



UNIVERSIDAD DE EXTREMADURA

Escuela Politécnica

Máster en Ingeniería en Telecomunicación

Trabajo Fin de Máster

Development of an autonomous system for acoustic
capture and identification of animal species



UNIVERSIDAD DE EXTREMADURA

Escuela Politécnica

Máster en Ingeniería en Telecomunicación

Trabajo Fin de Máster

Development of an autonomous system for acoustic
capture and identification of animal species

Autor: Diego Martínez Megías

Tutor: Antonio Gordillo Guerrero

GENERAL TABLE OF CONTENTS

TABLE INDEX.....	6
FIGURES INDEX.....	7
FORMULA INDEX.....	9
SUMMARY.....	10
1. INTRODUCTION.....	11
1.1 <i>Single Board Computers (SBC)</i>	11
1.2 Communication protocols in IoT devices.....	12
1.3 Sound and components	12
1.4 Scope of the project	13
2. OBJECTIVES	14
3. STATE OF THE ART	15
3.1 Single Board Computers (SBC) and Arduino	15
3.1.1 Energy consumption	15
3.1.1.1 <i>Raspberry Pi</i>	15
3.1.1.2 <i>BeagleBone</i>	16
3.1.1.3 <i>ODROID</i>	17
3.1.1.4 <i>Arduino</i>	18
3.2 Configuration and programming capability.....	20
3.2.1 Programming	20
3.2.2 Operating systems	20
3.2.3 Configuration	21
3.3 Sound capture	21
3.3.1 Dynamic moving coil microphone.....	21
3.3.2 Dynamic ribbon microphone	23
3.3.3 Condenser microphone	25

3.4 Species recognition	27
3.5 3.5 Wireless connectivity protocols	28
3.5.1 Wi-Fi	29
3.5.2 LoRa	30
3.5.3 Zigbee	31
3.5.4 NB-IoT	32
3.6 3.7 Connectivity protocols between devices	33
3.6.1 MQTT	33
3.6.2 HTTP	34
3.6.3 CoAP	35
4. MATERIAL AND METHOD	38
4.1 ITERATION 1	38
4.1.1 Objectives	38
4.1.2 Results	38
4.2 ITERATION 2	39
4.2.1 Objectives	40
4.2.2 Results	40
4.3 ITERATION 3	40
4.3.1 Objectives	41
4.3.2 Results	41
4.4 ITERATION 4	41
4.4.1 Objectives	41
4.4.2 Results	41
5. IMPLEMENTATION AND DEVELOPMENT	42
5.1 ITERATION 1	42
5.1.1 Raspberry Pi Zero W	42

5.1.2 Microphone selection	43
5.1.1.1 Polar diagram	43
5.1.1.2 Frequency response	45
5.1.2 Energy management system	46
5.1.3 Data collection	49
5.1.4 Wi-Fi module control	53
5.2 ITERATION 2	54
5.4.2 Data processing	54
5.4.2.1 Parameter acquisition	54
5.4.2.1.1 Zero Crossing Rate (ZCR)	55
5.4.2.1.2 Spectral Entropy	55
5.4.2.1.3 Spectral Centroid	56
5.4.2.1.4 Roll-Off Factor	57
5.4.3 Data management	57
5.3 ITERATION 3	60
5.4 ITERATION 4	61
6. RESULTS AND DISCUSSION	64
7. CONCLUSIONS	66
8. FUTURE WORK	67

TABLE INDEX

Table 1: Comparative table of power consumption for different Raspberry Pi models	16
Table 2: Comparative table of power consumption for different BeagleBone models.	17
Table 3: Comparative power consumption table for different ODROID models	17
Table 4. Power consumption comparison table for different Arduino models.....	18
Table 5: Comparison of different microphones based on their polar diagram.....	43
Table 6: Witty Pi 3 Mini device power consumption in standby mode.....	49

FIGURES INDEX

Figure 1: Example of a BeagleBone Black SBC model.....	11
Figure 2: Comparative graph of current consumption (in mA) for different models of SBCs and Arduino.....	19
Figure 3: Graphical representation of dynamic moving coil microphone operation.....	22
Figure 4: Frequency response of Shure Beta 58A microphone. The dashed lines correspond to different distances from the focus	23
Figure 5: Graphical representation of dynamic ribbon microphone operation.....	24
Figure 6: Frequency response for Superlux R102 and Superlux R102MKII microphones.....	24
Figure 7: Graphical representation of condenser microphone operation.....	25
Figure 8: Frequency response of Sennheiser E 965 microphone.....	26
Figure 9: Coverage map for GSM, UMTS, LTE and 5G technologies in Spain as of 03/05/2023.....	28
Figure 10: Temporal and spectral representation of Chirp signals used in Chirp Spread Spectrum (CSS) modulation in LoRa technology.....	30
Figure 11: Example of network topology used in Zigbee.....	32
Figure 12: Basic operation diagram of the MQTT protocol.....	33
Figure 13: Comparación del ancho de banda requerido para <i>MQTT</i> y <i>HTTP</i>	35
Figure 14: Simple network showing the operation of CoAP.....	36
Figure 15: Number of packets per message for different CoAP and MQTT connections.....	36
Figure 16: Graphical example of omnidirectional microphone operation	44
Figure 17: The phase cancellation effect.....	44
Figure 18: Primo EM272 condenser microphone.....	45
Figure 19: Primo EM272 microphone with windshield installed.....	46
Figure 20: Raspberry Pi Zero W with Witty Pi 3 Mini installed.....	47

Figure 21: Witty Pi 3 Mini device configuration panel.....	48
Figure 22: .wpi file defined for use in Witty Pi 3 Mini running a 5 minute on every 20 minute off schedule	48
Figure 23: Recorder class defined for recording audio samples on the <i>SOLMoth</i> device.....	50
Figure 24: Graphical visualization of the bit depth concept in audio sample coding.....	51
Figure 25: Contents of the <i>recordingaudio.service</i> file by which the system service is defined.....	52
Figure 26: Example of directory contents where the audio samples recorded by the <i>SOLMoth</i> device during its execution are stored.....	53
Figure 27: Wi-Fi module power-on function on Raspberry Pi Zero W via system command.....	54
Figure 28: Function used in the <i>SOLMoth</i> project to extract parameters from an input audio file.....	57
Figure 29: Function used in the <i>SOLMoth</i> project to check the size of a given directory.....	58
Figure 30: Function used in the <i>SOLMoth</i> project to iterate through all audio files in a given directory, obtaining the parameters returned by the <i>data_extraction</i> function, storing them in a .csv file, and finally deleting the file.....	59
Figure 31: Status of the <i>processingaudio.service</i> once it is defined by the system.	59
Figure 32: Example of system output in sending data to an MQTT broker.....	60
Figure 33: Header of the <i>mqtt_request</i> function used for the management and sending of data to a given MQTT broker.....	60
Figure 34: Example of message received in MQTT broker where the processed data from three audio samples have been published.....	61
Figure 35: Image of the complete first version of the <i>SOLMoth</i> device without an external casing.....	62

FORMULA INDEX

Formula 1. Calculation of the autonomy of the device with a 20000mAh battery....	19
Formula 2. Charge of a capacitor.....	25
Formula 3. Zero crossing rate.....	55
Formula 4. Spectral entrophy.....	55
Formula 5. Probabilistic distribution for the spectral power of the signal with respect to the number of points on frequency	56
Formula 6. Definition of signal spectral power	56
Formula 7. Spectral centroid.....	56
Formula 8. <i>Roll-Off</i> Factor.....	57

SUMMARY

The evolution of technology has directly affected telecommunications, with the emergence of the ability to connect multiple everyday devices to the Internet, allowing their remote control or the sending of data from them; this is what we call **the Internet of Things (IoT)**.

Likewise, the rapid growth of the global population and industry has led to a considerable increase in pollution and the temperature of our planet, directly affecting bird migration and the periodicity with which birds migrate [1].

The convergence of these phenomena leads us to propose a system that allows us to monitor birds and other animals in nature, being able to identify the population of these at each time of the year. Thus, the *SOLMoth* device, developed in this project, is responsible for autonomously capturing the various bird sounds in the area where it is installed for future classification, sending the data wirelessly.

SOLMoth is an open device that offers a great number of possibilities, starting with the integration of new devices up to the realization of a classification based on Artificial Intelligences. In this sense and in collaboration with the Fab lab Smart Open Lab, a community located in the *Escuela Politécnica de Cáceres* (EPCC), we developed an initial prototype based on Raspberry Pi Zero W where we execute the aforementioned functionalities.

The specifications of autonomy, audio sample recording, data processing, data sending, OS configuration, as well as the integration and configuration of multiple components in the same environment have been achieved, concluding in a robust device whose functionality is in accordance with the specifications.

1. INTRODUCTION

This project aims to converge different technologies and methodologies in a single device. To do this, we must evaluate the state of the art in the various areas involved here and make a selection, seeking as main objectives to be open source and economical.

In this sense, the different technologies to be integrated are presented below:

1.1 *Single Board Computers (SBC)*

The Internet of Things has brought the sensorization of multiple environments in our daily lives. In general, various physical phenomena are sensed to collect information and communicate with other devices, commonly Single Board Computers (SBC). An example of an SBC is shown in Figure 1:

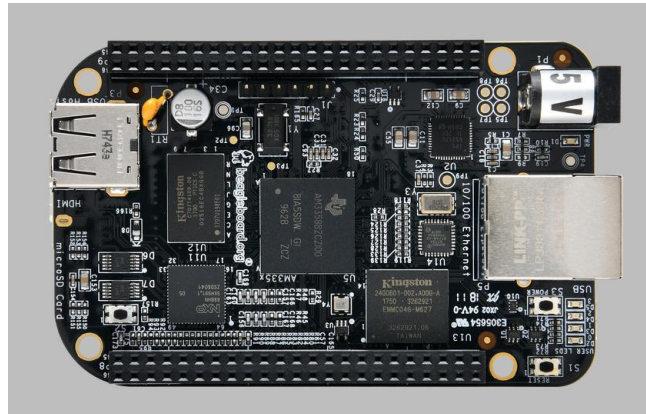


Figure 1: Example of a BeagleBone Black SBC model.

Nowadays it is common to work with Arduino [2] or Raspberry Pi [3], due to their popularity to carry out projects and formalize communication with various sensors, although there are multiple other options on the market (BeagleBone, ODROID...).

In general, they are chosen for their price, community and supporting documentation.

1.2 Communication protocols in IoT devices

- The communication between an IoT device and the server or gateway with which it communicates is essential because part of its lightness, efficiency or scalability depends solely on this protocol. In this sense, today we find multiple protocols with this approach:
- MQTT: Lightweight and efficient message communication protocol, highly scalable and popular today.
- HTTP: Standard communication protocol designed for communication between browsers and web servers.
- CoAP: Protocol designed for use on simple, low-resource devices with similarity to the HTTP protocol.

In general, the choice of a communication protocol for the development of an IoT application will seek to prioritize the factors previously mentioned, in order to maintain a robust communication and whose messages are not very heavy.

1.3 Sound and components

Sound capture is a process that can be performed in multiple ways. Thus, we can use digital audio recorders, devices with integrated microphones or USB microphones. In general, we need both a microphone that captures the signal, generating an analog signal, and a sound card that is responsible for converting this signal from analog to digital.

Within the world that encompasses sound cards and microphones, we have plenty of possibilities and methods that are based on taking advantage of various physical phenomena.

An important concept in this section is the polar diagram of the microphone, i.e. the sensitivity of the microphone in relation to the angle of incidence of the sound. In general, when recording ambient sound, the sound can be received from anywhere, especially if we are talking about birds, which is why it is important to choose a

microphone that allows us to receive sound from as many angles as possible, i.e. omnidirectional.

In general, the choice of components and protocols can be very specific depending on the focus of the project, so later we will talk about the elements chosen for this project taking into account the specifications sought.

1.4 Scope of the project

Based on these technologies and in complement with the knowledge in sound acquired during the degree and master, *SOLMoth* has been proposed as a development that unifies electronics and acoustic signal processing, together with wireless networks and IoT communication protocols to formalize a robust and autonomous device capable of capturing sounds, processing them and sending the data to a server.

The concept idea of *SOLMoth* is to develop a utility application that, complimented with a future development using a Machine Learning algorithm [4], is considered capable and effective in performing an accurate classification of birds and animals found in nature. In this sense, *SOLMoth* is expected to be a support device in the control of bird migration or in the detection of illegal hunting of animals, among many other possible uses.

2. OBJECTIVES

The objective of this project is the development and realization of an initial functional prototype for the SOLMoth device, carrying out a study of the components to be chosen and selecting the technology to be used, together with an investigation of the treatment and processing of the acquired signals.

The following objectives are to be achieved:

- Integrate electronics from multiple devices running under the same Operating System.
- To provide autonomy to the device, allowing it to be installed in rural areas where it is not possible to supply it with power from the electrical grid.
- Capture acoustic signals, store them and process them based on future objectives for the realization of Machine Learning algorithms.
- To develop data transmission using the MQTT protocol, initially based on Wi-Fi wireless communication.
- Study wireless technologies in rural environments with low coverage.

