile app.

This work proposes a new system to monitor and control the water flow through a water distribution pipeline, having in mind the reduction of water wasted and a monetary saving by the final user.

2. Related Work

Water is a crucial natural resource and its widely mishandled. There is an estimate that one third of the world water utilities have a loss of water of around 40% due to leakage [5]. Traditional, pipeline leakage detection requires periodical inspection with human involvement, which makes it slow and inefficient for timely and fast detection. This leads to an increase in the development of methods to detect, locate, and fix leaks and its something we can see in numerous different projects.

In [6] the authors created a system to inform and control the water usage in households and buildings. The system, a WSN, uses different types of sensors to measure the water flow along the plumbing system. The system can detect water leaks, but only if there is a sudden decrease in the water flow and it cannot identify where the leakage occurred, only that it occurred.

The authors of [7] also developed a WSN with thermal sensors to detect water leaks in the pipes. In this case, the temperature of the soil above the leakage will change, so it is possible to detect with some assurance the location of the leak in the pipe. The problem with this system lies with the inevitable difference of soils around the world. It is easier to detect temperature changes of the soil on a sand type of soil than on concrete or brick.

In [8] it is presented a system of NB-IoT using ultrasonic sensors to detect, with precision, the location of leaks in the pipelines. The system uses the sensors to detect the minimal change of water passing by the pipes and when detected, inform the user where the leak happened. However, this system requires expensive sensors which creates a system that it is not accessible to all users.

We can see that there are already various types of WSN solutions for water leak control and detection, but all with some limitations that have prejudice its efficiency in precisely detecting the leaks. By using low cost hardware combined with ML and a communication protocol our proposal aims to solve this problem.

3. Materials and Methods

The achieve the main objective of detecting leaks in pipes in real time using machine learning, there was a need to design a control and monitoring system to be applied in water distribution pipelines, in public and private domain. This system intends to use multiple sensors with real time data collection, mainly water flow parameters to improve the efficiency and the early detection of leaks, with the support of machine learning techniques. Figure 1 shows a possible implementation of the system in a drip irrigation system.

As can be seen in Figure 2, the system is divided in multiple modules, with both software and hardware parts, each with its individual purpose, from the hardware nodes for data collection, the server for data processing, and finally the user.

The system consists of various water flow measuring sensors, spread throughout the water distribution pipeline that will collect information as water flows by them. This information is then sent to the aggregation node, the main hub of our system and that will be responsible to communicate with the server and send to it the information from the sensors. In the server the information will be saved and will also run through the ML algorithm to be studied and interpreted. From here, depending on the analysis done by the algorithm the information is shown to the user as being all

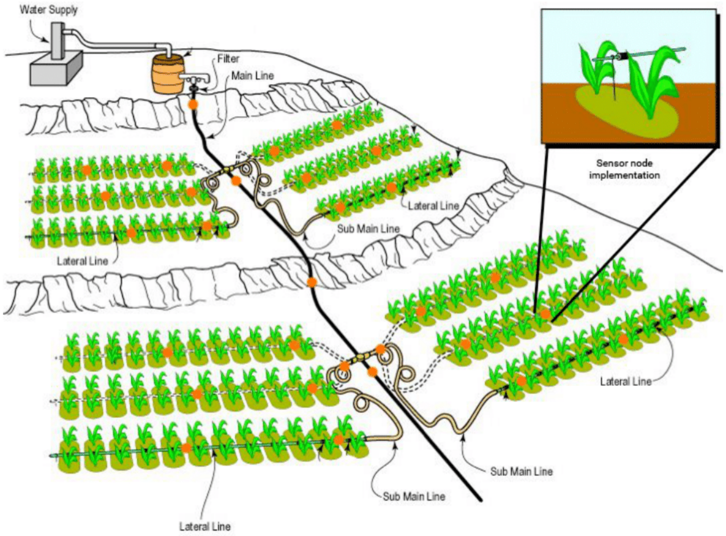


Figure 1. System implementation

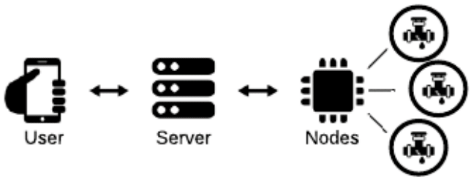


Figure 2. System Architecture

normal within the system or will alert the user for potential locations of water leaks. This logic can be better comprehended in Figure 3.

3.1. Data Collection

The developed system confides in a network of sensors, as showed in Figure 2, that conjointly create a typical WSN, capable of retrieving data from the sensors and send them to a cloud server.

Each of the nodes was created in a way to get the maximum efficiency, low cost and low power with the best microcontroller, communication protocols and supplementary hardware being used. For that, the ESP32 microcontroller was used as the basis of each node, due to its dual core chip which includes Wi-Fi and BLE, 32 GPIO ports, that can be assigned different programming functions, 12 analog ports, and low power capabilities, with the ability to enter into a deep sleep state. [9,10]. Another core feature and hardware the nodes share is the ability to communicate via an LoRa peer-to-peer network using the RFM95W LoRa transceiver, that grants a long range spectrum communication and immunity to high frequency, all while reducing power consumption. It supports high performance FSK modes for systems and delivers exceptional noise, selectivity, receiver linearity for a much lower current consumption than other devices [11].

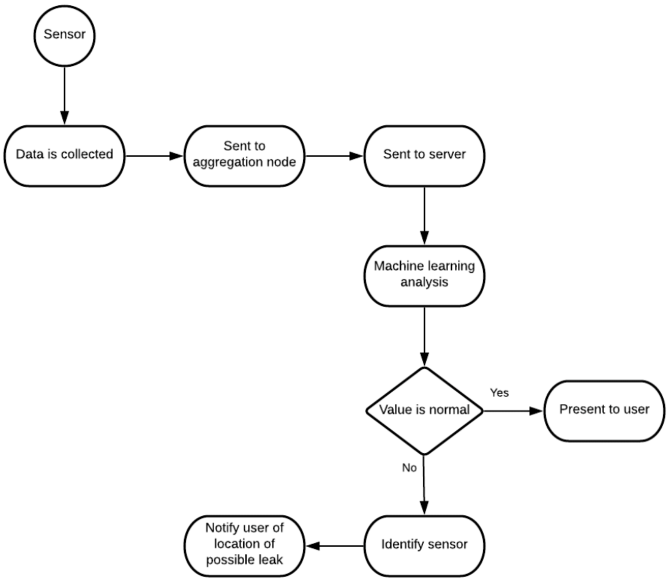


Figure 3. System Logic

The specification for each node and additional hardware are explained in the following sections.

3.1.1. Aggregation Node

The aggregation node is the central node of the network and the one responsible for keeping the network connected. The aggregation node does not collect any data, just sends the information it receives to the server. In our system it works as a bridge between the sensors, that gather the data, and the server that stores the data.

