uc3m | Universidad Carlos III de Madrid

University Degree in Computer Engineering

2019-2020

Bachelor Thesis

“Pet monitoring system using artificial intelligence”

Alejandro Marchán Álvarez

Tutor

Ángel García Olaya

Leganés 2020

ABSTRACT

The following bachelor thesis covers the development of a low cost dog monitoring system that detects barks using artificial intelligence.

The main motivation for the development of this system is to improve the bark detection method of existing systems, which is based on the use of a sound threshold, in order to reduce the number of false positive notifications.

The pet monitoring system is divided into two main components, the monitoring system, which is built of a Raspberry Pi with a microphone, camera and speaker attached, that runs the artificial intelligence model in order to detect barks in real time, and a mobile application, whose purpose is allowing the user to use the system features, such as displaying real-time video and audio captured by the Raspberry Pi, changing the system settings or checking the bark record.

In order to develop the software used in this project, different technologies have been studied and used. First of all, the Python programming language along with different libraries, such as Tensorflow or LibROSA have been used for creating a deep learning model capable of classifying sounds in real-time. Secondly, the mobile application has been developed using the Ionic Framework with Angular. Thirdly, in order to share real-time video and audio between the Raspberry Pi and the mobile application, the WebRTC protocol has been studied and implemented on both sides. Lastly, in order to deploy a server that shares the recorded data from the monitoring system to the mobile application securely, the HTTPS protocol has been used by means of a Python library called *sauth*.

Keywords

Artificial intelligence, deep learning, sound classification, mobile application, TensorFlow, Keras, WebRTC, Raspberry Pi, Ionic, Angular, Python

RESUMEN

El siguiente Trabajo de Fin de Grado cubre el desarrollo de un sistema de monitorización de perros de bajo coste que detecta ladridos empleando inteligencia artificial.

La principal motivación para el desarrollo de este sistema es mejorar el método de detección de ladridos de los sistemas existentes, el cual se basa en el uso de un umbral de sonido, con el fin de reducir el número de notificaciones erróneas.

El sistema de monitorización de mascotas se divide en dos componentes principales, el propio sistema de monitorización, que está compuesto por una Raspberry Pi con un micrófono, una cámara y un altavoz, el cual ejecuta el modelo de inteligencia artificial para detectar ladridos en tiempo real, y una aplicación móvil, cuyo propósito es permitir al usuario utilizar las funciones del sistema, entre las cuales se encuentran mostrar vídeo y audio capturado por la Raspberry Pi en tiempo real, cambiar la configuración del sistema o comprobar el registro de ladridos.

Para desarrollar el software de este proyecto se han estudiado y utilizado diferentes tecnologías. En primer lugar, se ha empleado el lenguaje de programación Python junto con diferentes bibliotecas, como Tensorflow o LibROSA, para crear un modelo de aprendizaje profundo capaz de clasificar sonidos en tiempo real. En segundo lugar, la aplicación móvil se ha desarrollado utilizando el Framework de Ionic junto con Angular. En tercer lugar, para compartir vídeo y audio en tiempo real entre la Raspberry Pi y la aplicación móvil, se ha estudiado e implementado el protocolo WebRTC en ambas partes. Por último, para desplegar un servidor que comparta los datos registrados mediante el sistema de monitorización a la aplicación móvil de forma segura, se ha empleado el protocolo HTTPS por medio de la librería de Python *sauth*.

Palabras clave

Inteligencia artificial, aprendizaje profundo, clasificación de sonido, aplicación móvil, TensorFlow, Keras, WebRTC, Raspberry Pi, Ionic, Angular, Python

ACKNOWLEDGEMENTS

First, I would like to thank my family for supporting me throughout my whole educational life, by trusting, supporting and helping me even in the toughest moments.

I would also like to thank my friends, without them this road would have been much harder and longer. As well as my coworkers at Dominion, from which I have learnt many of the things embodied in this document.

Lastly, I would like to give special thanks to my tutor, Ángel, for guiding and supporting me along the process of making this project and for providing the idea on which it is based.

INDEX

FIGURE INDEX

TABLE INDEX

1. INTRODUCTION

This section provides an explanation about the reasons that led to the development of a pet monitoring system using artificial intelligence, defines the main objectives that should be achieved by the end of the process and presents the different parts in which this document is divided.

1.1. Motivation

During the years, owning a pet has been more common all around the world. In fact, according to the study *"Informe Sectorial"* conducted by AMVAC in 2017 [1], 39.7% of Spain households own a pet, which represents 18.63 million homes in the country.