3. STATE OF ART

3.1 Single Board Computers (SBC) and Arduino

Single Board Computers (SBC) are a category of electronic devices that are based on the implementation of all their components on a single Printed Circuit Board (PCB). In this sense, the use of these devices has become extremely popular in the last decades due to their advantages of size, price, power consumption and computational capacity.

In the Internet of Things arena, SBCs have proven to be one of the most economical and versatile options due to their configurability and connectivity. The most important features to consider when selecting an SBC are presented:

3.1.1 Energy consumption

One of the great advantages of SBCs is their power consumption. In this sense, SBCs can operate for up to several months with a simple portable battery or batteries, leaving behind more bulky and expensive power supply methodologies such as Uninterruptible Power Supply (UPS).

The vast majority of these devices use low power consumption processors, operating at low voltages around 5V. However, the overall power consumption will depend on other factors, such as the number of components connected to the I/O interface, the use of wireless communications or data storage.

Some of the most popular SBCs are described below:

3.1.1.1 Raspberry Pi

Raspberry Pi [3] is one of the widest and most widely used ranges, presenting a large number of models with different computational capacity and resources. In this sense, Table 1 shows a summary of the approximate consumptions [5] for each of these:

Pi Model	Pi State	Power Consumption
3 B+	HDMI off, LEDs off	350 mA (1.7 W)
3 B+	HDMI off, LEDs off, onboard WiFi	400 mA (2.0 W)
3 B	HDMI off, LEDs off	230 mA (1.2 W)
3 B	HDMI off, LEDs off, onboard WiFi	250 mA (1.2 W)
2 B	HDMI off, LEDs off	200 mA (1.0 W)
2 B	HDMI off, LEDs off, USB WiFi	240 mA (1.2 W)
Zero 2 W	HDMI off, LED off	100 mA (0.6 W)
Zero 2 W	HDMI off, LEDs off, onboard WiFi	120 mA (0.7 W)
Zero	HDMI off, LED off	80 mA (0.4 W)
Zero	HDMI off, LED off, USB WiFi	120 mA (0.7 W)
B+	HDMI off, LEDs off	180 mA (0.9 W)
B+	HDMI off, LEDs off, USB WiFi	220 mA (1.1 W)
A+	HDMI off, LEDs off	80 mA (0.4 W)
A+	HDMI off, LEDs off, USB WiFi	160 mA (0.8 W)

Table 1: Comparative table of power consumption for different Raspberry Pi models [3].

3.1.1.2 BeagleBone

Beaglebone [6] is a development board similar to the solution presented by Raspberry Pi. In this sense, Beaglebone is focused on the use of Open Source software and resources [7], i.e., developments whose source code and rights are in the public domain. In addition, most of its models have more than the 40 generic pins that Raspberry Pi has.

As in the previous section, Table 2 shows the approximate power consumption for each of the models in a normal operating regime:

Model	Average Current Draw (mA)
BeagleBone Black	210 - 460
BeagleBone Green	210 - 460
BeagleBone Green Wireless	210 - 460
BeagleBone Blue	210 - 460
BeagleBone AI	500 – 1000

Table 2: Comparative table of power consumption for different BeagleBone models.

3.1.1.3 ODROID

ODROID is a series of development boards developed by the Korean company Hardkernel [8]. Thus, they present a set of highly configurable and powerful SBCs in various models that, as in the previously presented devices, offer a wide variety of available communication ports, storage card slots and General Purpose Input/Output (GPIO), i.e., generic pins whose behavior can be defined by the user in order to develop a specific input or output action.

Table 3 presents a summary of the most popular models of the brand with their approximate power consumption in normal operation:

Modelo	Average Current Draw (mA)
ODROID-C2	350 - 400
ODROID-XU4	600 - 700
ODROID-N2	500 - 600
ODROID-H2+	1000 - 1500

Table 3: Comparative power consumption table for different ODROID models.

3.1.1.4 Arduino

Arduino is an Open Source hardware and software development company that consists of several microcontroller development boards [9], i.e., integrated circuits with logic-arithmetic units that are slower than a microprocessor because they are intended to fulfill specific tasks. Technically, Arduino is not considered an SBC, since it falls short of being a full computer, however it is widely used in IoT projects and sensor integration, both for wired and wireless modality.

The Arduino development environment includes numerous programming libraries and device integration, which has made it, for power, simplicity and economic cost, one of the most widely used devices in sensor integration and electronics projects.

As in previous sections, Table 4 shows the different Arduino models together with their approximate power consumption in normal operation.

Modelo	Average Current Draw (mA)
Arduino UNO	30 - 40
Arduino Nano	20 - 25
Arduino Mega	40 - 50
Arduino Leonardo	20 - 25
Arduino Due	70 - 80
Arduino Pro Mini	10 - 20
Arduino LilyPad	20 - 25
Arduino Esplora	20 - 25
Arduino Robot	60 - 70
Arduino Tian	70 - 80
Arduino Yun	170 - 200
Arduino Industrial 101	25 - 30
Arduino MKRFOX1200	20 - 25
Arduino MKR1000	30 - 40
Arduino MKRZero	20 - 25
Arduino MKR Vidor 4000	30 - 40

Table 4: Power consumption comparison table for different Arduino models.

As a summary and for comparative purposes, we can see in Figure 2 a comparison of the power consumption of different SBCs with respect to time [10] when they are connected to the Internet via Ethernet cable (Beaglebone Black, Raspberry Pi Model B and Intel Galileo), via WiFi (Arduino Yun) or connected via a keyboard and monitor (Raspberry Pi Model A):



Figure 2: Comparative graph of current consumption (in mA) for different models of SBCs and Arduino.

As we can see, the power consumption can be very variable depending on the model and application we are using. As a general rule and as an example, if we were working with a 20000mAh portable battery, the duration of the board, in hours, powered by this battery would follow the following relationship:

$$\text{Duration (h)} = \frac{20000\text{mAh}}{\text{SBC Consumption (mA)}} \quad [1]$$

Thus, comparing with Figure 1, the autonomy of the Raspberry Pi Model A would be approximately 150h. Making the same calculation for Intel Galileo, we would have about 30-40h, more than three times shorter than the best case.

This is why, in energy-autonomous systems, it is crucial to manage consumption and correctly choose the board that best meets the characteristics of the project. This will essentially depend on the components and peripherals used by the device and the amount of computation performed.

3.1.2 Configuration and programming capability

One of the most important processes when choosing a development board will be to know the specifications of our project. In this sense, we can distinguish the SBC and Arduino in several points:

3.1.2.1 Programming

In the case of software programming, Arduino uses its own programming language, based on C/C++, while the SBCs, by using an Operating System, can opt to use the various existing programming languages. In this sense, it could be said that Arduino is focused on electronics and robotics, while SBCs are general purpose, being able to adapt to the needs of more generic projects.

3.1.2.2 Operating systems

SBCs have the advantage of running Operating Systems, such as Linux, Android or Windows. In a common way, Linux will be the Operating System used in these devices, since it is an Open Source Operating System where we can access to a wide range of tools and applications.

On the other hand, Arduino does not use an Operating System, but makes use of its programming environment to interact directly with the connected devices and define the program flow. This concept is commonly referred to as firmware [11], i.e., an

unalterable computer program that defines the operating logic of the electronic circuit.

The use of Operating Systems allows us to launch various system services, being able to access various projects or programs that we can use in parallel. This means running a greater number of simultaneous tasks constantly, which means that consumption is always higher in this type of system.

3.1.2.3 Configuration

In terms of device configuration, SBCs are more complex to configure than Arduino because it requires an understanding of how the operating system works. Likewise, they offer more configuration and customization, allowing complex structures for projects that require it.

Arduino, on the contrary, is simple and straightforward, so it will be suitable for less complex projects and where multiple functionalities are not developed. If this is the case, a program flow must be correctly defined, based, for example, on a state machine [12].

3.3 Sound capture

Sound pickup is the process by which sound waves are captured and converted to electrical waves for further processing. Thus, these are generated by a microphone, a device that, based on some physical principle, is responsible for the conversion of sound signal to electrical signal.

In this sense, the most commonly used types of microphones are:

3.2.1 Dynamic moving coil microphone

A dynamic microphone is an electroacoustic transducer [13] that uses a diaphragm and a voice coil or ribbon to convert sound waves into electrical signals. Thus, the diaphragm is a moving membrane that vibrates in response to incident sound waves together with a moving coil within a magnetic field generated by a permanent magnet. In other words, the principle of induction is used to generate the electric current.

Based on this principle, the diaphragm will cause the voice coil to move within a magnetic field, generating a voltage representative of the incident sound. This principle can be visualized in Figure 3:

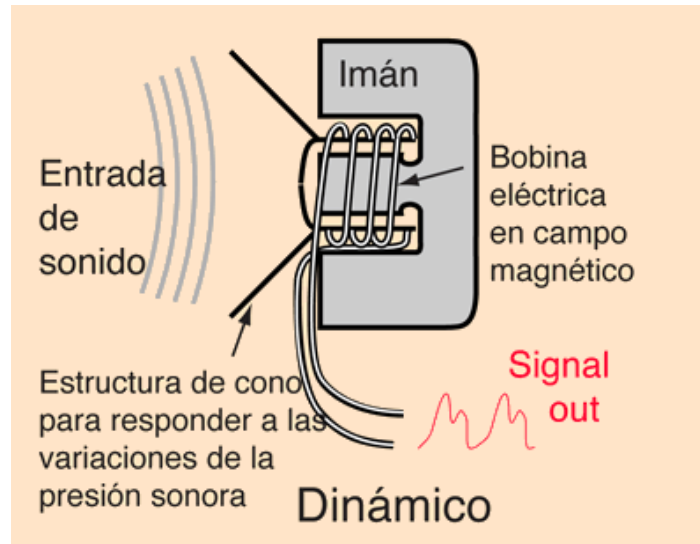


Figure 3: Graphical representation of dynamic moving coil microphone operation.

A great advantage of these types of microphones is that they are rugged and provide good performance in outdoor or noisy environments, such as concerts, however, the frequency behavior is not linear. An example of the frequency response of a dynamic moving coil microphone is presented in Figure 4 for the Shure Beta 58A microphone [14], designed for stage vocals:

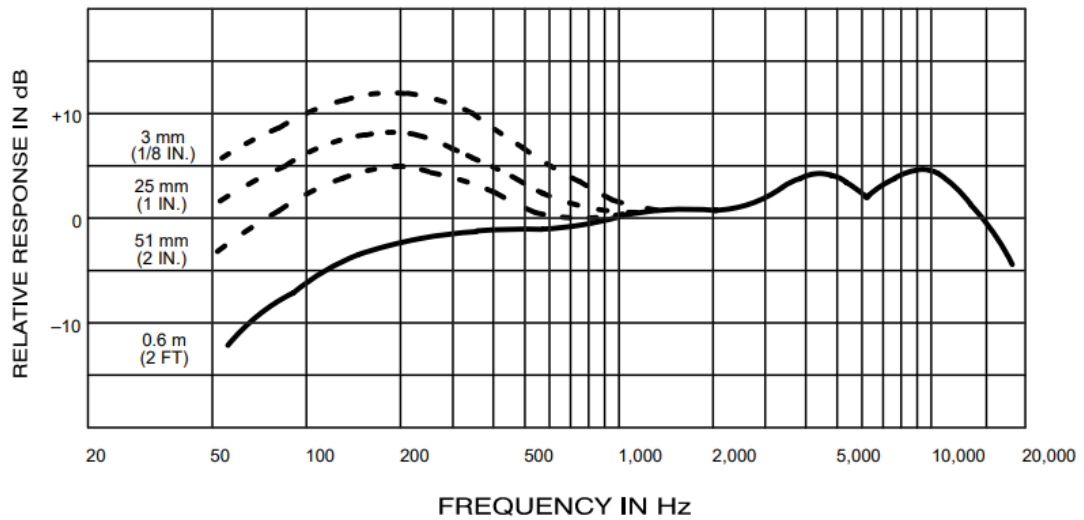


Figure 4: Frequency response of Shure Beta 58A microphone. The dashed lines correspond to different distances from the focus.

It is possible to observe, based on Figure 4, how for distances closer to the microphone the response in the lower frequencies increases, placing this response in the frequency zone where the human voice works in a fundamental way. In addition, we can verify that this frequency response provided by the microphone is not linear.

3.2.2 Dynamic ribbon microphone

Ribbon microphones are also within the group of dynamic microphones, being its principle of operation such that a metal tape is fixed between the poles of a magnet [15], so that the vibration that can perform is of the order of micrometers. Thus, the incident sound pressure wave moves the metallic tape within the magnetic field generated by the magnet, generating a voltage image of the incident acoustic signal.

This principle can be seen in Figure 6:

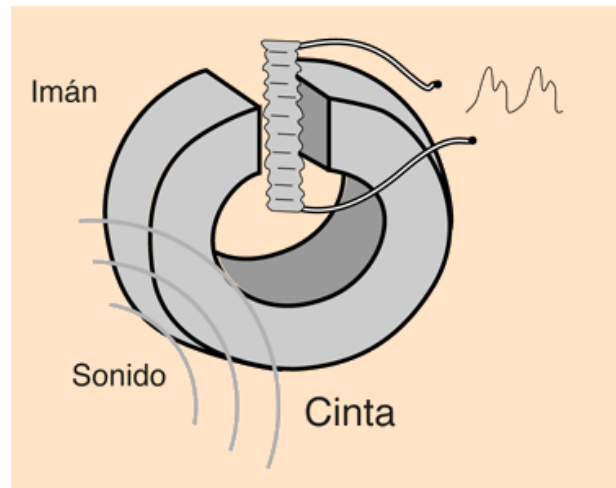


Figure 5: Graphical representation of dynamic ribbon microphone operation.