In Figure 4 is possible to see how the aggregation node of our system is built.

As said, the aggregation node receives messages from the other nodes via LoRa. For the server communication, Message Queue Telemetry Transport (MQTT) is used, via a NB-IoT connection, using the SIM7000E module. Communication will be detailed ahead.

3.1.2. Sensor Node

Sensor node is the simplest and the lowest level of a WSN. Their sole purpose is to collect data from the attached sensors and send it to the aggregation node. In order to perform these tasks, the sensor node needs a microcontroller and a set of sensors that can be permuted from node to node, depending on the different needs for the solution.

In Figure 5 is represented the sensor node.

The sensor nodes consists of, as mentioned before, an ESP32 microcontroller, a RFM95W module and an array of sensors suited for retrieving information and send it to the aggregation node. To transmit the information collected from the sensors to the aggregation node, we make use LoRa through the RFM95W module that allows for a much lower power consumption.

Seeing as the node only needs to be awaken when water is flowing by, we can power the sensor node with a battery which allows for a completely wireless node and a longer lifetime. And with the only goal of measuring the water flow of the water pipelines, we just need one type of sensor,

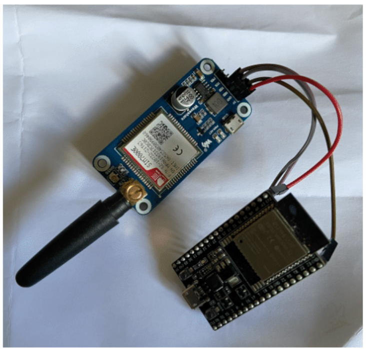


Figure 4. Aggregation Node

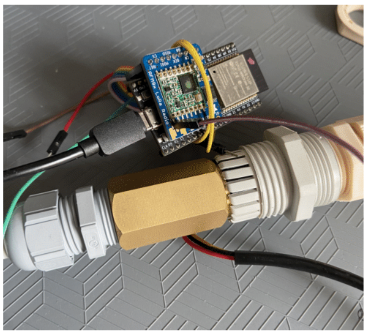


Figure 5. Sensor Node

distributed through the sensor nodes to retrieve the information from the water distribution pipeline. For this, the used sensor to perform this task was the YF-B2 [12]. This sensor consists of a water rotor and a hall-effect sensor. When water flows through it the rotor rolls and its speed changes with the different rate of flow.

3.2. Communications

Communication is the most crucial part of a WSN as it is responsible for transmitting the data collected from the sensors but also to keep the network connected and to assure that the system abides to the tasks that were specified. Communication also helps to improve the area that can be covered, with some protocols allowing the network to expand in terms of area size.

The projected system, composed by two sets of nodes, it will require two types of communication protocols, one for the communication between nodes and the aggregation node and the other for the communication between the aggregation node and the server.

3.2.1. Node-to-node communication

The node-to-node communication is how the information travels inside the WSN. Because sensors can be far from each other and so are power by batteries, a communication protocol capable of long range transmission and low power consumption is essential.

With this in mind and in order to have all the nodes in the network connected LoRa was implemented, using a RMF95W module. LoRa is a physical layer that uses CSS modulation, with the same low power characteristics of the FSK modulation (which is part of many other communication protocols) but with an increase in range. [9,13]. It uses a star topology which allows for a lower power consumption and a reduction in the network complexity and capacity.

To implement LoRa connection in the node, the RadioHead library was used [14], which enables node addressing, with unique addresses being set for each node, guaranteeing differentiation in the network. With the RadioHead library, the nodes in the network can communicate in two different ways. By sending a broadcasting message to every node with the destination ID included in the message and by directly messaging the node by using its destination address.

3.2.2. Node-to-server communication

The node-to-server communication is how the information containing the data gathered from the sensors leaves the network. To connected with a server outside of the network, it requires some kind of internet connection and this is were NB-IoT was used.

NB-IoT is a standard based LR-WAN system, developed to allow devices to connect through mobile phone signals [15]. It uses devices with low data rate that take advantage of the cellular network to communicate between each other but with the convenience of consuming less energy. The main focus of NB-IoT is improving IoT solutions coverage with low cost devices, battery life for up to 10 years and high connectivity.

In order to always maintain a connection between the network and the server the MQTT protocol was selected based on its architecture that fits IoT projects and for its lower power consumption's when facing typical HTTP requests.

MQTT is a communication protocol built on top of TCP protocol, with a low complexity and with the main goal of connecting devices to embedded networks. Is a many-to-many type of communication and it consists of three components: publisher, broker and subscriber [16]. The publisher, to communicate via MQTT, sends a message with two components: a topic and a message. The broker receives the message and distributes it between multiple devices. This devices, registers as the subscribers for the specific topics of a publisher, in order to gather the data [17].

In this publish/subscribe model, the subscriber can only receive messages from the topic they have subscribed to. This is ensured by the broker that also makes available the different topics. Contrary to HTTP, with this type of publish/subscribe model, is possible for several publishers to communicate at the same time, so that one node does not need to end its connection with the server for other to make a new connection [18]. Since it provides a simple implementation and provides routing for low power, low cost and low memory devices in low throw low bandwidth networks [19], is one of the most suitable connection protocols for IoT.

For this, the PubSubClient [20] library was used to implement the MQTT on the aggregation node.

3.3. Data Analysis

For the system to be able to collect and analyse in real time the data that is sent by the nodes, a set of scripts were developed. The proposed goal of the pretended system is to identify in real time, the locations of possible water leaks and for that we make use of Machine Learning.

The first step for the system to be able to predict leaks is to receive the information from the sensors and pre-process them, including saving the values and corresponding timestamps in a database, for future analysis and to feed information to the mobile application. The pre-processing consists on a set of pre-validations and calculations, such as guarantee the values are within the expected and calculate averages and fluctuation between sensors.

For that a Python script was developed, using the Paho Python MQTT library [21], to be able to receive the values from the network.

Afterwards, the scripts puts the received values as inputs in a Machine Learning algorithm to predict the final output and warn the user if needed. This was implemented in the same script, and the algorithm chosen for this task was previously trained and fitted as described in the next Section.

3.4. Data Visualization

To provide the user a way to see the data collected from the sensors of the system, an Android mobile application was developed from scratch. The objective of the mobile application was to give the user a dashboard of the water distribution system, where it is possible to check all the values from the sensors in real time, everywhere.

Regarding the values retrieved from the sensor nodes, the application gives the user the opportunity to check in real time, not only the latest values, but also the current conditions of the corresponding pipes, as evaluated by the Machine Learning algorithms, as well as historical data and conditions. For the real time system, the Paho Java MQTT library [21] was used to subscribe to the topic the network publish the information.

If the user does not check the application on a regular basis, it will warn the user via a notification if some problem is encountered by the system.

In order for the user to have a visual interaction with the system and to be able to perform the described functionalities, the developed Android application is composed by a set of screens. Figure 6 presented the developed screens.