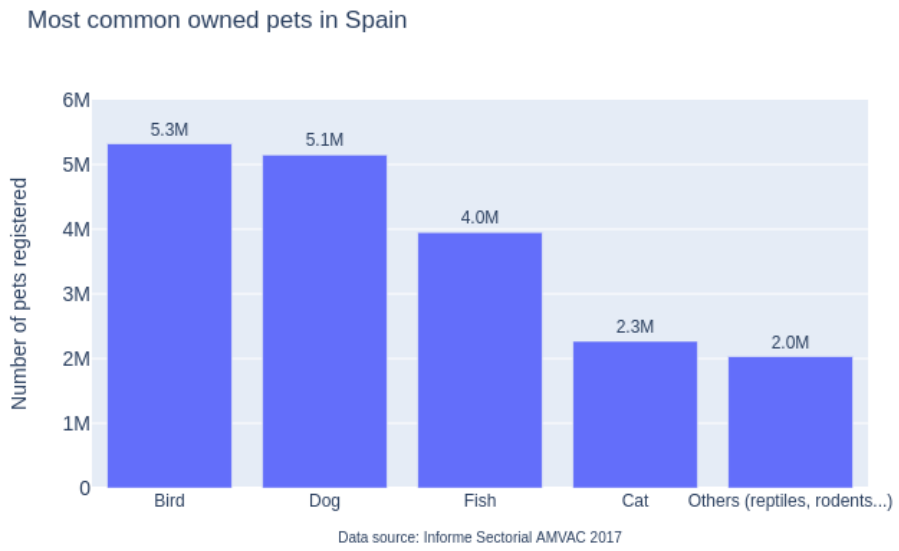

Fig. 1.1 "Most common owned pets in Spain in 2017" [1]

There are many different types of pet animals that people may have, but as shown in *Fig 1.1*, one of the most common is usually dogs. According to a research conducted in the U.K., the majority of dogs (53%) are left alone for 4 or more hours per day [2], during this time many owners have felt the need to be able to know how their pets are and what they are doing, as many of them bother the neighbors, get separation anxiety or something happens that makes them feel stressed.

In order to solve this problem, during the last decade, many solutions have arised that allow owners to monitor their dog, mainly via a mobile application connected to a security camera or to another device. These applications usually have the functionality to send a notification to the user when a bark is detected, in order to do so, they use a sound threshold that when it is surpassed sends a notification.

This approach, as expected, sends many false positive barking notifications to the user, as no further comprobations are made to ensure that the sound heard was a bark and not anything else.

Keeping this in mind, the main motivation of the following bachelor thesis is to solve the problem of detecting dog barks by using artificial intelligence in order to improve systems that monitor pet dogs.

1.2. Objectives

The main objective of this project is to develop a system, composed of a mobile application and a remote monitoring device, to monitor pet dogs that detects barks by using artificial intelligence while keeping it low cost. In order to accomplish this objective the project can be divided in four different subgoals:

1. Choosing suitable technologies in order to make the system inexpensive while meeting the requirements established.

2. Developing an artificial intelligence model capable of recognising bark sounds in real time from a portable device.

3. Implementing a protocol that allows two devices to send video and audio to each other in real time.

4. Developing an application that registers when a bark is detected and allows the user to program different actions.

To achieve these subgoals, an exhaustive research in artificial intelligence, real time communication and mobile application development will be carried out, in order to introduce these subjects, analyse the tools used to work with them as well as understand how to use them.

1.3. Document structure

The document is divided in eight main chapters, each of them with a different purpose, with the final objective of explaining the context and steps followed to develop the project.

- **Introduction:** this chapter explains the main reasons that led to the development of the project at issue as well as defines the main objectives that should be achieved by the end of the process. Lastly, a detailed explanation of the document structure is given.

- **Background:** this chapter explains the different technologies that are used nowadays in order to achieve the different parts in which this project can be divided, such as sound classification, real time communication or app development.

- **Related work:** in this chapter a study about the existing similar systems in the market is carried out.

- **Analysis:** this chapter provides a description of the system along with its capabilities, the design alternatives that could have been chosen and the legislation that must be taken into account. Also, the use cases and the requirements are defined.

- **Design and implementation:** in this chapter, each of the modules in which the system can be divided are described in depth as well as how the chosen technologies to implement each of them are used.