One of the advantages of this type of microphone is that it accentuates the low frequencies, since we find the resonant frequency of the ribbon at these frequencies. Regarding the frequency response, this is commonly linear, as can be seen in Figure 6 for the Superlux R102 and Superlux R102MKII microphone [16].

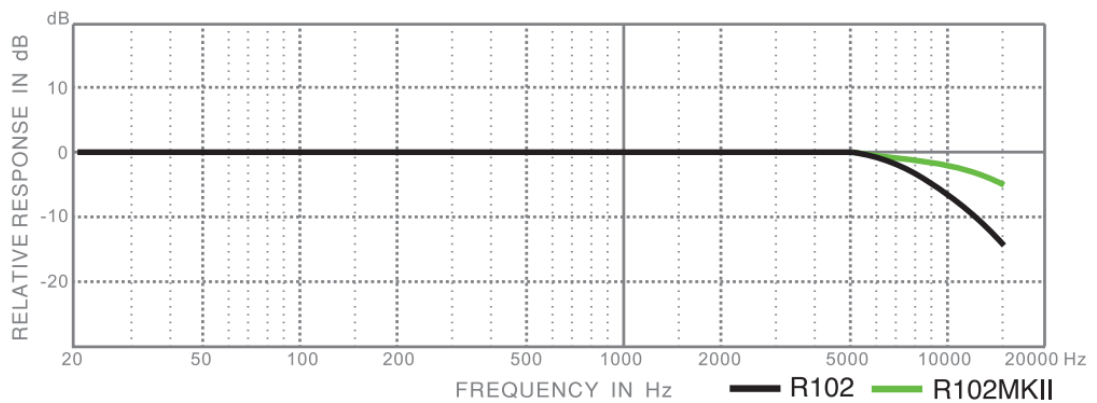


Figure 6: Frequency response for Superlux R102 and Superlux R102MKII microphones.

In general, these microphones can be at the level of condenser microphones, which we will discuss below, however, they are very sensitive to wind, vibrations and rapid movements, which is why their use is usually limited to studios.

3.2.3 Condenser microphone

A condenser microphone uses a condenser plate and a diaphragm to convert sound signals into electrical signals. Thus, the condenser plate is an electrically charged sheet of metal that is placed near the diaphragm, varying the distance between the two when a signal strikes the diaphragm, causing it to vibrate. This distance variation causes a variation in the capacitance of the capacitor, generating an electrical signal, as shown in Figure 7 [17]:

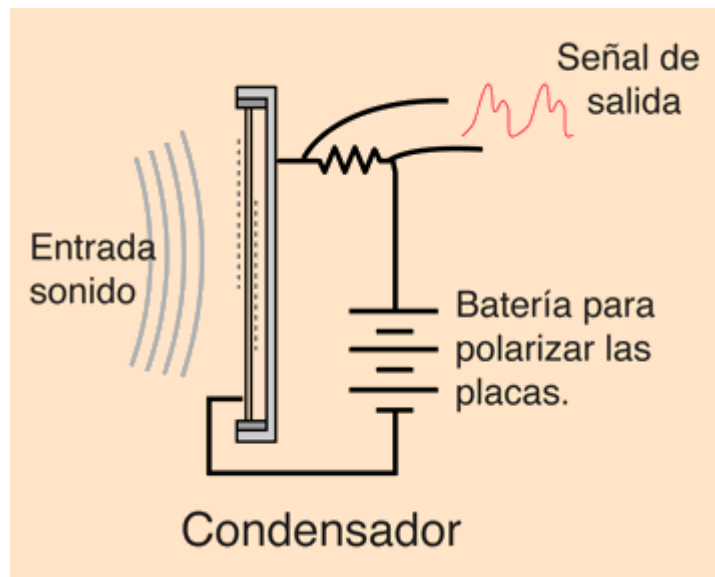


Figure 7: Graphical representation of condenser microphone operation.

This physical principle is based on the fact that the charge of a capacitor is given by the formula:

$$Q = CV = \frac{\epsilon \cdot A \cdot V}{d} \quad [2]$$

Where Q is the capacitor charge, C is the capacitance of the capacitor, V is the battery voltage, A is the area of each plate and d is the separation between these plates.

Based on Figure 7, we can visualize that a change in the distance between the plates of the capacitor, due to an incident signal, will cause a change in the charge Q of the capacitor, generating an electric current through the resistor. According to Formula 2, as in dynamic microphones, this electrical signal will be an image of the incident sound pressure wave, turning this microphone into an electroacoustic transducer.

Unlike dynamic moving coil microphones, this type of microphone is much more sensitive, allowing us to pick up signals with lower sound pressure level or at a far distance. In the same way, this type of microphone usually has a wider frequency response than the previous ones. This phenomenon can be seen in Figure 8, where the frequency response of the Sennheiser E 965 microphone is presented [18]:

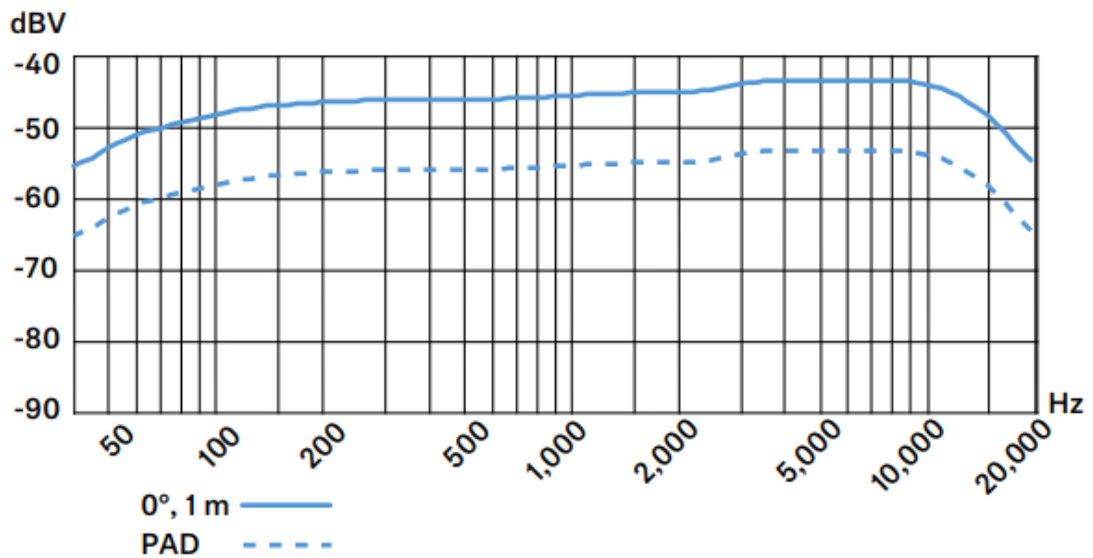


Figure 8: Frequency response of Sennheiser E 965 microphone [17].

However, the price of these microphones is usually higher than the previous ones, adding the fact that they need a battery or an external power supply to operate.

3.4 Species recognition

The recognition of animal species has undergone great advances in recent years, especially with the emergence and development of new tools in the field of Artificial Intelligence, a discipline that combines various algorithms with the aim of mimicking

human intelligence. In this sense, one of the biggest challenges in this field is the fact that there is a great variability and complexity in the sounds emitted by animals, especially in terms of frequency, duration and amplitude.

In order to be able to perform species recognition, it is essential to have a database large enough to encompass multiple sounds for the same animal, since even within the same species, the song or sound emitted can vary significantly.

Regarding classification algorithms, Convolutional Neural Networks (CNNs) [19], Recurrent Neural Networks (RNNs) [20], Long Short-Term Memory (LSTM) [21] and Support Vector Machines (SVM) [22] are the most widely used, although the variability of the approach in the project itself causes greater success in some algorithms or others, especially in those of the supervised type [23].

In general, any classifier algorithm receives as input parameter different characteristic values of the subject to be predicted. In the case of audio signals, it is common to work with mathematical parameters calculated directly in the signal or even psychoacoustic values such as sharpness or roughness. Thus, results obtained from predictions using mathematical values have shown high performance in speech and music classification [24].

Within these parameters, also called low-level parameters, we find data such as:

- Root Mean Square Value (*RMS*)
- Spectral Centroid
- Bandwidth (*BW*)
- Zero Crossing Rate (*ZCR*)
- *Roll-Off* factor
- Energy ratios of the signal
- Pitch

As a general rule, it is not common to work only with a single parameter, but for the same signal, multiple low-level data of the signal are calculated, obtaining a vector of representative values.

Finally, the selected Artificial Intelligence algorithm will be trained with a percentage of the species database obtained and will make subsequent predictions based on the generated inputs given, ending with the calculation of its effectiveness.

3.6 Wireless connectivity protocols

Wireless networks are networks that use radio waves to connect devices avoiding the use of wiring [25]. These networks allow devices to perform data exchanges even in remote locations, which has led to a boom in connectivity and provided coverage in a large percentage of the area of Spain and the entire world. As can be seen in Figure 9 [26] for GSM, UMTS, LTE and more recently, 5G technologies.



Figure 9: Coverage map for GSM, UMTS, LTE and 5G technologies in Spain as of 03/05/2023

Regarding the connection for IoT devices, in locations inside the home it is common to make use of Wi-Fi technology, while for remote locations in recent years numerous technologies and protocols have appeared that allow this connection, as is the case of LoRa, Zigbee or NB-IoT, which are described below:

3.6.1 Wi-Fi

Wi-Fi technology is a set of wireless communication standards that enables data exchange between devices. It is based on the use of high-frequency radio waves for data transmission according to the 802.11 standard, a family of wireless standards created by the Institute of Electrical and Electronic Engineers (IEEE).

When we talk about Wi-Fi technology, it is common nowadays to talk about dual band, i.e. devices operating in the 2.4 GHz or 5 GHz band, each of which has a number of advantages and disadvantages, especially in the area of data transfer speed, range or possible interference with household devices.

The most recent Wi-Fi standard is 802.11ax, also known as Wi-Fi 6, which is based on Multiple Input Multiple Output (MU-MIMO) technology, improving energy efficiency and higher data transfer.

When it comes to the use of chips such as the ESP8266, Arduino UNO or even Raspberry Pi, it is common to use the 802.11 b/g/n WiFi standard set, where:

- 802.11b: It is the oldest of the three, operating in the 2.4 GHz band and allowing data transfer of up to 11Mbps.
- 802.11n: This standard also operates in the 2.4GHz band, however, it can also operate at 5GHz and reaches bit rates of up to 600Mbps. This standard uses more advanced technologies such as MIMO (Multiple Input Multiple Output), allowing the use of multiple antennas for emission and transmission.
- 802.11g: This wireless standard operates at a frequency of around 2.4GHz and provides transfer speeds of up to 54Mbps. It is an improvement over the 802.11b standard, using OFDM (Orthogonal Frequency-Division Multiplexing) modulation and DBPSK/DQPSK (Differential Binary/Quadrature Phase Shift Keying) modulations.

3.6.2 LoRa

LoRa (Long Range) technology is a type of long distance wireless communication with low power consumption. It is based on the use of spread spectrum modulation, i.e. the widening of the signal to be transmitted within the frequency spectrum with respect to the minimum bandwidth for data transmission. Thus, LoRa achieves very high ratios of range to energy used for transmission.

Within LoRa technology, we can highlight:

- In Europe, LoRa operates at frequencies of 868MHz in Europe, 915MHz in America and 433MHz in Asia.
- LoRa is capable of reaching up to 15km in urban areas and 50km in rural areas, although some trials have reached distances of up to 832km [27].

LoRa's transfer speed is relatively low, especially when compared to other technologies, which is why its area of interest lies in sending data mainly from sensors, since they are not bulky. Thus, handling bit rates between 0.3-50 kbps, it relies on a modulation called Chirp Spread Spectrum (CSS) that gives it this great range by using as carrier signal a Chirp type signal whose frequency is variable in time, as shown in Figure 10 [28].

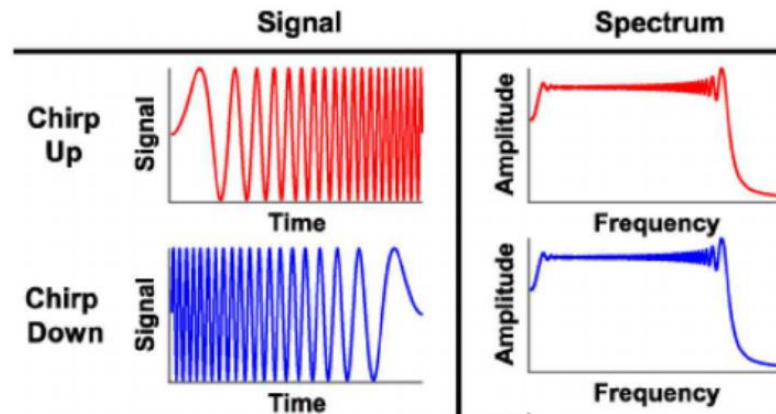


Figure 10: Temporal and spectral representation of Chirp signals used in Chirp Spread Spectrum (CSS) modulation in LoRa technology.