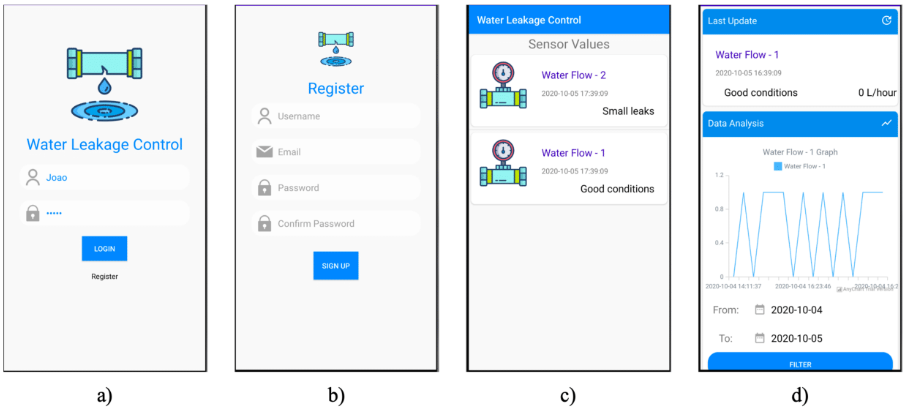


Figure 6. Android Screens: a) Login, b) Register, c) Dashboard, d) Sensor detail

4. Machine Learning Study

As previously said, the central part of the system is the machine learning algorithm, which is used to predict water leaks and inform the user of the leak location. To be able to do this, the algorithm must first be trained to understand the system and the environment so that in case a water leak happens, the system may know what is happening and what to do.

One of the goals of this paper was to understand what was the best algorithm to use in our system. For that five different classification algorithms were tested, to check which as the best accuracy.

For that the following algorithms were tested:

1. **Support Vector Machine (SVM)** - Used mostly for classification, it classifies the data by building n dimensions between two classes and finding an optimal hyperplane to categorize the data, using the distance between the neighboring points and differentiating between the classes with minimum error margin [22]. In a more simple explanation, given training data, the algorithm outputs the best hyperplane that classifies new examples [23].
2. **Decision Trees (DT)** - Through an hierarchical partition of training data, a certain feature is used to split the data, with this split being done iteratively till the leaf node contains a number of records that can be used to classify the data [24,25]. However, as described in [26] this algorithm faces some limitations because a small change in the training dataset can lead to a substantial change in the tree, making it harder to predict the next values with precision.
3. **Random Forest (RF)** - Best applied for classification problems, integrates the process of aggregation bagging and DT, by choosing a subset of features from an individual nodes of the tree and that way it avoids the correlation on the boostrapped set [25], working with an assortment of trees, in which each tree gives a classification.
4. **Neural Networks (NN)** - Functions identical to the nervous system, using neurons in various layers in bio-system like shape. Each of the neurons analyzes parts of the input and sending the information to the next layer and neurons continuously, until its able to reach a valid output [24]. Ideally used in non linear and complex problems which requires a big computational power which is not fitting for data from IoT systems [25]. In our case, we used MLP a variation of NN algorithm which consists of multiple neurons organized into layers [27].

5. **XGBoost** - With a similar model to DT, the objective of this algorithm is, as the name suggests, boosting the performance of the model. It creates a sequence of models and rather than training all individually, it models in succession such that these models attempt to correct the mistakes of the models before [28]. The first model is built on original dataset, with the second model improving the first model, the third model improving the second, and so on. The models are added sequentially until no further improvements can be made.

4.1. Dataset Creation

In order to train the algorithm, first we need to create a dataset with similar data to the one that will be used in the future, containing all possible outputs. As if an output never happens in the dataset, the algorithm will not be able to predict it. In the intended system the ML algorithm will only be used for analysing a specific scenario: water leaks.

For this scenario, and in order to create the needed dataset, a small water distribution system was created in which our system was implemented, as shown in Figure 7.

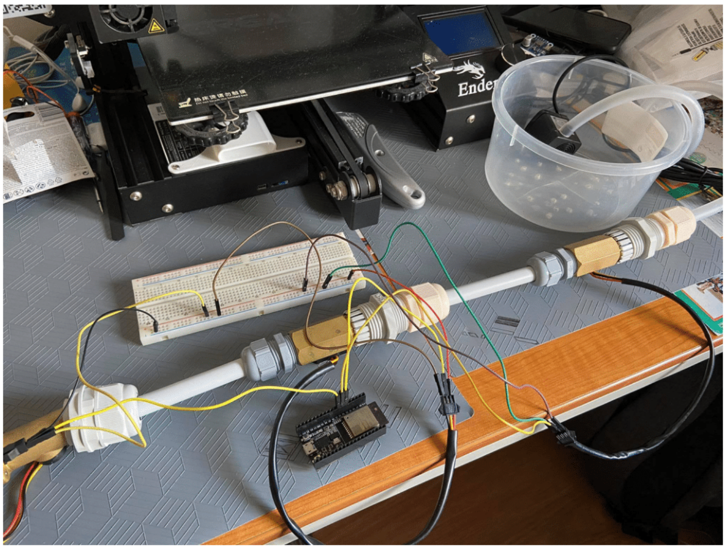


Figure 7. System Prototype

The system collected data every 5 minutes, about 12 data samples an hour, with all samples being recorded with a value collected and the corresponding pipe to be able to classify the collected information, for all possible outputs.

To achieve this outputs, in the test scenario, different holes were carved in the pipes, with size variances and with pipes with more holes than others, to simulate a rupture in the system and match to the desire output.

Table 1 shows the desired outputs and the meaning of each for the system.

Table 1. Intended Outputs

Output	Meaning
0	No leaks in the system
1	Micro leaks in the system
2	Minor leaks in the system
3	Major leaks in the system

In order to have a good dataset, that allow for the proposed Machine Learning methodology to work properly, for each of the output possibilities, multiple tests were performed, changing the duration of the test, pressure applied by the pump or length of the pipes, each of these being also performed multiple times to guarantee that small changes in data are recorded.

These tests allow us to obtain 5607 records containing information from three sensors that were spread through the same path of pipe, and which analyse the conditions of the pipes in two different zones. Since multiple situations and scenarios were applied to this data collection, it is possible to assume a good reliability on the data collected and that will train the algorithms, so it can be applied to other scenarios.

Although the collected data shows information from multiple sensors, each individual record only has the timestamp and the data collected from the sensor. So, the first thing that we intend to analyze was how the data that is inputted into the algorithm can affect its accuracy. For that two datasets were created, using the same collected data. One is the data collected according to the sequence of sensors in the system and the other where the data was clustered, to show how the entire path of pipes is working. This was done in order to understand which of the datasets presented the best results for the intended scenario and system.

When creating the dataset, besides the timestamp and collected sensor data, other features were added to each record, based on calculations and pre-processed data from the sensor values.

Table 2 presents the fields of each dataset.