- **Testing:** this chapter collects all the tests that have been carried out to ensure that the system fulfils the requirements defined.

- **Project management:** this chapter describes the schedule followed to implement the following project, the estimated budget required and the socio-economic environment in which it is presented.

- **Conclusions:** this is the final chapter of the bachelor thesis, its purpose is to revise the achievement of the initial objectives as well as providing recommendations for future work.

Besides the previous chapters already described, at the end of the document there is a list of all the consulted resources for this bachelor thesis and two appendices in which the user manual and the system installation manual can be found.

2. BACKGROUND

In the following chapter of the document an in depth explanation as well as an analysis of the key components of this thesis is conducted, also the main technologies used nowadays in each of them as well as the ones chosen for the development of the project are described.

2.1 Artificial intelligence

Artificial intelligence (AI) is a branch in computer science defined as the ability of computer or computer-controlled robots to perform tasks commonly associated with intelligent beings [3], some of these tasks are reasoning, planning, learning or recognition among others. As the field of artificial intelligence is very large, it is divided in many sub-fields based on technical considerations, different goals or the use of particular tools [4]. The sub-field of AI in which this section will be focused is machine learning.

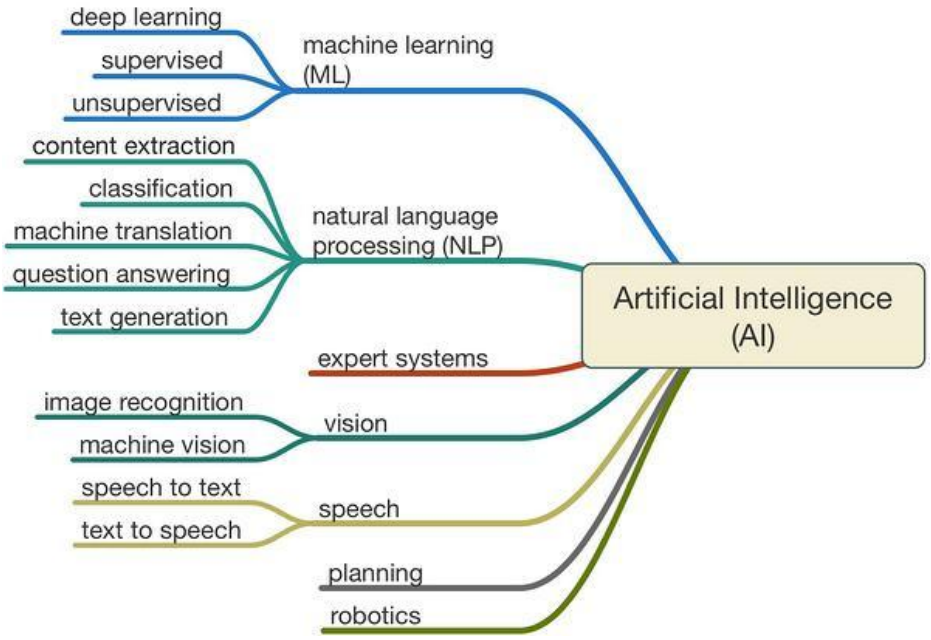

Fig. 2.1 "Artificial intelligence sub-fields" [5]

2.1.1 Machine learning

Machine learning is the study of computer algorithms that improve automatically through experience, in other words, involves computers discovering how they can perform tasks without being explicitly programmed to do so [6]. In order to achieve this, machine learning is commonly divided in two main different types of techniques [7][8][9][10][11]:

- **Supervised learning**: these techniques take a set of input-output pairs and analyze it to produce a function that generates accurate predictions for new input data. In order

to develop these predictive models, supervised learning techniques are divided depending on the type of the desired output:

- **Classification:** they predict discrete output, this kind of models distribute input data into different pre-established categories. One example problem of this kind of task is deciding whether an email is genuine or spam.

- **Regression:** they predict continuous output, instead of dividing data into different categories, these techniques give non pre-established, computed values to a given input. One example problem of this kind of tasks is predicting currency fluctuation in time.

- **Unsupervised learning:** these techniques, in contrast to the supervised learning techniques receive non labeled data and try to find hidden patterns and intrinsic structures in it. One of the most common unsupervised learning techniques is clustering:

  - **Clustering:** it is used to find patterns and groupings in data so that elements in the same group are more similar to each other than to the ones in other groups. One example of this kind of tasks is discerning between different types of tissue in a three-dimensional image.