In general, the popularity of LoRa technology has been increasing in recent years, especially in areas of limited connectivity and in the specific field of the Internet of Things, thanks to its range and low power required to operate.

Regarding its integration into SBCs and Arduino, there are multiple small and inexpensive modules such as Dragino LoRa/GPS HAT [29] for Raspberry Pi or the Adafruit RFM96W LoRa Radio Transceiver Breakout [30] for Arduino.

3.6.3 Zigbee

Zigbee is a low-power wireless technology and data transfer widely used in the home automation, industry and sensor monitoring sectors. This technology is based on the IEEE 802.15.4 standard and has the following features:

- Like Wi-Fi and Bluetooth technologies, it operates at a frequency of 2.4GHz, which allows a data transfer of 20-250 kbps at a range of 10-400 m, depending on the working environment.
- Its power consumption and hardware size is notably much smaller than Wi-Fi technology. Compared to Bluetooth Low Energy (BTE), despite the low power consumption of both, Zigbee provides a much greater range, as well as a larger number of devices to connect.
- Zigbee works with a mesh topology, where we find the roles of coordinator, router and end device, as we can see in Figure 11 [31], which allows greater robustness and efficiency.

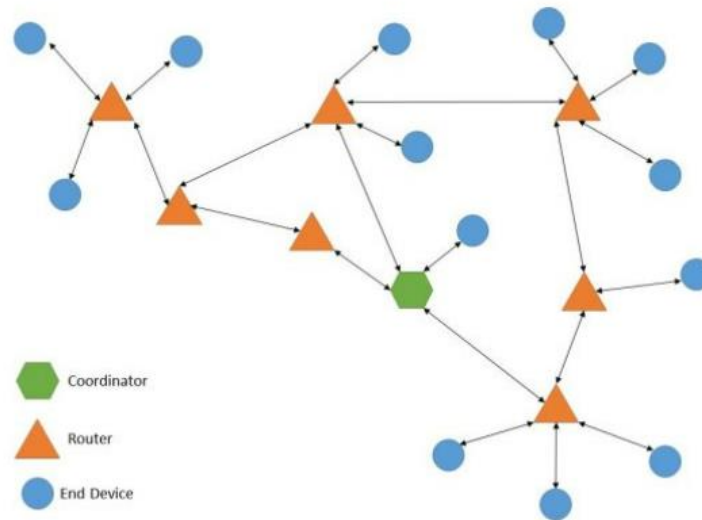


Figure 11: Example of network topology used in Zigbee.

Regarding their use in SBCs and Arduino, Zigbee modules are usually easy to integrate and take up little space, as well as being economically priced, such as Digi International's XBee [32] or Texas Instruments' CC2531 [33] module.

3.6.4 NB-IoT

NarrowBand Internet of Things (NB-IoT) [34] is a cellular network technology designed specifically for use in IoT. Thus, carriers enable data transmission for IoT devices at low power and bandwidth, providing a cost-effective and highly efficient alternative to conventional cellular network technologies.

Thus, some of the features of NB-IoT are:

- Working frequency around 200 kHz, reaching distances of up to several kilometers with data transfer of around 20-250 kbps.
- Single Carrier - Frequency Division Multiple Access (SC-FDMA) modulation similar to LTE (4G) technology.
- Like all other cellular network technologies, it requires an Internet Service Provider (ISP) to provide its services.

Regarding its use in SBCs and Arduino, there are currently multiple modules available in the market, such as the SIM7000E [35] model from SIMCom, which is compatible and integrable in both.

3.7 Connectivity protocols between devices

Inter-device connectivity protocols refer to those messaging protocols or network protocols whose objective is the exchange of messages between devices.

In IoT devices there is the use of previously existing protocols that are used for sending messages, as well as others specifically designed to be lightweight and limited to sending sensor data. Some of these protocols are shown below:

3.7.1 MQTT

MQTT (Message Queuing Telemetry Transport) is a lightweight and efficient message communication protocol used for data transfer between Internet of Things (IoT) connected devices, which has become one of the most popular protocols for communication between IoT devices due to its simplicity and efficiency.

MQTT is designed to communicate small data packets from IoT devices over a network efficiently and reliably, using a publish-and-subscribe approach. This means that devices can publish data to a specific topic, while other devices can subscribe to that topic and receive the data. An example of this message flow [36] can be seen in *Figure 2*:

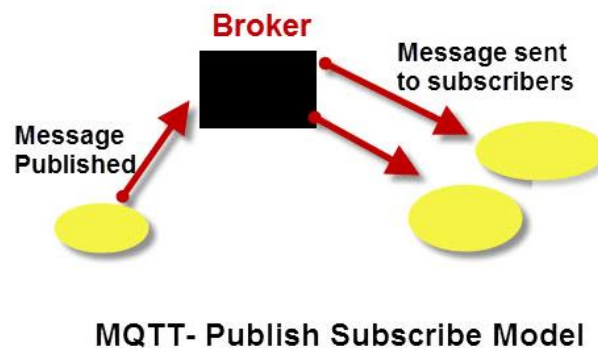


Figure 12: Basic operation diagram of the MQTT protocol.

One of the reasons why MQTT is so popular is its ability to operate on low-speed or unstable networks.

3.7.2 HTTP

HTTP or Hypertext Transfer Protocol is a protocol for the transmission of information through files. Its main design is focused on browsing between web servers, however, it is possible to use it for other purposes.

In this sense, HTTP is a protocol that does not store information about the state of the previous connection, so it is common to make use of cookies or information that the server stores in the client system for a period of time of validity to speed up the connection.

In the field of the Internet of Things, it is possible to use HTTP for different devices to send data to a web service, however, the energy and bandwidth consumption can be high [37], especially in networks where the number of devices is high, as we see in Figure 13:

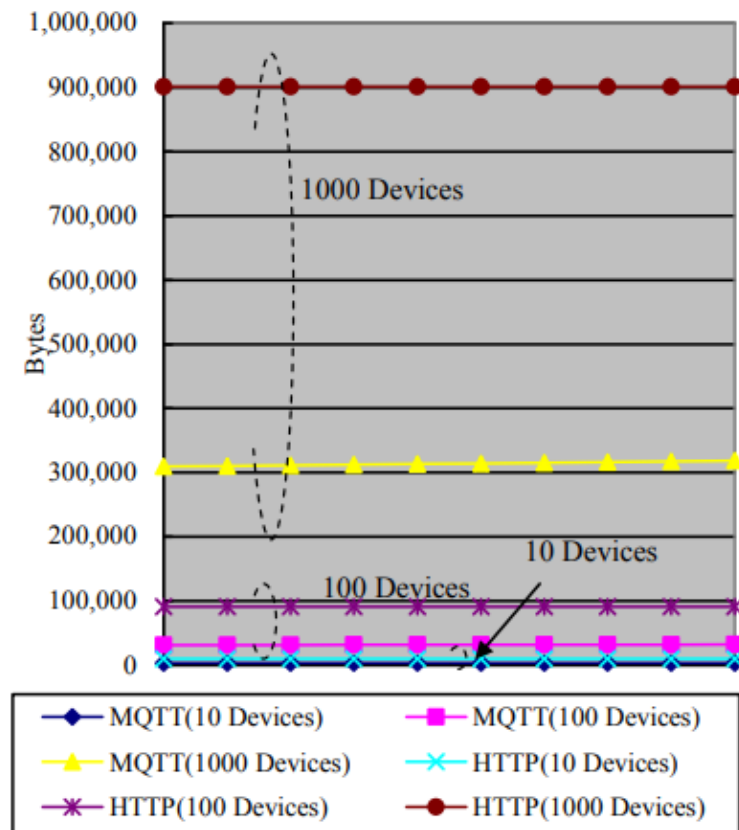


Figure 13: Comparación del ancho de banda requerido para MQTT y HTTP.

As we can see, the bandwidth required for 1000 devices in HTTP is three times the bandwidth required in MQTT.

3.7.3 CoAP

CoAP or Constrained Application Protocol is an application layer protocol derived from HTTP that works with the request/response concept. In this sense, HTTP works with the TCP transport protocol, however, CoAP works with UDP [38], presenting status codes just like MQTT but without guaranteeing the arrival of data.

An example of a CoAP network is shown in the Figure 14:

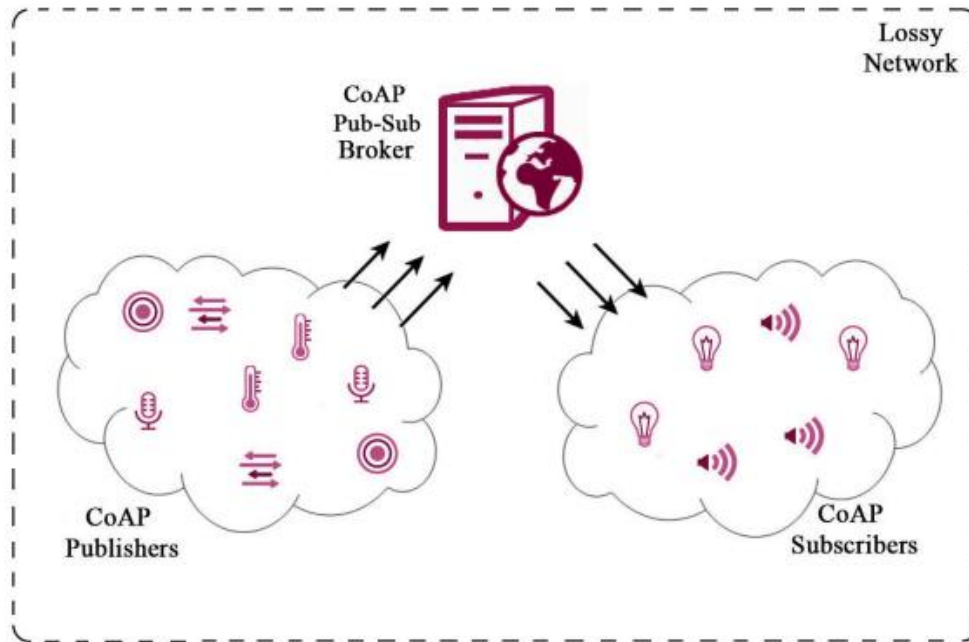


Figure 14: Simple network showing the operation of CoAP.

In general, CoAP tends to have higher latencies than MQTT, although its bandwidth consumption is significantly lower [39], as shown in Figure:

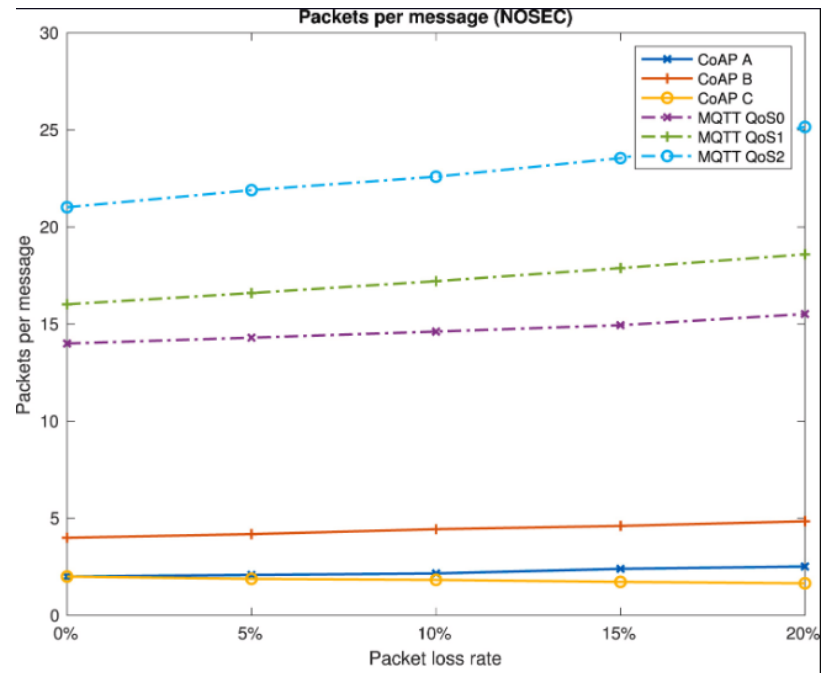


Figure 15: Number of packets per message for different CoAP and MQTT connections.

Finally, CoAP looks like a possible competitor to MQTT, however, the big difference between the two is that MQTT has been on the market much longer, which has made it more easily implementable thanks to the existing libraries and documentation.

4. MATERIAL AND METHOD

The development of this project has been carried out using an incremental iterative method based on iterations [40]. Thus, multiple iterations or interests are initially proposed within the project, defining a progressive planning and integration of the final device.

In each of these iterations, objectives have been proposed together with an associated temporal planning, which allows us to define a workflow and progress flow within the design and implementation of the device, based on the student's objective.