Table 2. Dataset Features

Feature	Description
id	ID of the data input
$sensorID_i$	Corresponding sensor
$value_i$	Value collected
$average_i$	Average from last 5 values for that sensor
$diff\_ref_i$	Difference from Sensor 1
$diff\_sen_i$	Difference from previous sensor
$hasProblems_i$	Final output

The difference between dataset, is that the Standard one only has one entry for each of the features, while the Clustered one has three entries for *sensorID, value, average, diff_ref* and *diff_sen*, each one regarding the three sensors used in the test.

With this, the two dataset created have 5607 and 1869 entries, for the standard and clustered respectively.

In order to reach the goal of detecting water leaks, the algorithms must be trained and tested with both datasets. This guarantees that when used in a real life scenario the results will be precise. And for that, the datasets were divided into a train and test, with 70% and 30%, respectively.

4.2. Model Analysis

For the model analysis, an array of ML algorithms were used in order to determine which is the best one to use in our system.

For our scenario a total of 12 tests for each algorithm was performed, each test with different parameters, with the objective to train the algorithms to understand which algorithm presented the best accuracy so that it could be implemented in our system.

Table 3 shows all the performed tests, identifying the used dataset, target and unused features.

Table 3. Test Scenarios

Dataset	Test N°	Target	Unused Features
Clustered	1	hasProblems2	average, diff_sens, diff_ref
	2	hasProblems3	
	3	hasProblems2	diff_sens, diff_ref
	4	hasProblems3	
	5	hasProblems2	diff_ref
	6	hasProblems3	
	7	hasProblems2	-
	8	hasProblems3	
Standard	9	hasProblems	average, diff_sens, diff_ref
	10		diff_sens, diff_ref
	11		diff_ref
	12		-

In these test cases, the target output is related to if a specific sensor has problems or if everything is working correctly in the system, with "*hasProblems2*" or "*hasProblems3*", for the clustered dataset, indicating a problem on section 2 or 3 respectfully, and as for the standard dataset, "*hasProblems*", indicating a problem on that specific section.

Each of the tests was conducted using Python, the scikit-learn library [29] and the Jupyter platform. For each of the algorithms a script was developed using the corresponding library for classification, from scikit-learn, ad the default configurations were used. As said, 70% of the dataset was used for training and 30% for testing.

Table 4 presents the results for each test.

Table 4. Tests Results

Algorithm	Accuracy [%]											
	1	2	3	4	5	6	7	8	9	10	11	12
RF	75.40	70.05	80.75	72.73	80.75	73.62	80.57	74.69	74.51	77.00	77.82	84.79
DT	76.65	70.94	80.21	71.30	80.04	72.19	79.14	70.05	74.03	77.06	77.42	84.02
NN	59.71	60.42	67.74	67.91	75.40	70.23	73.44	72.19	51.69	55.44	74.98	80.68
SVM	26.91	26.56	46.17	46.60	51.34	31.02	57.22	41.35	53.77	51.75	29.47	27.09
XGBoost	76.65	70.94	80.93	75.22	81.46	75.76	81.11	75.04	74.99	77.01	77.78	84.73

Figure 8 represents the same results, allowing for a better analysis.

4.3. Remarks

The objective of all of these tests was to determine which had the best accuracy when given inputs related to our water leakage system, as well as check which was the best dataset composition. With the results from the perform tests it is take conclusions and define the best algorithm and dataset for our intended system.

As it is possible to attest, the datasets have similar results but with different behaviours, with the accuracy for most of the algorithms being within the same range, of about 70% to 85%, excluding SVM, that for both datasets presented an accuracy within the range of 27% to 53%.

We can also assure that the variables have an important role for the output of each test, always having an influence on the algorithm that is tested. In the clustered dataset, for the algorithms RF and DT, we can see that from the first to the second test for the same target, so test 1 and 3 for "*hasProblems2*" and test 2 and 4 for "*hasProblems3*", the tendency is for the accuracy to increase with less unused features. This behaviour then changes, and for tests 3 to 5 and 7 for "*hasProblems2*" and test 4 to 6 and 8 for "*hasProblems3*" the accuracy decreases, concluding that this algorithms work better when the number of unused features is bigger. For the algorithms NN, SVM and XGBoost, the behaviour is the same for all the targets when the unused features decrease, with all the algorithms increasing

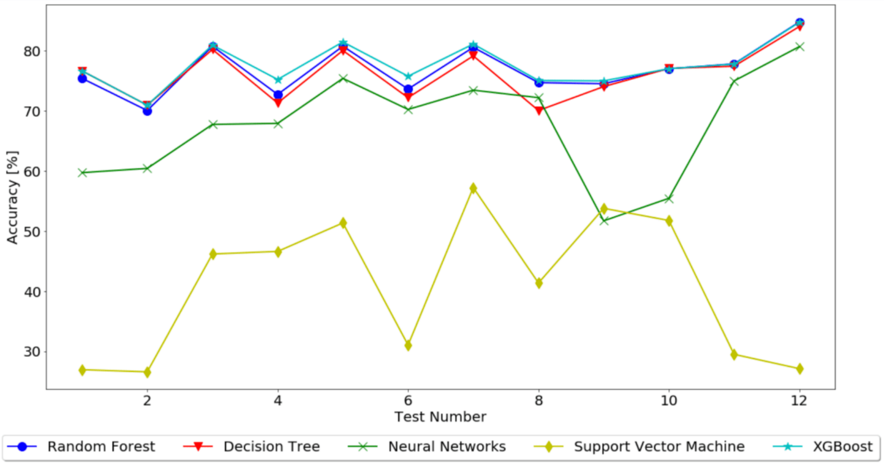


Figure 8. Test Results

its accuracy from test to test, thus concluding that this algorithms work better with the lesser unused features. For the standard dataset, all the tests have the accuracy increasing in each test with lesser unused features, with the exception being SVM, where the accuracy decreases when the number of unused features also decreases.

As we can see from the results, SVM showed the worst results in almost all the tests with most results being below 50% accuracy, so it can be discarded as the algorithm to used.

The NN algorithm, even though being better than SVM with no outputs lower than 50% accuracy, it still got final results less than expected and intended.

RF, DT and XGBoost presented very similar final outputs in terms of accuracy, with no results below 60% accuracy and with a few final outputs above 80%.

Is is also important to mentioned that, for the RF, DT and XGBoost, the accuracy increases depending on fewer parameters being dropped from the dataset

So, with what is showed from the results of all the tests is data the best dataset is the standard one, where there is no differentiation from the data, and the best algorithm is the RF, with a best accuracy result of, approximately, 85%, when using all the features on the dataset.

5. Experimental Implementation

The goal of the proposed developed system was to be fitted on a set of pipelines that supply water in irrigation systems or households, to warn the user when situations such as leaks start to appear, not only to notify the user but also to help prevent this type of situations to evolve to bigger ruptures or other problems, such as water and monetary waste.