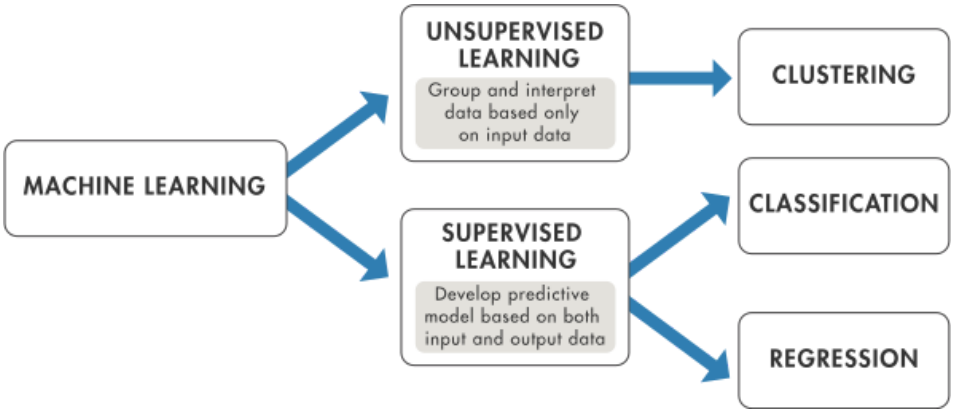

Fig. 2.2 "Machine learning simple taxonomy" [7]

2.1.1.1 Artificial neural networks

There are many different methods to implement the techniques mentioned above, one of the most common due to its capabilities is artificial neural networks (ANN), these systems are collections of connected nodes known as artificial neurons, each neuron can transmit or receive signals from other neurons. The "signal" at a connection is a real number, and the output of each neuron is computed by a function (which in most cases is non-linear), called *activation function*, applied to the sum of its inputs. Both these connections and the neurons typically have weights that get modified during the learning process in order to give or reduce

importance of the signal in them. In ANNs, neurons are usually aggregated into layers, which may have different activation functions and number of neurons. Finally, generally neurons in contiguous layers are connected between them and signals travel from the first layer, known as the *input layer*, to the last layer, known as the *output layer*. The rest of the layers between the input and the output layers are known as *hidden layers* [12][13].

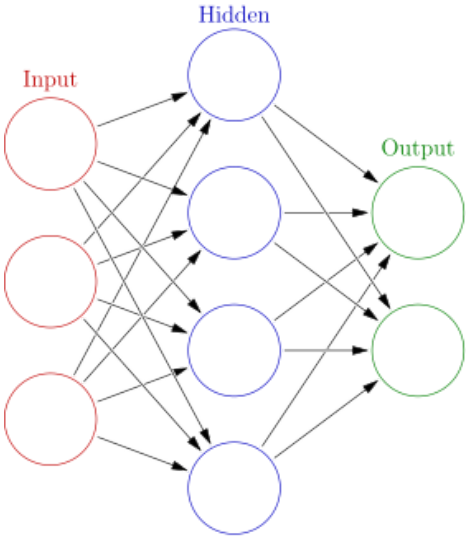

Fig. 2.3 "Basic artificial neural network architecture" [14]

2.1.1.2 Deep learning

In addition to the basic architecture of ANNs, there are many other types, one of them being deep neural networks (DNN). The main difference with the basic architecture with only one hidden layer is that DNNs have multiple hidden layers, usually ANNs with only one hidden layer are called "*shallow*", hence ones with more are called "*deep*". Having more layers allows this kind of ANNs to model more complex data such as images or sound more accurately and efficiently, as each layer adds a new level of abstraction to the features extracted in the previous one. This process is called "*hierarchical feature learning*". Using this kind of more complex ANNs to implement the techniques listed above is called ***deep learning*** [15][16].