The different iterations carried out within the project are presented below:

4.1 ITERATION 1

In this first iteration of the project, the main objectives of the project and the approach have been defined, based on its budget and scope. The aim is to characterize the consumption of the different devices and to carry out a battery-based power supply study. Thus, the objectives and results are presented:

4.1.1 Objectives

- Define the project.
- Choose and acquire components based on market research, prioritizing documentation, consumption and Open Source.
- Integrate hardware and software for audio sample capture.
- Calculate the consumption of the different components involved.
- Integrate battery power supply for the device.
- Integrate HAT type auxiliary system for power control.

4.1.2 Expected results

- Definition of the device operation, being this an autonomous device capable of capturing audio samples based on events defined by software, processing them mathematically according to future interests

for application in Machine Learning algorithms and sending them via Wi-Fi wireless technology in an initial phase.

- Study of the different options offered by the current market to achieve the objectives of the project, taking into account economic and technical aspects.
- Realization of service in Python [41] by which we capture audio samples given an event.
- Optimization of the Linux Operating System so that certain components only start working by software definition. Disabling of processes or hardware modules whose operation is unnecessary in the project definition.
- Communication with development companies in order to obtain the consumption of the devices, which, initially, was not provided in the technical data sheet of the devices.
- Integration of HAT type power controller device for Raspberry Pi, together with RTC [42] included, to realize efficient battery consumption and turn on the device only at the most relevant hours.
- Device power supply based on external battery, obtaining the minimum needs of this to power up the device.
- Procurement of components required for the next phase.

4.2 ITERATION 2

In Iteration 2 we propose an efficient management of the acquired audio samples together with a study of the signals, by which, based on the extensive existing documentation on the subject, we obtain characteristic parameters of each signal based on certain mathematical formulas.

4.2.1 Objectives

- Investigate, study and integrate characteristic parameters of audio signals as input for Machine Learning algorithms.

- Develop service software for audio sample management.
- Develop service software for processing and storing characteristic parameters of existing audio samples.

4.2.2 Expected results

- - Expansion of the audio sample capturing service, by which we set a minimum duration of audio samples, saving them with the name of the date on which the sample was captured and storing them in the same sample directory.
- Development of system service and associated software by which we check the size of the directory where the samples are stored, efficiently managing the capacity of the device and processing all the samples obtained according to the interests, storing the derived data in a .csv file for further processing.
- Research and implementation of code to obtain certain characteristic parameters for each signal. Thus, once a storage alarm is triggered, all signals are processed, obtaining these parameters and storing them in the .csv file.

4.3 ITERATION 3

In this phase of the project, the development for sending messages to a MQTT type broker is proposed, considering the integration of MQTTS in the code [43].

In the same way, initially the use of a short-range wireless technology and simple integration, WiFi, has been contemplated, considering in future extensions of the device its adaptation to long-range technologies such as LoRa, for its integration in purely rural environments.

4.3.1 Objectives

- Develop software for sending data through MQTT and MQTTS.
- Manage communication and WiFi module to optimize process consumption.

- Development of processing algorithms and conversion to JSON (JavaScript Object Notation) [44] format for later sending.

4.3.2 Expected results

- A management function has been developed for MQTT and MQTTS communication, implementing certain callbacks [45] for communication control and debugging.
- Development of functions for saving and reading .csv files. In the same way, given a list of data, a function has been developed so that the message sent via MQTT is in JSON format, collected in the standard STD 90 RFC 8259 [46].

4.4 ITERATION 4

Finally, Iteration 4 is about performing a field test, analyzing the performance of the device along with the duration and possible improvements of the device.

4.4.1 Objectives

- Perform a field test analyzing the operation of the device.
- Identify areas for improvement.

4.4.2 Expected results

- Device autonomy based on mathematical calculations.
- Correct recording of acoustic signals and sending of obtained parameters.
- Switching the device on and off according to the defined schedule.
- Reception of parameters in MQTT broker.

5. IMPLEMENTATION AND DEVELOPMENT

This section will describe the overall process used for the implementation and development of the project, from its initial phases to the field testing of the functional device. In this sense and based on the state of the art, as well as the work methodology used, we will explain the decisions and the results that these produce in the device.

5.1 ITERATION 1

Within this first iteration, the main objective was to choose and obtain the components to be used, as well as to develop and test the recording of audio samples.

5.1.1 Raspberry Pi Zero W

The choice of device is one of the most important decisions of the project, since it constitutes the core of the device. In this sense, as we saw in section 3, we had the possibility of choosing between any of the SBCs available on the market or opt for the Arduino range. It was decided to opt for the use of a Raspberry Pi Zero W, a model that, as we can see in Table 1, is one of the models with the lowest power consumption.

In general, in a positive way we can highlight its following features:

- Low power consumption compared to competitors or other models of the same brand.
- Possibility of remote maintenance of the device thanks to WiFi connectivity.
- High processing capacity and connectivity thanks to its specifications.
- Use of open operating system and possibility of launching multiple services in parallel [47].
- Possibility to install additional modules (40 GPIO pins).
- Economical price of approximately 18€.

This choice resides, mainly, in the processing and cost ratio that the device provides. Another very viable option would have been the use of Arduino type devices, however we have chosen an SBC thanks to the possibility of working with an Operating System and defining multiple services in parallel.

5.1.2 Microphone selection

A technical study of the market was carried out for sample capture based on the focus of the project. Thus, among the fundamental aspects in the choice of the microphone, we have taken into account:

5.1.2.1 Polar diagram

The polar diagram or pickup pattern of a microphone can be defined as the shape or behavior that a microphone has with respect to the direction from which the sound comes [47]. Microphones usually have one or multiple patterns or lobes through which they receive the sound, so they are classified into different types of microphones according to their polar pattern (omnidirectional, bidirectional, supercardioid...). Some examples of these are shown in Table 5:


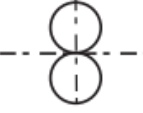


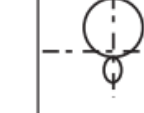
Microphone	Omnidirectional	Bidirectional	Directional	Supercardioid	Hypercardioid
Directional Response Characteristics					
Voltage output	$E = E_0$	$E = E_0 \cos \theta$	$E = \frac{E_0}{2}(1 + \cos \theta)$	$E = \frac{E_0}{2}[(\sqrt{3} - 1) + (3\sqrt{3} - 3)\cos \theta]$	$E = \frac{E_0}{4}(1 + 3 \cos \theta)$
Random energy Efficiency (%)	100	33	33	27	25
Front response Back response	1 1	1 1	∞ 1	3.8 1	2 1
Front random response Total random response	0.5 0.5	0.5 0.5	0.67 0.67	0.93 0.93	0.87 0.87
Front random response Back random response	1 1	1 1	7 7	14 14	7 7
Equivalent distance	1	1.7	1.7	1.9	2
Pickup angle (2θ) for 3 dB attenuation	—	90°	130°	116°	100°
Pickup angle (2θ) for 6 dB attenuation	—	120°	180°	156°	140°

Table 5: Comparison of different microphones based on their polar diagram.

For the project specifications, an omnidirectional microphone has been chosen, since the device will be located in open spaces and, unlike other use cases, it is necessary to be able to pick up sound coming from all possible directions. A graphical example of the operation of an omnidirectional microphone is shown in Figure 16:

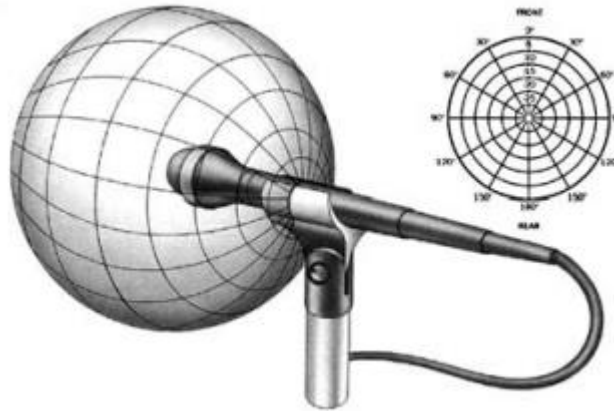


Figure 16: Graphical example of omnidirectional microphone operation.

Additionally, this type of microphones allows us to significantly reduce the probability of an effect produced in other types of microphones with unidirectional polar diagram, the phase cancellation, observable in Figure 17, by which a signal is canceled in the receiver due to the effects of a 180° phase shift between the original signal and the same signal produced by a reflection in the environment.

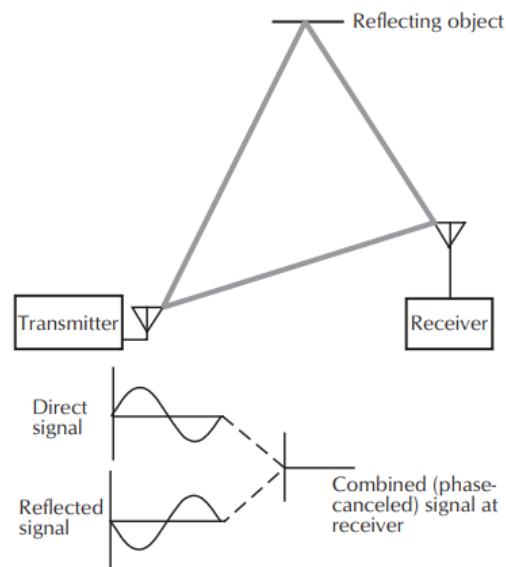


Figure 17: The phase cancellation effect [15].

5.1.2.2 Frequency response

Another important parameter in the choice of microphone is its frequency response. As mentioned in section 3.2.3, a characteristic of condenser microphones is that they often provide a flatter and wider frequency response, allowing us to capture a wider frequency range without "filtering" and not being more sensitive to some frequencies than others.

In the same way, condenser microphones usually have a higher sensitivity than other microphones, which is a crucial factor for recording in nature, where the transmitter is often far away from the microphone and the sound pressure level received is low.

It is for this reason that the choice of microphone for our project has been a condenser microphone. Specifically, we have chosen to use the Primo EM272 microphone [48], a condenser microphone of high sensitivity, lightweight and economical that will allow us to receive signals at frequencies in the range of 60 Hz to 18 kHz, with a cost of 37 €. An image of this microphone is shown in Figure 18:



Figure 18: Primo EM272 condenser microphone.

This microphone requires a 3V power supply and has a maximum consumption of 600 μ A, which makes it a more than acceptable solution for our project.

Additionally, we have opted for the use of an Antipop/Windshield, a device that is commonly installed on microphones to prevent unwanted wind or environmental noise effects in outdoor recordings. An image of the Primo EM272 microphone with a windshield installed is shown in Figure 19:



Figure 19: Primo EM272 microphone with windshield installed.

5.1.2 Energy management system

Power management and device autonomy is one of the most critical points, especially if we want the device to work for a long time. Thus, a simple 5000-20000mAh battery can give us a lot of hours of use if we work with Arduino or a low-power SBC, as we saw in section 3.1.1.4.

However, the project specifications did not define a need for the device to be recording continuously, a decision highly supported by the findings of Diego Gil and Diego Llusia in [49], where, despite not being able to specify the reason for the increase in birdsong at dawn, it occurs mainly in the early morning hours. In the same way, there are some species that sing when the light begins to decrease [50], as is the case of the nightingale or quail.

Therefore, we have defined a working schedule for our device from 07:00 to 09:00 and from 20:00 to 22:00, on a daily basis, seeking to work in the hours when birds sing the most. This functionality is impossible to achieve in Raspberry Pi Zero W without an external device, where after a review of the market, we have chosen to use the UUGEAR Witty Pi 3 Mini device [51].

Witty Pi 3 Mini is a small board with Real Time Clock (RTC) that allows us to manage the power of our Raspberry Pi. It has a pHAT format by which we can use it as an add-

on module to our Raspberry Pi, increasing very slightly the size of our total module, as we can see in Figure 20:

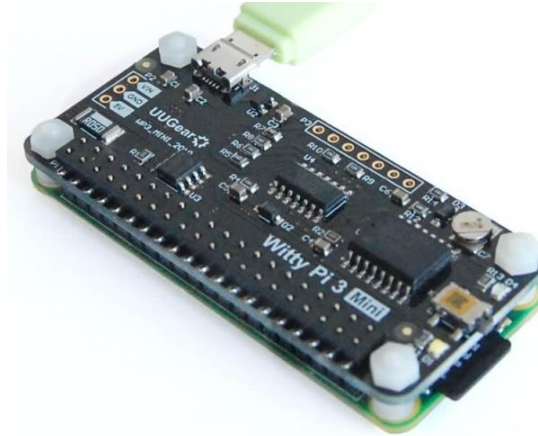


Figure 20: Raspberry Pi Zero W with Witty Pi 3 Mini installed.