Due to the current pandemic situation, the planned real case implementation was not possible, and as such, an experimental implementation was conducted, simulating the real environment.

For that the same scenario that was used for the training was used in the experimental scenario. As in the previous data collection test, the system was left running or several hours, collecting data from the individual sensors as water flows through them. The scenario started with all the pipes in perfect conditions and over time some holes where made in them to start simulating water leaks and see if the algorithm works accordingly with its purpose.

6. Results & Discussion

After the implementation, the system prototype was put to test, working for several hours during a day, collecting a total of 3703 data entries from three different sensors. The actual test, in a real environment, would extend throughout a longer period of hours and days since irrigation in fields is done at specific times of each day, but as mentioned before, this was not possible so a smaller system was used.

As said, with the implementation done and the Machine Learning algorithm ready, the data was analyzed as soon as it was sent from the ESP32 to the server.

Figure 9 presents the collected sensor data with the predicted value for problems.

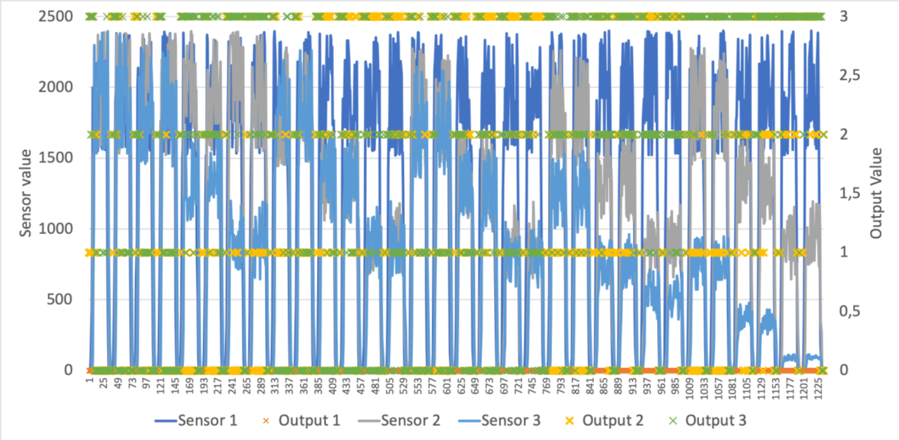


Figure 9. Test Reults

To evaluate the results obtained, when holes were done in the pipes, the timestamp was annotated, to later be able to compare the values obtained with the reality.

Table 5 shows the matrix analysis from the obtained results.

Table 5. Results matrix analysis

		Real Value			
		0	1	2	3
Predicted Value	0	1584	268	123	83
	1	113	179	90	36
	2	63	75	186	74
	3	88	99	224	428

As is possible to see, looking for the diagonal in the matrix, the system is able to detect more correct situations than wrong ones. There are some mistakes between the 0 and 1 values, indicating that for detecting minor leaks the system still needs to be improved. Also some situation were the output was 2 were identified as 3. The more concerning situation is the major leaks that were identified as no problems in 83 cases and situations were no leak is present are 88 times identified as majors leaks.

Overall, the system obtained a 75% accuracy when detecting leaks in the experimental system, which is a good result. Although it is 10% lower, in terms of accuracy, when comparing to the trained algorithm, that can be justified by the small dataset used to train the model, that can lack some of the specific solutions that the system can encounter, or for some environment specification that differ both tests.

Besides this difference, in conclusion a 75% accuracy is still a great result than can help improve the early detection of leaks.

7. Conclusions

In this paper, an IoT system was presented capable of monitor water distribution systems and to find and locate, with precision, water leaks by using low cost sensors and collecting data in real time. The main goals of the system were achieved and the system proved to be efficient and reliable. Every part of the system was previously and properly tested before reaching the final prototype and its implementation.

Trough the ML test scenarios it was possible to study which is the best algorithm to use in this scenario. Multiple models were tested, in various configurations and different datasets, in order to achieve the best configuration possible. Not only it was possible to conclude that when using more features the accuracy increases but also that Random Forest achieves the best accuracy in almost every scenarios, making it the best Classification algorithm to use, with almost 85% accuracy.

Besides a theoretical implementation and the machine learning study, also an experimental implementation was presented, to validate the methodology used and the results obtained in our study, in a real case scenario.

In that, the system was able to detect with a 75% accuracy the presence of leaks in a pipeline. Although the results obtained are lower than those obtained in the laboratory test phase, it is possible to conclude that our system is able to help prevent water loses and pipes malfunction. The lower results indicate that the system still needs some improvements and further testing.

As a final note, it is possible to verify that the developed system meets all the conditions essential for a complete and functional intelligent system and has well that the system is at the level of the other systems already in the market and academic world, with the benefit of being a low cost version but with high quality, efficiency and reliability, which means that in the future it can be adjusted and introduced as a new market solution.

Author Contributions: conceptualization, J.A.C, A.G. and P.S.; methodology, A.G.; software, J.A.C and A.G.; validation, A.G.; formal analysis, A.G.; investigation, J.A.C and A.G.; resources, A.G.; data curation, A.G.; writing–original draft preparation, J.A.C and A.G.; writing–review and editing, P.S.; visualization, A.G.; supervision, P.S.; project administration, A.G.; funding acquisition, A.G. and P.S

Funding: This work was supported in part by ISCTE - Instituto Universitário de Lisboa from Portugal under the project ISCTE-IUL-ISTA-BM-2018

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Al-fuqaha, A.; Member, S.; Guizani, M.; Mohammadi, M.; Member, S. Internet of Things : A Survey on Enabling. *IEEE Communications Surveys & Tutorials* **2015**, *17*, 2347–2376.
2. Saha, H.N.; Mandal, A.; Sinha, A. Recent trends in the Internet of Things. 2017 IEEE 7th Annual Computing and Communication Workshop and Conference, CCWC 2017, 2017. doi:10.1109/CCWC.2017.7868439.
3. Rabeek, S.M.; Beibei, H.; Chai, K.T. Design of wireless iot sensor node platform for water pipeline leak detection. *Asia-Pacific Microwave Conference Proceedings, APMC* **2019**, *2019-Decem*, 1328–1330. doi:10.1109/APMC46564.2019.9038809.
4. Rajurkar, C.; Prabaharan, S.R.S.; Muthulakshmi, S. IoT based water management. *Proc. Int. Conf. Nextgen Electronic Technologies: Silicon to Software (ICNETS2)* **2017**, pp. 255–259. doi:10.1109/ICNETS2.2017.8067943.
5. Gupta, G. Monitoring Water Distribution Network using Machine Learning. Master's thesis, KTH ROYAL INSTITUTE OF TECHNOLOGYSCHOOL OF ELECTRICAL ENGINEERING, 2017.

6. Fikejz, J.; Rolecek, J. Proposal of a smart water meter for detecting sudden water leakage. *12th International Conference ELEKTRO 2018, 2018 ELEKTRO Conference Proceedings* **2018**, pp. 1–4. doi:10.1109/ELEKTRO.2018.8398316.