This device allows us to know multiple relevant data, such as the voltage provided by the battery, the specific consumption of the device at a given time or easily synchronize the current date of the Raspberry Pi, obtained through an NTP (Network Time Protocol) server with its RTC (model DS3231, high precision), allowing to have a high precision time synchronization for long periods of time. An example of the interface provided is shown in Figure 21:

```
SOLMoth@SOLMoth:~/Desktop/wittypi $ ./wittyPi.sh
=====
Witty Pi - Realtime Clock + Power Management for Raspberry Pi
          < Version 3.51 >      by Dun Cat B.V. (UUGear)
=====
>>> Current temperature: 25.50°C / 77.9°F
>>> Your system time is: Sun 19 Mar 2023 11:04:46 CET
>>> Your RTC time is:    Sun 19 Mar 2023 11:04:45 CET
>>> Vout=4.48V, Iout=0.17A
Now you can:
 1. Write system time to RTC
 2. Write RTC time to system
 3. Synchronize time
 4. Schedule next shutdown
 5. Schedule next startup
 6. Choose schedule script [in use]
 7. Set low voltage threshold
 8. Set recovery voltage threshold
 9. View/change other settings...
10. Reset data...
11. Exit
What do you want to do? (1~11)
```

Figure 21: Witty Pi 3 Mini device configuration panel.

Additionally, and the main reason for its choice, Witty Pi 3 Mini is designed to allow you to define a calendar or schedule for turning off and turning on your device in a simple way. Thus, the user is provided with a service by which a .wpi file is defined to specify the schedule to be followed.

This .wpi file contains, in a simple way, data referring to the specific schedule to be followed, in order to subsequently indicate, by means of "Option 6", which can be seen in Figure 22, the desire to adjust to this schedule. An example of a .wpi file is shown in Figure 22:

```
# Turn on Raspberry Pi for 5 minutes, in every 20 minutes
BEGIN    2021-02-01 00:00:00
END      2035-07-31 23:59:59
ON       M5      # keep ON state for 5 minutes
OFF      M15     # keep OFF state for 15 minutes
```

Figure 22: .wpi file defined for use in Witty Pi 3 Mini running a 5 minute on every 20 minute off schedule.

The use of Witty Pi 3 Mini has allowed us to know, as we can see in Figure 21, the consumption in full operation of our device, which is usually between 150-170mA.

Thus, we can calculate the current lifetime of the *SOLMoth* device with an 8000mAh battery, with which tests have been performed:

$$Duration (h) = \frac{8000mAh}{170mA} = 47,05 \text{ h}$$

In the same way, knowing that the device is currently working 4 hours a day, we would obtain a total duration of:

$$Total \text{ duration (days)} = \frac{47,05 \text{ h}}{4 \text{ h/day}} = 11,76 \text{ days}$$

In case of using a 20000mAh battery, this total duration would be approximately one calendar month, which is within the initial expectations of the project.

According to the Witty Pi 3 Mini documentation [47], the standby consumption is approximately 1mA, as we can see in Table 6, being able to work in temperatures between -30°C and 80°C, allowing its operation in Spain at any time of the year.

StandbyCurrent	~ 1mA
Operating Temperature	-30°C~80°C (-22°F~176°F)

Table 6: Witty Pi 3 Mini device power consumption in standby mode.

5.1.3 Data collection

The capture of the audio samples has been done using the Python *pyAudio* library by which we have defined a `recorder` class where the different methods of recording, listening and writing to disk have been structured. This class can be visualized globally in Figure 23:


```
class Recorder:

    @staticmethod
    def rms(frame):...

    def __init__(self):...

    def record(self):...

    def write(self, recording):...

    def listen(self):...
```

Figure 23: Recorder class defined for recording audio samples on the *SOLMoth* device.

In this sense, each of the methods or functions within this class acts thanks to global variables of the system referred to sample capture, which are discussed below:

- Block or chunk: In audio recording, a chunk is a section of data that is processed or transmitted at a given time. Referring to the Operating System, we could define it as a data buffer, so that we send blocks of data instead of working continuously with the data we receive.

This value has been defined as 1024, a value commonly used in sound recording in SBCs such as Raspberry Pi

- Bit depth: Amount of information used to represent the amplitude of the audio signal. A graphical display of this parameter is shown in Figure 24 [52].

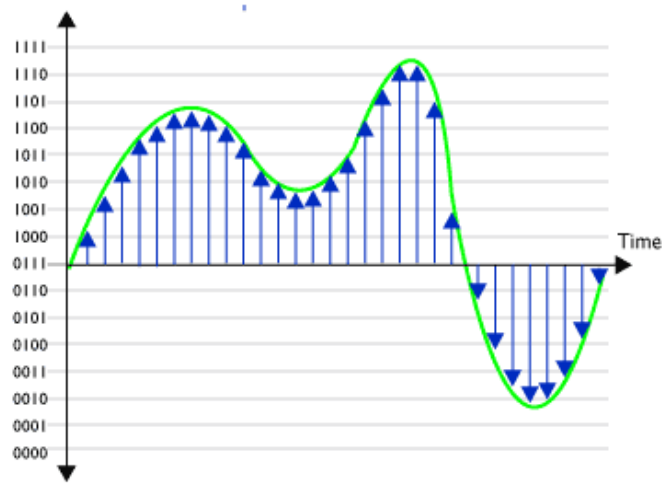


Figure 24: Graphical visualization of the bit depth concept in audio sample coding.

Thus, a bit depth of 16 bits has been chosen to capture samples in the project, which is equivalent to 65535 different levels of definable amplitude, being sufficient for the scope of the project.

Sometimes, in studio or professional environments, it is common to see 24 bit depth. However, the quality of the recording does not depend solely on this parameter, so the relationship between quality and computational gain has not been considered sufficiently relevant for implementation in the project.

- **Sampling frequency:** The sampling frequency or sampling rate is the number of samples per second taken from the analog signal for its conversion to digital signal. This can be seen again in Figure 24 as it is the rate at which samples are taken from the analog signal. Thus, in digital audio recording, the most commonly used sampling rate is 44.1kHz.

As we discussed above, our microphone reaches frequencies up to 18kHz, so 44.1kHz sampling rate is sufficient for our project. This argument is supported by the Nyquist Theorem [53] whereby, in order to avoid aliasing effects, the sampling frequency should be at least twice as high as the highest frequency to be recorded.

It is possible to modify these parameters in order to capture higher frequencies, where we find animal songs such as bats. However, those microphones that pick up

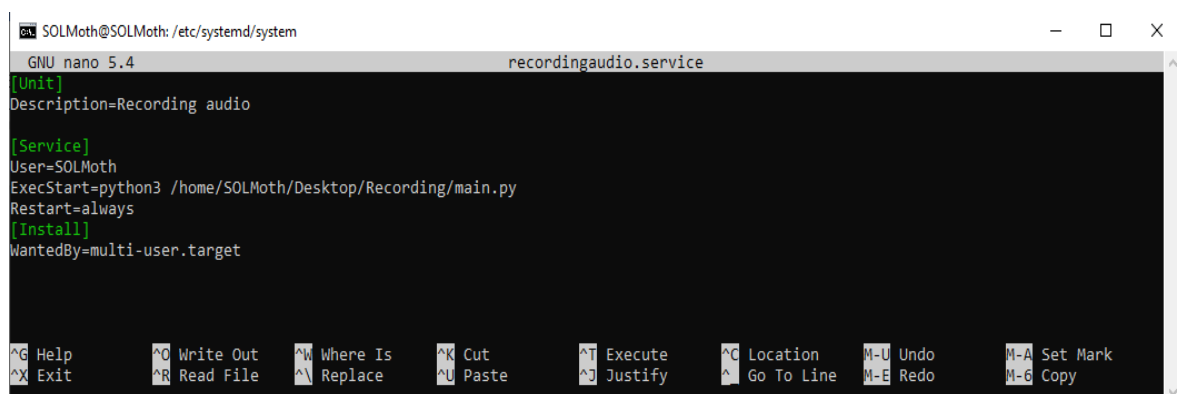
frequencies above 20kHz have a considerable cost increase that make us exceed the initial budget of the project.

The data acquisition occurs in such a way that, continuously in time, the device is in active listening, i.e., receiving audio data packets at each instant of time. Thus, an RMS (Root Mean Square) calculation function or root mean square value of the received signal is defined. In this sense and knowing that we work with a bit depth of 16 bits, we will always obtain maximum amplitude values of ± 32767 , so that the RMS calculation allows us to establish a threshold value from which we define that an event is being triggered, launching the recording and writing of the audio. That is, we define a percentage of the amplitude of the received signal that, once exceeded, is detected in software as an event and therefore, the recording starts.

This software, written in the Python programming language [41] must be launched as a service of the operating system. In this sense, all the executables of the project are located within a common directory and are defined in such a way that:

- They are executed automatically every time the device is turned on.
- In case of error, it has an automatic restart mechanism.

Thus, we will define a file called *recordingaudio.service* that we will place in the */etc/systemd/system/* directory. The contents of this file can be seen in Figure 25 and once defined, it will be necessary to enable the service and reload the system configuration by executing the *daemon-reload* command.



```
SOLMoth@SOLMoth: /etc/systemd/system
GNU nano 5.4 recordingaudio.service
[Unit]
Description=Recording audio

[Service]
User=SOLMoth
ExecStart=python3 /home/SOLMoth/Desktop/Recording/main.py
Restart=always
[Install]
WantedBy=multi-user.target

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute  ^C Location  M-U Undo    M-A Set Mark
^X Exit      ^R Read File ^_ Replace   ^U Paste     ^J Justify  ^_ Go To Line M-E Redo    M-G Copy
```

Figure 25: Contents of the *recordingaudio.service* file by which the system service is defined.

In this way, the first part of the program, by which we actively listen, would be defined and, once the amplitude threshold for the received signal is exceeded, defined in the `rms` function of the `recorder` class, which we can see in Figure 23, we record audio signals with a minimum duration of 5 seconds and store them in a common directory, where each recording has as name the exact date of its recording, as we can see in Figure 26:

```
SOLMoth@SOLMoth:~/Desktop/Recording $ cd records/
SOLMoth@SOLMoth:~/Desktop/Recording/records $ ls -l
total 3288
-rw-r--r-- 1 SOLMoth SOLMoth 303148 Apr  6 09:51 20230406-095103.wav
-rw-r--r-- 1 SOLMoth SOLMoth 458796 Apr  6 09:51 20230406-095138.wav
-rw-r--r-- 1 SOLMoth SOLMoth 2170924 Apr  6 09:53 20230406-095301.wav
-rw-r--r-- 1 SOLMoth SOLMoth 417836 Apr  6 09:55 20230406-095550.wav
```

Figure 26: Example of directory contents where the audio samples recorded by the *SOLMoth* device during its execution are stored.

5.1.4 Wi-Fi module control

Raspberry Pi Zero W uses a Broadcom BCM43438 [54] Wi-Fi module with Wi-Fi and Bluetooth connection capabilities. In this sense and according to the official documentation of the device, the consumption of this can vary from a few milliamps to more than 100mA, so a simple mechanism has been developed to activate the use of this only when it is necessary to send data, thus increasing the autonomy of the device.

To do this, we have developed two functions included in the data processing service, so that when sending data, the WiFi module is initialized beforehand by inserting system commands through Python, as we can see in Figure 27:

```
def turn_on_wifi():  
    """  
    Turns on the Wi-Fi module on the Raspberry Pi  
    """  
    subprocess.run(["sudo", "ifconfig", "wlan0", "up"])
```

Figure 27: Wi-Fi module power-on function on Raspberry Pi Zero W via system command.

In other words, the Wi-Fi module is only switched on when it is necessary to send data to the MQTT broker and is subsequently switched off in order to reduce the device's energy consumption and, therefore, increase its autonomy.

5.2 ITERATION 2

The software is one of the keys to the project, since it is where we define how we act, on the basis of which events and what we do with the data generated. In general, section 5.2.1 is divided into multiple subsections, where we begin by explaining the different variables to be obtained and end with the management of the device and data storage.

5.2.1 Data processing

Data processing is the methodology followed in the project for the treatment of audio files stored by the signal recording service. In this sense, we will first process the data in order to obtain certain parameters of interest and, once processed, store them correctly for their subsequent sending to the MQTT broker.

5.2.1.1 Parameter acquisition

The acquisition of parameters consists of reading each of the signals acquired by the device for processing, finally obtaining a list of values for each of the parameters. In this initial phase of the project, we have chosen to calculate four statistical parameters

for each of the signals: Zero Crossing Rate (ZCR), Spectral Entropy, Spectral Centroid and Spectral Roll-Off Factor, which are explained below:

5.2.1.1.1 Zero Crossing Rate

The Zero Crossing Rate (ZCR) is the ratio by which the signal waveform crosses through the zero amplitude value [55]. That is, it is a measure of the number of times the signal changes its polarity in each period.

The ZCR is defined such that:

$$zcr = \frac{1}{T-1} \cdot \sum_{t=1}^{T-1} 1_{\mathbb{R}<0} \cdot (s_t s_{t+1} - 1) \quad [3]$$

Being s the audio signal and calculated in each time period T .

This parameter is commonly used in audio and music detection and analysis.

5.2.1.1.2 Spectral Entropy

Spectral Entropy is a measure of the spectral distribution of power. In this sense, we can say that Spectral Entropy is a way of quantifying the amount of information contained in the signal spectrum, so that large values for this parameter indicate greater complexity in the frequency content of the signal.

This parameter is based on Shanon's concept of entropy for information theory, where the level of randomness or certainty is measured in a probability distribution.

Spectral Entropy is defined such that:

$$H(t) = - \sum_{m=1}^N P(t, m) \log_2 \cdot P(t, m) \quad [4]$$

Where N is the total number of points in frequency and P is the probability distribution such that:

$$P(m) = \frac{S(m)}{\sum_i S(i)} \quad [5]$$

$S(m)$ being the spectral power of the signal such that:

$$s(m) = |X(m)|^2 \quad [6]$$

Where $X(m)$ is the Discrete Fourier Transform for $x(n)$.

The calculation of this parameter is intended for the recognition of bird species due to the difference in complexity and variability in the song of species. The use of this parameter has been shown to improve the performance of certain audio signal classification algorithms [56].

5.2.1.1.3 Spectral Centroid

The Spectral Centroid [57] is commonly referred to as the center of gravity of the spectral power distribution of the signal. Thus, it is a measure of the spectral content of the signal, where a high value for the Spectral Centroid indicates higher concentration of energy at high frequencies and vice versa.

The Spectral Centroid is defined such that:

$$\frac{\sum_{n=0}^{N-1} f(n)x(n)}{\sum_{n=0}^{N-1} x(n)} \quad [7]$$

Where f_n is the frequency value for the position n for a bandwidth between 0 and $N-1$. $x(n)$ refers to the spectral value at position n .

Being a parameter of recurrent use in the classification and recognition of audio signals [58].

5.2.1.1.4 Roll-Off Factor

The Roll-Off Factor is a measure of how far the signal energy exceeds a certain probability in the cumulative probability distribution [59]. That is, it is the specific frequency by which we find below a certain percentage of the signal energy.

This parameter allows us to identify the silence of the signal, so that:

$$\sum_{k=b_1}^i s_k = k \cdot \sum_{k=b_1}^{b_2} s_k \quad [8]$$

Where s_k is the spectral value at position k and b_1 and b_2 is the bandwidth where the Roll-Off factor is defined.

5.2.1.2 Data management

Once the different functions that calculate the parameters mentioned above have been defined, a data extraction function is used to calculate the different parameters and return them in the form of a list, as shown in Figure 28:

```
def extract_audio_features(filename):
    """
    Extract audio features from an audio file.

    :param filename: path to the audio file.
    :return: a tuple containing the zero-crossing rate, the spectral centroid, the spectral entropy, and the spectral
    rolloff of the audio signal.
    """
    [Fs, x] = audioBasicIO.read_audio_file(filename)
    x = audioBasicIO.stereo_to_mono(x)
    zcr = zero_crossing_rate(x)
    centroid = spectral_centroid(x, Fs)
    entropy = spectral_entropy(x, 10)
    rolloff = rolloff_factor(x, 0.8, Fs)

    return zcr, centroid, entropy, rolloff
```

Figure 28: Function used in the SOLMoth project to extract parameters from an input audio file.

For the calculation of Spectral Entropy, we define a number of blocks in which we divide the signal before calculating it, for greater accuracy and analysis. Similarly, in

the case of calculating the Roll-Off factor, a cutoff at 80% of the signal has been defined.

A method has been defined in which the size of the audio sample storage directory is checked every minute. If the directory size exceeds 100MB, data extraction from all files in the directory is executed, as we can see in Figure 29 where the method is shown:

```
def check_file_size(data_directory: str, csv_path: str) -> None:
    """
    Check the size of files in a given directory and extract data if the size limit is exceeded

    Args:
        data_directory (str): Path of the directory to check the size of files in
        csv_path (str): Path of the CSV file to save the extracted data to

    Returns:
        None
    """
    size = 0
    for ele in os.scandir(data_directory):
        size += os.path.getsize(ele)
    print("Current folder size: {}".format(size))
    if size > 1e+8:
        print("Size limit detected. Extracting data features...")
        dir_data_extraction(data_directory, csv_path)
```

Figure 29: Function used in the SOLMoth project to check the size of a given directory.

Where the *dir_data_extraction* function is responsible for executing the *extract_audio_features* function on each of the files in the directory, storing the data obtained for each file in a row of a .csv file, as we can see in Figure 30:

```
def dir_data_extraction(data_directory: str, csv_path: str) -> None:
    """
    Extract data from audio files in a given directory and save it to a CSV file

    Args:
        data_directory (str): Path of the directory containing audio files to extract data from
        csv_path (str): Path of the CSV file to save the extracted data to

    Returns:
        None
    """
    for filename in os.listdir(data_directory):
        f = os.path.join(data_directory, filename)
        if os.path.isfile(f):
            data = extract_audio_features(f)
            print(filename + " || Feature extraction: " + str(data))
            save_into_csv(data, csv_path)
            os.remove(f)
            print("File removed successfully")
```

Figure 30: Function used in the *SOLMoth* project to iterate through all audio files in a given directory, obtaining the parameters returned by the *data_extraction* function, storing them in a .csv file, and finally deleting the file.

Once a file has been processed and its parameters obtained, stored in the .csv file, it is deleted to provide free space to the device. In the same way as with the audio recording service, a system service is defined that is automatically executed once the device is turned on and restarted in case of failure. Thus, the general functionality of this service is to check the size of the directory where audio recordings are stored every minute, where if it exceeds a certain size, it processes all the files and stores those parameters in a .csv file.

An example of the service status once defined can be seen in Figure 31:

```
● processingaudio.service - Processing audio
   Loaded: loaded (/etc/systemd/system/processingaudio.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2023-03-20 18:00:38 CET; 2 weeks 2 days ago
     Main PID: 259 (python3)
        Tasks: 1 (limit: 415)
            CPU: 1h 1min 16.293s
       CGroup: /system.slice/processingaudio.service
               └─259 python3 /home/SOLMoth/Desktop/Processing/main.py
```

Figure 31: Status of the *processingaudio.service* once it is defined by the system.

5.3 ITERATION 3

In this phase of the project, the development for sending messages to an MQTT broker type is proposed. In this sense, various return and message sending management functions have been defined using this protocol to simplify and implement their integration into the program.

Thus, a function has been developed where the publication is managed in an MQTT broker whose topic is composed of the name of the device and its MAC address, as we can see in Figure 32:

```
Checking directory size at 2023-04-07 10:30:53.878432
Connected succesfully, sending data to server
Topic is: SOLMoth/c87f54075c6b
```

Figure 32: Example of system output in sending data to an MQTT broker.

In this sense, a callback or return function is defined that is executed when trying to make an MQTT request, managing the return code of the connection and allowing data to be sent if it is correct.

Thus, an additional function has been defined to read the .csv file where all the obtained parameters for each of the audios are stored and sent as a payload within the function presented in Figure 33:

```
def mqtt_request(host: str, port: int, topic: str = None, payload: str = None):
    """Publishes data to the specified MQTT broker.

    Args:
        host: A string that contains the IP address of the MQTT broker.
        port: An integer that contains the port number of the MQTT broker.
        topic: A string that contains the topic where the data is to be published.
        payload: A string that contains the data to be published.

    Returns:
        A boolean that represents the status of the connection.
    """
```

Figure 33: Header of the mqtt_request function used for the management and sending of data to a given MQTT broker.

Within MQTT, it is possible to switch to MQTTS by adding the use of credentials as well as the use of a TLS certificate, both of which are included in the project and the different connection management functions, but not used in the current version of the device.

Once the message is generated in JSON format with the data read from the .csv file, it is published to the MQTT broker as shown in Figure 34:

```
{
  "20230319-110612.wav": [
    0.012347773552426312,
    0.8613011241235157,
    3.1641528703537944,
    0.748096530720339
  ],
  "20230319-111207.wav": [
    0.00997258643179509,
    0.6094619691748331,
    3.241736253568175,
    0.8691057477678571
  ],
  "20230319-111314.wav": [
    0.015587469177188208,
    0.763516277442693,
    3.1306965860640528,
    0.8813063401442308
  ]
}
```

Figure 34: Example of message received in MQTT broker where the processed data from three audio samples have been published.

Where each object labeled with a date in the sample capture contains the four calculated parameters in the order discussed in this document, namely: ZCR, Spectral Entropy, Spectral Centroid, and Roll-Off Factor.

5.4 ITERATION 4

Finally, in order to visualize the entire system, Figure 35 shows an image of the *SOLMoth* device without its casing, where it is possible to observe the components: Raspberry Pi Zero W, Witty Pi 3 Mini, Sound Blaster Play! 4 sound card, Primo

EM272 microphone, and a portable 8000mAh battery, along with the wiring of the system.



Figure 35: Image of the complete first version of the SOLMoth device without an external casing

Thus, a publication of results every 10 minutes has been observed, specifying a maximum size for the data storage folder of 50 MB. This equates to 5 MB per minute of data acquisition, which, based on the experiments carried out, means at least one audio file stored every minute.

Similarly, the duration of the device has been approximately 10 natural days with a portable battery of 8000mAh, which is slightly lower than calculated based on the information provided by Witty Pi 3 Mini. This result is expected, as the current spikes at the start of the device are not being taken into account.

Finally, proper storage management and communication with the MQTT broker have been observed. Regarding Wi-Fi communication, it represents a significant limitation

in terms of the distance to the nearest router or repeater, obtaining maximum distances of 40m outdoors.

6. RESULTS AND DISCUSSION

In this section, we will discuss the results obtained once each part of the first functional prototype of SOLMoth has been defined and configured. The following achievements have been accomplished:

- Configuration and energy optimization of the Operating System, based on techniques for accessing device modules, controlling their turning on and off.
- Providing device autonomy of over two weeks using an external 8000mAh battery. Integration and configuration of power management device and precise scheduling of operation.
- Integration of sound card and microphone in Raspberry Pi using Open Source, configuring their operation and controlling their behavior via software.
- Development of software and audio recording system service based on defined events.
- Development of software and system service for audio sample processing based on parameters of interest for future application of Artificial Intelligence algorithms.
- Proper management of device storage.
- Development of correct communication with MQTT broker based on return functions that manage the exchange of data. Similarly, TLS security mechanisms have been implemented for its future adaptation to MQTTS.

Once the complete system and all its variables were defined, a final field test was carried out for the project, obtaining the following results:

- Publication of results approximately every 10 minutes. The number of files sent depends solely on the given circumstances, i.e., the behavior of the environment in that time period.

- Device duration around 10 days with a portable battery of 8000mAh. Slightly lower than the calculated value based exclusively on the data provided by Witty Pi 3 Mini under normal operating conditions.
- Correct management of storage and communication with MQTT broker. Limitations in distance of around 40m are observed due to the wireless Wi-Fi technology, which directly affects the energy consumption of the device.

The code used in the different services of the system can be found in the following GitHub repositories, so they are available to the general public:

- Audio recording service:
https://github.com/diegomm27/SOLMoth_AudioRecording
- Audio processing and data sending service:
https://github.com/diegomm27/SOLMoth_AudioProcessing

7. CONCLUSIONS

The creation of SOLMoth has constituted the development of an ambitious project whose development limits are imposed by the temporal planning of the subject. In this sense, the project encompasses multiple simultaneous developments that converge into a final device, making it a complex process of integration and debugging.

SOLMoth aims to characterize animal species using an autonomous device capable of obtaining audio samples, processing them, and communicating wirelessly for data transmission. Thus, a work methodology based on the integration of Open Source devices, economically priced and accessible to everyone, has been developed.

Firstly, the results obtained in the recording of audio samples have presented a high signal-to-noise ratio. Similarly, the multiple variations in the detection threshold have caused a correct adjustment of it, so that false signal recordings occur with very low probability. Both measures have been subjectively obtained through multiple audio recordings captured directly from the device.

Likewise, a proper management of the device's storage, which is limited by its SD card, has been developed, obtaining characteristic parameters of the different signals obtained in search of a future implementation of an Artificial Intelligence algorithm for the classification of animal species. These parameters have been possible to send thanks to the development of communication through the MQTT protocol and Wi-Fi wireless technology.

In addition, the autonomy of the device has been achieved based on initial duration objectives, thanks to the establishment of a work schedule, which has drastically reduced its daily energy consumption. Therefore, we can consider the device as autonomous and capable of remaining semi-perpetually operational.

Finally, it is possible to conclude that SOLMoth is a robust, autonomous, and useful device in the field in which it has been developed. Thus, in this project, it has been possible to present an initial phase of the device where all the previously commented functionalities are implemented and marked as objectives in the temporal planning of it, allowing for new improvements and modifications.

8. FUTURE WORK

The work carried out represents an initial phase of the device that leads to multiple additional developments. Thus, we present below the possible new avenues for the development of the SOLMoth device that would improve it:

- Adaptation to LoRa wireless communication technology: Changing the Wi-Fi technology to LoRa would significantly increase the range of the device, allowing it to be installed in remote areas and increasing autonomy.
- Development of a species classification algorithm based on Artificial Intelligence: First, it would be necessary to have a database of animal species in the rural area of Extremadura so that, together with the samples obtained by *SOLMoth*, the classification of species could be carried out.
- Development of a waterproof case for SOLMoth: This case would allow SOLMoth to operate in any season, regardless of the weather.
- Implementation of MQTTS: Development of a private MQTT broker along with the use of credentials and TLS certificate.