7. Awwad, A.; Yahya, M.; Albasha, L.; Mortula, M.M.; Ali, T. Remote Thermal Water Leakage Sensor With a Laser Communication System. *IEEE Access* **2020**, *8*, 163784–163796. doi:10.1109/access.2020.3022213.

8. Lin, H.; Lin, H.; Fang, X.; Wang, M.; Huang, L. Intelligent Pipeline Leak Detection and Analysis System. 2020 15th International Conference on Computer Science & Education (ICCSE), 2020, pp. 206–210. doi:10.1109/iccse49874.2020.9201761.

9. Glória, A.; Sebastião, P.; Dionísio, C.; Simões, G.; Cardoso, J. Water management for sustainable irrigation systems using internet-of-things. *Sensors (Switzerland)* **2020**, *20*. doi:10.3390/s20051402.

10. Maier, A.; Sharp, A.; Vagapov, Y. Comparative analysis and practical implementation of the ESP32 microcontroller module for the internet of things. 2017 Internet Technologies and Applications, ITA 2017 - Proceedings of the 7th International Conference, 2017. doi:10.1109/ITECHA.2017.8101926.

11. Adafruit. RFM95W. Online, available at https://www.adafruit.com/product/3072.

12. Botn'Roll. Water Flow Sensor YF-B2. Online, available at https://www.botnroll.com/en/biometrics/2543-water-flow-sensor-yf-b2.html.

13. Marais, J.M.; Malekian, R.; Abu-Mahfouz, A.M. LoRa and LoRaWAN testbeds: A review. 2017 IEEE AFRICON: Science, Technology and Innovation for Africa, AFRICON 2017, 2017. doi:10.1109/AFRCON.2017.8095703.

14. Airspayce. RadioHead Packet Radio library for embedded microprocessors, 2019. Online, available at http://www.airspayce.com/mikem/arduino/RadioHead/index.html.

15. Manatarinat, W.; Poomrittigul, S.; Tantatsanawong, P. Narrowband-internet of things (NB-IoT) system for elderly healthcare services. Proceeding - 5th International Conference on Engineering, Applied Sciences and Technology, ICEAST 2019, 2019. doi:10.1109/ICEAST.2019.8802604.

16. Ghosh, D.; Agrawal, A.; Prakash, N.; Goyal, P. Smart saline level monitoring system using ESP32 and MQTT-S. 2018 IEEE 20th International Conference on e-Health Networking, Applications and Services, Healthcom 2018, 2018. doi:10.1109/HealthCom.2018.8531172.

17. Bharati Wukkadada ; Kirti Wankhede ; Ramith Nambiar ; Amala Nair. Comparison with HTTP and MQTT In Internet of Things (IoT) - IEEE Conference Publication. *2018 International Conference on Inventive Research in Computing Applications (ICIRCA 2018)* **2018**.

18. Tantitharanukul, N.; Osathanunkul, K.; Hantrakul, K.; Pramokchon, P.; Khoenkaw, P. MQTT-Topics Management System for sharing of Open Data. 2nd Joint International Conference on Digital Arts, Media and Technology 2017: Digital Economy for Sustainable Growth, ICDAMT 2017, 2017. doi:10.1109/ICDAMT.2017.7904935.

19. Alqinsi, P.; Matheus Edward, I.J.; Ismail, N.; Darmalaksana, W. IoT-Based UPS Monitoring System Using MQTT Protocols. *Proceeding of 2018 4th International Conference on Wireless and Telematics, ICWT 2018* **2018**, pp. 1–5. doi:10.1109/ICWT.2018.8527815.

20. O'Leary, Nick . Arduino CLient for MQTT, 2020. Online, available at https://pubsubclient.knolleary.net/.

21. Ecplise Foundation . Ecplise Paho MQTT CLient, 2020. Online, available at https://www.eclipse.org/paho/.

22. Tang, Q.; Ge, X.; Liu, Y.C. Performance analysis of two different SVM-based field-oriented control schemes for eight-switch three-phase inverter-fed induction motor drives. 2016 IEEE 8th International Power Electronics and Motion Control Conference, IPEMC-ECCE Asia 2016, 2016. doi:10.1109/IPEMC.2016.7512836.

23. Ross, M.; Graves, C.A.; Campbell, J.W.; Kim, J.H. Using support vector machines to classify student attentiveness for the development of personalized learning systems. Proceedings - 2013 12th International Conference on Machine Learning and Applications, ICMLA 2013, 2013. doi:10.1109/ICMLA.2013.66.

24. Saravanan, R.; Sujatha, P. A State of Art Techniques on Machine Learning Algorithms: A Perspective of Supervised Learning Approaches in Data Classification. Proceedings of the 2nd International Conference on Intelligent Computing and Control Systems, ICICCS 2018, 2019. doi:10.1109/ICCONS.2018.8663155.

25. Mamdouh, M.; Elrukhsi, M.A.; Khattab, A. Securing the Internet of Things and Wireless Sensor Networks via Machine Learning: A Survey. 2018 International Conference on Computer and Applications, ICCA 2018, 2018. doi:10.1109/COMAPP.2018.8460440.

26. Li, L.; Chou, W.; Zhou, W.; Luo, M. Design Patterns and Extensibility of REST API for Networking Applications. *IEEE Transactions on Network and Service Management* **2016**. doi:10.1109/TNSM.2016.2516946.
27. Hastie, T.; Tibshirani, R.; Friedman, J. *Springer Series in Statistics The Elements of Statistical Learning - Data Mining, Inference, and Prediction*; Springer, 2009.
28. Seif, G. A Beginner's guide to XGBoost, 2019. Online, available at https://towardsdatascience.com/a-beginners-guide-to-xgboost-87f5d4c30ed7.
29. scikit-learn . Machine Learning in Python, 2020. Online, available at https://scikit-learn.org/stable/.

APPENDIX B

User & Technical Manual

University Institute of Lisbon

Department of Information Science and Technology

User & Technical Manual

João Alves Coelho

November 2020

Contents

List of Figures **v**

1 Platform **1**

1.1 Mobile Application 1

1.1.1 Authentication 1

1.1.2 List of Sensors 3

1.1.3 Sensor Details 3

1.2 Server & Database 4

1.3 Scripts 5

1.4 Arduino IDE 6

List of Figures

1.1 Register Screen 2
1.2 Login Screen 2
1.3 Main Dashboard Screen 3
1.4 Sensor Details Screen 3
1.5 Database Relational Model 4
1.6 Python Script Files 5
1.7 HTML Template Files 6

Chapter 1

Platform

In order to provide the best user experience, an Android application was developed so that the user can view all the information collected and processed. In the next sections it is discussed how the mobile application was developed and how the information was handled on the server.

1.1 Mobile Application

The mobile application was developed in Android, with the main goal of providing the user a way to monitor in real time the water flow of the water distribution system. The information is sent periodically, allowing the user to stay updated on the status of the system and check if there is any abnormal situation.

1.1.1 Authentication

In order to be apart and use the system, the user must register in the application, otherwise the access to the sensor information is denied. The user must provide information on registration, username, e-mail and password. Figure 1.1 is showed the Registration screen. If the user uses already registered information it will give and error, the same happens if the passwords do not match. If there are no errors, the user is registered in the mobile application with success.

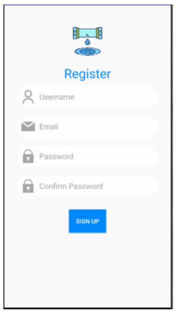
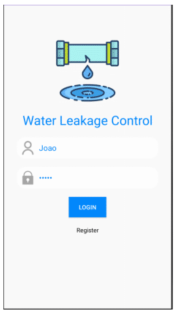


FIGURE 1.1: Register Screen

When the mobile application is launched, the Log In screen, Figure 1.2, is presented to the user. Each user has its username, email and password. In this screen the user needs to input its authentication to be able to access all of the mobile application functionalities.

If in the Log In screen the fields are filled with invalid credentials, an error message is presented to alert the user to verify what was inserted. In the case of being the first access of the user to the mobile application, the user needs to click on the "Register" button to access the Registration screen to perform the registration.



Joao

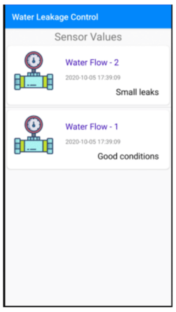
•••••

FIGURE 1.2: Login Screen

1.1.2 List of Sensors

The main screen is showed to the user after a successful authentication via the Log In screen,as represented in Figure 1.3. In this screen, is a list of all the sensors in the user system, with a card view for each sensors.

The card view contains the name of the sensor, the time the last sensor read was made and a message informing if there are leaks.



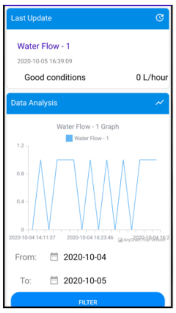
Water Flow - 2
2020-10-05 17:39:09
Small leaks

Water Flow - 1
2020-10-05 17:39:09
Good conditions

FIGURE 1.3: Main Dashboard Screen

1.1.3 Sensor Details

Sensor details screen is where all the information from the one sensor is displayed. On top, it shows a similar card view to the one in the main page, with a difference of now showing also the last value sent by the sensor. Below it shows an interactive graph that shows the last information analyzed by the machine learning algorithm.



Water Flow - 1
2020-10-05 16:39:09
Good conditions 0 L/hour

From: 2020-10-04
To: 2020-10-05

FIGURE 1.4: Sensor Details Screen

The graph also allows the user to select a specific interval of dates and it is also possible to see a timestamp and sensor value when clicking on a value on the graph.

1.2 Server & Database

The server is responsible for saving all the information collected by the sensors and for making all the necessary analyses so that the system can alert of possible water leaks. It is through the server that the mobile application gets the information to show the user.

Given the vast information collected from the sensors, and not only, it was required to structure the information and create a database. The relation diagram of the database is showed in Figure 1.5.

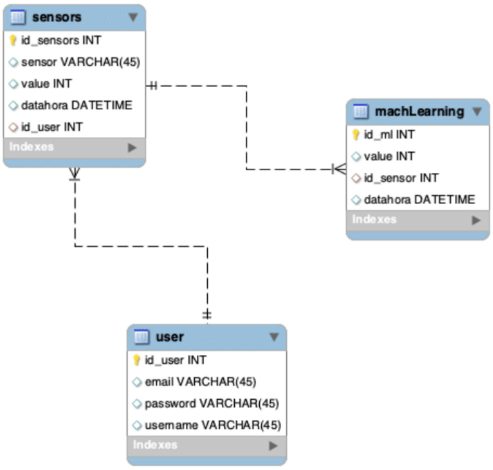


FIGURE 1.5: Database Relational Model

- **Sensors:** Table responsible for storing all information of the sensors, from name to value collected. The primary key is the id;

- **Mach Learning:** Stores all data related to the machine learning, with a timestamp and the calculated value. The primary key is the id;

- **User:** Responsible for storing the user information. It is from this table that the verification for authentication is made. The primary key is the id;

1.3 Scripts

The scripts where necessary to retrieve the information from the database, run in the background, and are responsible to make the information available in the mobile application.

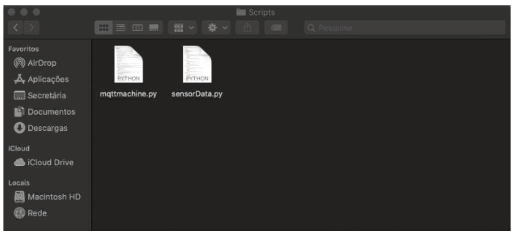


Figure 1.6: Python Script Files

- **sensorData.py:** Connection to the database to provide Flask with the specific information for the sensors;

- **mqttmachine.py:** Responsible for the connection to the database to store the data, using Paho Library. Also where the machine learning algorithm does the analysis of the data for the predictions and detection of water leaks;

The data gathered from the sensors, is then sent via Flask to a localhost for the mobile application to be able to retrieve it.

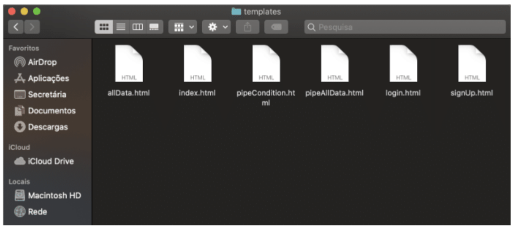


FIGURE 1.7: HTML Template Files

- **allData.html:** Where all the data from the sensors is presented via Flask;
- **index.html:** Where the last values from the sensors is presented via Flask;
- **pipeConditions.html:** Where the last values from the machine learning is presented via Flask;
- **pipeAllData.html:** Where all the data from the machine learning is presented via Flask;
- **login.html:** Where the data related to the login of the user is presented via Flask;
- **signUp.html:** Where the data related to the registration of the user is presented via Flask;

1.4 Arduino IDE

All the microcontrollers where configured using the Arduino IDE, which can be downloaded to any computer, from the Arduino website: https://www.arduino.cc. The code was developed in C/C++ and the libraries used where:

- Paho;