BIBLIOGRAPHICAL REFERENCES

1. **Benjamin M. Van Doren.** How migratory birds might have tracked past climate change. January 10, 2022 119 (3) e2121738119. <https://www.pnas.org/doi/10.1073/pnas.2121738119>
2. **ARDUINO.** Open source hardware and software development. [Online] 2023. [Cited: abril 15, 2023] [Arduino - Home](https://www.arduino.cc/)
3. **Raspberry Pi.** Single board computers from *Raspberry Pi Foundation*. [Online] 2023. [Cited: abril 15, 2023] <https://www.raspberrypi.com>
4. **O. Simeone,** "A Very Brief Introduction to Machine Learning with Applications to Communication Systems," in *IEEE Transactions on Cognitive Communications and Networking*, vol. 4, no. 4, pp. 648-664, Dec. 2018, doi: 10.1109/TCCN.2018.2881442.
5. **GEERLING, Jeff.** Power Consumption Benchmarks. [Online] 2023. [Cited: febrero 02, 2023] [Power Consumption Benchmarks | Raspberry Pi Dramble](https://www.geerling.com/power-consumption-benchmarks-raspberry-pi-dramble)
6. **Beagleboard.** Open hardware computers. [Online] 2023. [Cited: abril 15, 2023] <https://beagleboard.org/bone>
7. **Open Source.** Modern software ecosystem. [Online] 2023. [Cited: abril 15, 2023] <https://opensource.org>
8. **Hardkernel.** Open software company. [Online] 2023. [Cited: abril 15, 2023] <https://www.hardkernel.com>
9. **Tutorialspoint.** Company that provides learning material on technical subjects. [Online] 2023. [Cited: abril 15, 2023] [Difference between Microprocessor and Microcontroller - Tutorialspoint](https://www.tutorialspoint.com/microcontroller/microcontroller-tutorialspoint)
10. **TORRES, Abigail.** Embedded Linux Board Comparison. Part 2 – Power Usage, Temperature, Development. [Online] 2023. [Cited: febrero 03, 2023]. <https://www.radiolocman.com/review/article.html?di=163106>
11. **Firmware.** Definition and specifications. [Online] 2023. [Cited: febrero 03, 2023]. <https://www.techtarget.com/whatis/definition/firmware>
12. **Aitzol Zuloaga, Jaime Jiménez, Jesús Lázaro, Carlos Cuadrado, Unai Bidarte.** Departamento de Tecnología Electrónica, Universidad del País Vasco.

Definición de máquinas de estados, eventos y acciones en pequeños procesadores.

[Online] 2023. [Cited: marzo 20, 2023]. [Departamento de Tecnología Electrónica - UPV/EHU](#)

13. **M. Olmo, R. Nave.** Dynamic microphones. [Online] 2023. [Cited: febrero 03, 2023] [Microphones - GSU](#)
14. **Shure Incorporated.** Shure Beta 58A datasheet. [Online] 2023. [Cited: febrero 03, 2023] https://www.shure.com/es-ES/productos/microfonos/beta_58a?variant=Beta%252058A
15. **Glen M. Ballou** Dynamic microphones. [Online] 2023. [Cited: febrero 03, 2023]
16. **Goang-Fann Co. Ltd.** Superlux R102 y Superlux R102MKII datasheet. [Online] 2023. [Cited: febrero 09, 2023] <https://manualzz.com/doc/55722326/superlux-r102-mkii-classical-ribbon-microphone-specificat...>
17. **M. Olmo, R. Nave.** Micrófonos de condensador. [Online] 2023. [Cited: febrero 09, 2023] [Microphones - GSU](#)
18. **Sennheiser.** e-965 datasheet. [Online] 2023. [Cited: febrero 09, 2023] <https://es-mx.sennheiser.com/vocal-condenser-microphone-studio-live-recording-e-965>
19. **Keiron O'Shea, Ryan Nash.** An Introduction to Convolutional Neural Networks. [arXiv:1511.08458](https://arxiv.org/abs/1511.08458)
20. **Alex Sherstinsky.** Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network. [arXiv:1808.03314](https://arxiv.org/abs/1808.03314)
21. **Ralf C. Staudemeyer, Eric Rothstein Morris.** Understanding LSTM -- a tutorial into Long Short-Term Memory Recurrent Neural Network. [arXiv:1909.09586](https://arxiv.org/abs/1909.09586)
22. **Evgeniou, Theodoros & Pontil, Massimiliano.** (2001). Support Vector Machines: Theory and Applications. 2049. 249-257. 10.1007/3-540-44673-7_12.
23. **Cances, L., Labbé, E. & Pellegrini, T.** Comparison of semi-supervised deep learning algorithms for audio classification. J AUDIO SPEECH MUSIC PROC. 2022, 23 (2022). [Online] 2023. [Cited: febrero 09, 2023] [Comparison of semi-supervised deep learning algorithms for ... - Springer](#)

- 24. Breebaart, Jeroen & McKinney, Martin.** (2004). Features for Audio Classification. 10.1007/978-94-017-0703-9. [Online] 2023. [Cited: marzo 01, 2023]
- 25. SALAZAR, Jordi.** Redes inalámbricas. [Online] 2023. [Cited: marzo 01, 2023]
- 26. nPerf.** Map of 3G / 4G / 5G coverage by Movistar Movil, Spain. [Online] 2023. [Cited: marzo 01, 2023] <https://www.nperf.com/es/map/ES/-/168910.Movistar-Movil/signal/?ll=38.839707613545144&lg=-4.2187500000000001&zoom=6>
- 27. TELKAMP, Thomas.** LoRa World Record. [Online] 2023. [Cited: marzo 01, 2023] <https://www.thethingsnetwork.org/article/lorawan-world-record-broken-twice-in-single-experiment-1>
- 28. Zhang, Chenlu & Dang, Hua & Xiong, Yifeng & Yan, Tianqi.** (2019). Spread spectrum algorithm resistance to wideband non-stationary interference. The Journal of Engineering. 2019. 10.1049/joe.2019.0714. [Online] 2023. [Cited: marzo 01, 2023]
- 29. Dragino Technology.** LoRa GPS HAT. [Online] 2023. [Cited: marzo 01, 2023] <https://www.senzary.com/es/shop/dragino-technology-lora-gps-hat/>
- 30. Adafruit RFM96W.** LoRa Radio Transceiver Breakout 433MHz – RadioFruit. [Cited: marzo 01, 2023] [Adafruit RFM96W LoRa Radio Transceiver Breakout - 433 MHz](#)
- 31. Venco – Electrónica Industrial.** Qué es ZigBee, cómo funciona y características principales. [Online] 2023. [Cited: marzo 07, 2023] <https://www.vencoel.com/que-es-zigbee-como-funciona-y-caracteristicas-principales/>
- 32. Digi XBee.** Módulos inalámbricos, herramientas para desarrolladores y bibliotecas de software. [Cited: marzo 14, 2023] [Digi XBee Ecosistema | Todo lo que necesita para explorar y crear ...](#)
- 33. Texas Instruments.** CC2531 System-on-Chip Solution for IEEE 802.15.4 and ZigBee Applications datasheet. [Cited: marzo 14, 2023] <https://www.ti.com/product/CC2531>

- 34. E. M. Migabo, K. D. Djouani and A. M. Kurien,** "The Narrowband Internet of Things (NB-IoT) Resources Management Performance State of Art, Challenges, and Opportunities," in IEEE Access, vol. 8, pp. 97658-97675, 2020, doi: 10.1109/ACCESS.2020.2995938.
- 35. BricoGeek.** Shield LTE/GPS SIM7000E for Arduino. [Cited: marzo 14, 2023] <https://tienda.bricogeek.com/shields-arduino/1197-shield-lte-gprs-gps-sim7000e-para-arduino.html>
- 36. COPE, Luis.** How MQTT Works. [Online] 2023. [Cited: febrero 02, 2023.] [Beginners Guide To The MQTT Protocol - steves-internet-guide.com](https://steves-internet-guide.com/Beginners-Guide-To-The-MQTT-Protocol/)
- 37. T. Yokotani and Y. Sasaki,** "Comparison with HTTP and MQTT on required network resources for IoT," 2016 International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC), Bandung, Indonesia, 2016, pp. 1-6, doi: 10.1109/ICCEREC.2016.7814989.
- 38. F. Palmese, E. Longo, A. E. C. Redondi and M. Cesana,** "CoAP vs. MQTT-SN: Comparison and Performance Evaluation in Publish-Subscribe Environments," 2021 IEEE 7th World Forum on Internet of Things (WF-IoT), New Orleans, LA, USA, 2021, pp. 153-158, doi: 10.1109/WF-IoT51360.2021.9595725.
- 39. Victor Seoane, Carlos Garcia-Rubio, Florina Almenares, Celeste Campo.** Performance evaluation of CoAP and MQTT with security support for IoT environments, Computer Networks, Volume 197, 2021, 108338, ISSN 1389-1286, <https://doi.org/10.1016/j.comnet.2021.108338>.
- 40. Asana.** Cómo entender los procesos iterativos (con ejemplos). [Online] 2023. [Cited: febrero 02, 2023.] <https://asana.com/es/resources/iterative-process>
- 41. Python.** High level programming language. [Online] 2023. [Cited: febrero 02, 2023.] <https://www.python.org/about/>
- 42. Real Time Clock.** What is a Real Time clock (RTC)? [Online] 2023. [Cited: febrero 20, 2023.] <https://ecsxtal.com/what-is-a-real-time-clock-rtc/>
- 43. HIVEMQ.** MQTT Security Fundamentals. [Online] 2023. [Cited: febrero 20, 2023.] <https://www.hivemq.com/mqtt-security-fundamentals/>
- 44. JSON.** JavaScript Object Notation. [Online] 2023. [Cited: abril 25, 2023]. <https://www.json.org/json-en.html>

- 45. Callbacks.** "Callback function - MDN Web Docs Glossary ... - Mozilla Developer." [Cited: abril 25, 2023]. https://developer.mozilla.org/en-US/docs/Glossary/Callback_function.
- 46. STD 90 RFC 8259.** "Information on STD 90 » RFC Editor." [Cited: abril 25, 2023]. <https://www.rfc-editor.org/info/std90>.
- 47. Glen M. Ballou.** Polar diagrams. [Online] 2023. [Cited: marzo 21, 2023]
- 48. Micbooster.** Primo EM272Z1 Omni Electret Condenser Microphone. [Cited: febrero 20, 2023.] <https://micbooster.com/microphone-capsules/199-primo-em272.html>
- 49. Gil, Diego & Llusia, Diego. (2020).** The Bird Dawn Chorus Revisited. 10.1007/978-3-030-39200-0_3. [Online] 2023. [Cited: marzo 25, 2023]
- 50. RSPB.** Why do birds sing at night? [Online] 2023. [Cited: marzo 21, 2023] <https://www.rspb.org.uk/birds-and-wildlife/wildlife-guides/birdwatching/bird-behaviour/why-do-birds-sing-at-night/>
- 51. UUGEAR.** Witty Pi 3 Mini datasheet. [Online] 2023. [Cited: marzo 28, 2023] <https://www.uugear.com/product/witty-pi-3-mini-realtime-clock-and-power-management-for-raspberry-pi/>
- 52. Academia Unimusica.** La profundidad de bits. [Online] 2023. [Cited: marzo 28, 2023] https://www.unimusica-peru.com/produccion_musical_profundidad_bits.htm#:~:text=La%20Profundidad%20de%20Bits%20es.puede%20tener%20al%20ser%20grabada.
- 53. Symon Haykin.** Communication Systems 4th Edition.
- 54. BCM43438.** Datasheet [Online] 2023. [Cited: marzo 28, 2023] <https://www.alldatasheet.com/datasheet-pdf/pdf/1018493/CYPRESS/BCM43438.html>
- 55. ScienceDirect.** Audio features – Zero Crossing Rate [Online] 2023. [Cited: abril 04, 2023] [https://www.sciencedirect.com/topics/engineering/zero-crossing-rate#:~:text=Zero%20crossing%20rate%20\(ZCR\)—,signs%20of%20the%20amplitude%20values](https://www.sciencedirect.com/topics/engineering/zero-crossing-rate#:~:text=Zero%20crossing%20rate%20(ZCR)—,signs%20of%20the%20amplitude%20values).

- 56. Han, Ng & Muniandy, Sithi V & Dayou, Jedol & Ho, chong mun & Ahmad, Abdul & Dalimin, Noh.** (2010). Information Theoretic Approach Based on Entropy for Classification of Bioacoustics Signals. American Institute of Physics Conference Proceedings. 1250. 31-34. 10.1063/1.3469668. [Online] 2023. [Cited: abril 04, 2023]
- 57. ScienceDirect.** Audio features – Spectral Centroid Rate [Online] 2023. [Cited: abril 07, 2023] [Spectral Centroid - an overview | ScienceDirect Topics](#)
- 58. Nahian Ibn Hasan.** Bird Species Classification And Acoustic Features Selection Based on Distributed Neural Network with Two Stage Windowing of Short-Term Features. [Cited: abril 07, 2023] <https://arxiv.org/abs/2201.00124>
- 59. ScienceDirect.** Audio features – Zero Crossing Rate [Online] 2023. [Cited: abril 07, 2023] [Zero Crossing Rate - an overview | ScienceDirect Topics](#)