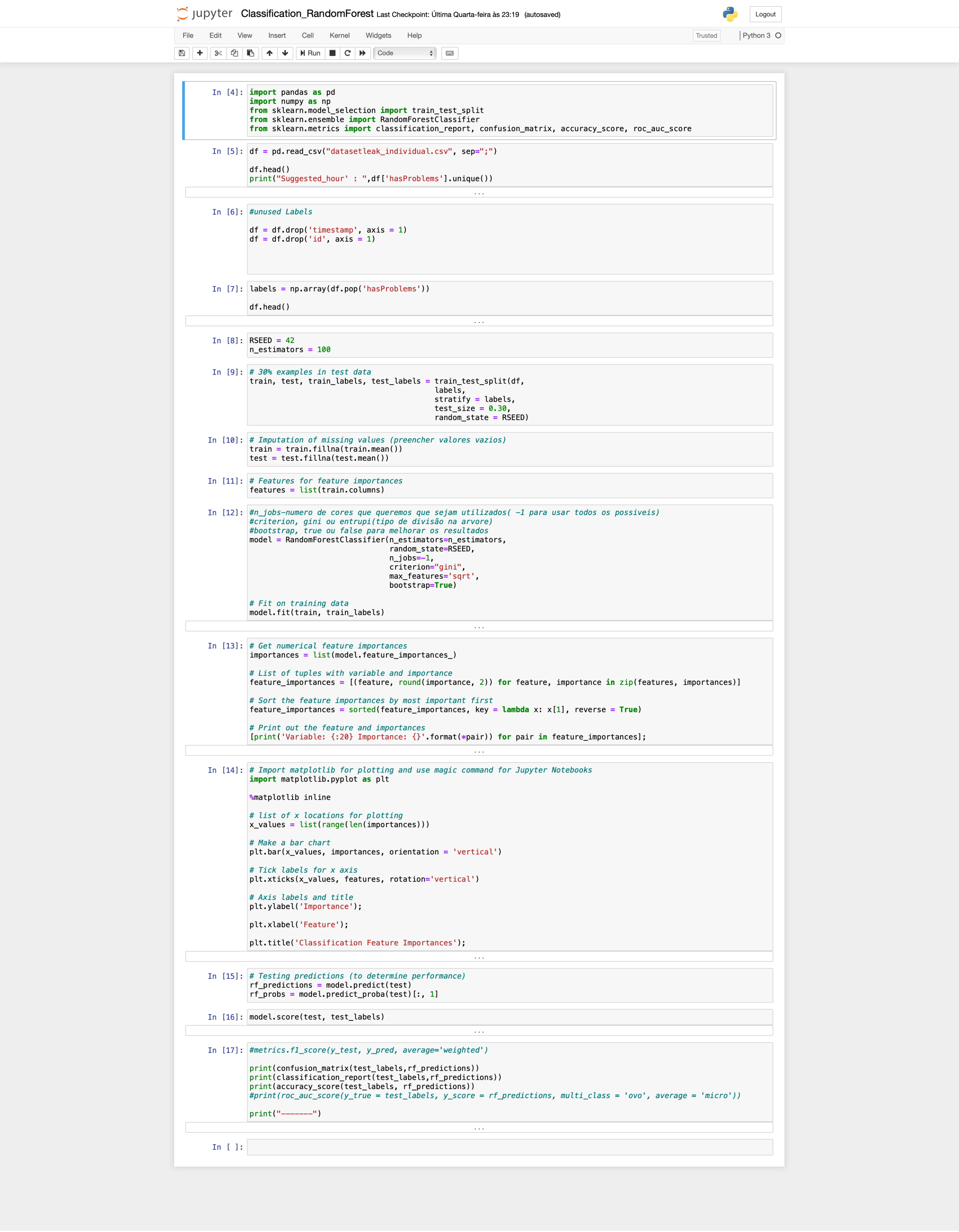
- PubSubClient;

The file created with the IDE to be deployed to the sensors:

- **flow.ino:** File with the code responsible for having the sensors collecting the data related to the water flow;

APPENDIX C

Detailed Image



jupyter
Classification_RandomForest Last Checkpoint: Última Quarta-feira às 23:19 (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3
Run
Code

```
In [4]: import pandas as pd
        import numpy as np
        from sklearn.model_selection import train_test_split
        from sklearn.ensemble import RandomForestClassifier
        from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, roc_auc_score

In [5]: df = pd.read_csv("datasetleak_individual.csv", sep=";")

        df.head()
        print("Suggested_hour' : ",df['hasProblems'].unique())
...

In [6]: #unused Labels

        df = df.drop('timestamp', axis = 1)
        df = df.drop('id', axis = 1)

In [7]: labels = np.array(df.pop('hasProblems'))

        df.head()
...

In [8]: RSEED = 42
        n_estimators = 100

In [9]: # 30% examples in test data
        train, test, train_labels, test_labels = train_test_split(df,
                                                                  labels,
                                                                  stratify = labels,
                                                                  test_size = 0.30,
                                                                  random_state = RSEED)

In [10]: # Imputation of missing values (preencher valores vazios)
         train = train.fillna(train.mean())
         test = test.fillna(test.mean())

In [11]: # Features for feature importances
         features = list(train.columns)

In [12]: #n_jobs-numero de cores que queremos que sejam utilizados( -1 para usar todos os possiveis)
         #criterion, gini ou entrupi(tipo de divisão na arvore)
         #bootstrap, true ou false para melhorar os resultados
         model = RandomForestClassifier(n_estimators=n_estimators,
                                        random_state=RSEED,
                                        n_jobs=-1,
                                        criterion="gini",
                                        max_features='sqrt',
                                        bootstrap=True)

         # Fit on training data
         model.fit(train, train_labels)
...

In [13]: # Get numerical feature importances
         importances = list(model.feature_importances_)

         # List of tuples with variable and importance
         feature_importances = [(feature, round(importance, 2)) for feature, importance in zip(features, importances)]

         # Sort the feature importances by most important first
         feature_importances = sorted(feature_importances, key = lambda x: x[1], reverse = True)

         # Print out the feature and importances
         [print('Variable: {:20} Importance: {}'.format(*pair)) for pair in feature_importances];
...

In [14]: # Import matplotlib for plotting and use magic command for Jupyter Notebooks
         import matplotlib.pyplot as plt

         %matplotlib inline

         # list of x locations for plotting
         x_values = list(range(len(importances)))

         # Make a bar chart
         plt.bar(x_values, importances, orientation = 'vertical')

         # Tick labels for x axis
         plt.xticks(x_values, features, rotation='vertical')

         # Axis labels and title
         plt.ylabel('Importance');

         plt.xlabel('Feature');

         plt.title('Classification Feature Importances');
...

In [15]: # Testing predictions (to determine performance)
         rf_predictions = model.predict(test)
         rf_probs = model.predict_proba(test)[:, 1]

In [16]: model.score(test, test_labels)
...

In [17]: #metrics.f1_score(y_test, y_pred, average='weighted')

         print(confusion_matrix(test_labels,rf_predictions))
         print(classification_report(test_labels,rf_predictions))
         print(accuracy_score(test_labels, rf_predictions))
         #print(roc_auc_score(y_true = test_labels, y_score = rf_predictions, multi_class = 'ovo', average = 'micro'))

         print("-------")
...

In [ ]:
```