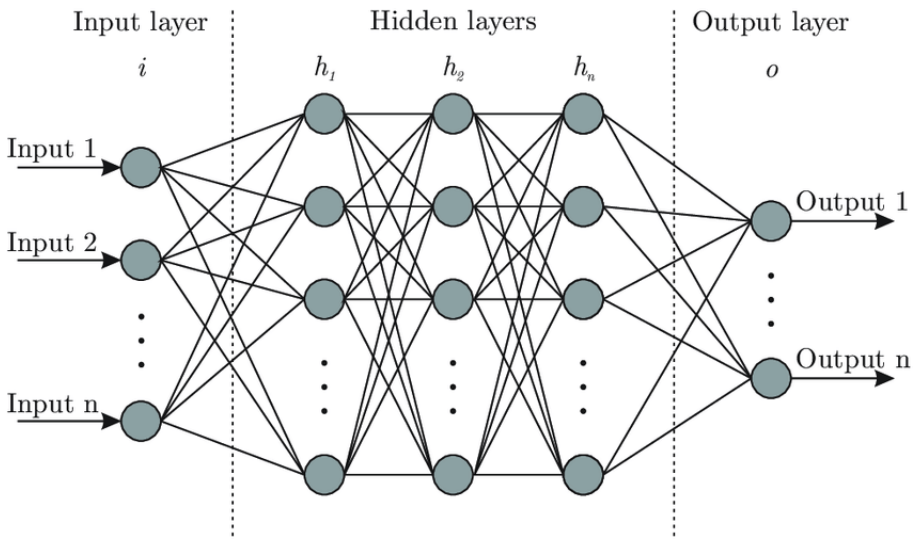

Fig. 2.4 "Deep neural network architecture" [17]

2.1.2 Deep learning applied to sound classification

As stated in the previous section, deep learning architectures are very powerful state of the art tools that can be used to work more efficiently with complex data. The process of solving a problem using machine learning, which includes deep learning as shown in *Fig 2.1*, has been described as follows by the Google Cloud developer Yufeng Guo in his article "*The 7 Steps of Machine Learning*" [18]:

1. **Gathering data:** the amount and quality of the data directly impacts the quality of the results obtained by the model, usually, the more data you have the better [19].

2. **Data preparation:** when using deep learning to classify sound, one of the most important tasks is extracting the sound features in a way that they are meaningful and can be used as an input to the model, in order to achieve this, some commonly used sound conversions are Mel-frequency cepstral coefficients (MFCCs), Chroma Features or Mel Spectrogram [20][21][22]. Once these features are obtained, the data is randomized and splitted in three parts, which are called the training, evaluation and test set. The first set has the most of the data which is used to train the model and the other two are used for validating and testing the trained model.

3. **Choosing a model:** there are many different deep learning models that can be used depending on the task at issue, for sound some common ones are deep neural networks, convolutional neural networks (CNN) or recurrent neural networks (RNN) [23][24].

4. **Training:** in this step the training set is used as the input to the model, in each iteration of the process of learning, the model tries to predict the output and compares the computed results with the right ones, then the model updates the weights

associated to each connection and neuron in order to improve the results for the next iteration. This process repeats until some condition is reached, usually a maximum number of iterations.

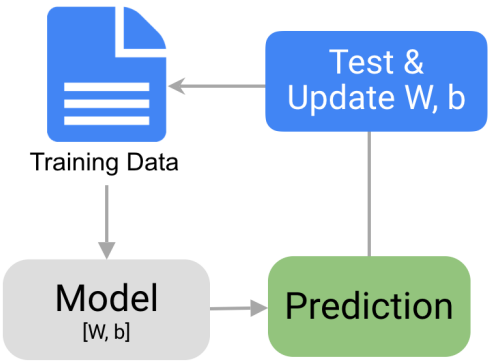

Fig. 2.5 "Model training process schema" [17]

5. **Evaluation:** in this step the evaluation and test sets are used, the main goal is to check how does the model respond to new data, in order to determine the quality of the model. Some common problems in this step are called "*overfitting*" and "*underfitting*". These terms refer, in the first case, to the model adjusting to the training data too well that it has not generalised knowledge so it can't perform well with new data, and in the second case, to the model not being able to predict well neither training or testing data [25].

6. **Hyperparameter tuning:** deep learning models have many different parameters that can be modified in order to achieve different results, such as "*learning rate*", number of iterations, number of hidden layers, number of neurons per layer, etc. But as Yufeng Guo states "*The adjustment, or tuning, of these hyperparameters, remains a bit of an art*" as there are not clear rules as to which are the best hyperparameters to use because they are heavily dependent on the dataset, model and training process.

7. **Prediction:** finally, the model can be used to compute the output of new data by extracting its features and feeding them to the obtained model.

2.1.3 Most popular programming languages for machine learning

In order to implement the machine learning models and techniques explained in the previous sections there are many different alternatives when it comes to choosing a programming language, according to the latest survey conducted by Kaggle, an online community of data scientists and machine learning developers, in which 23,859 users took part, the top 3 most used programming languages in this field turned out to be Python, SQL and R in that order [26].

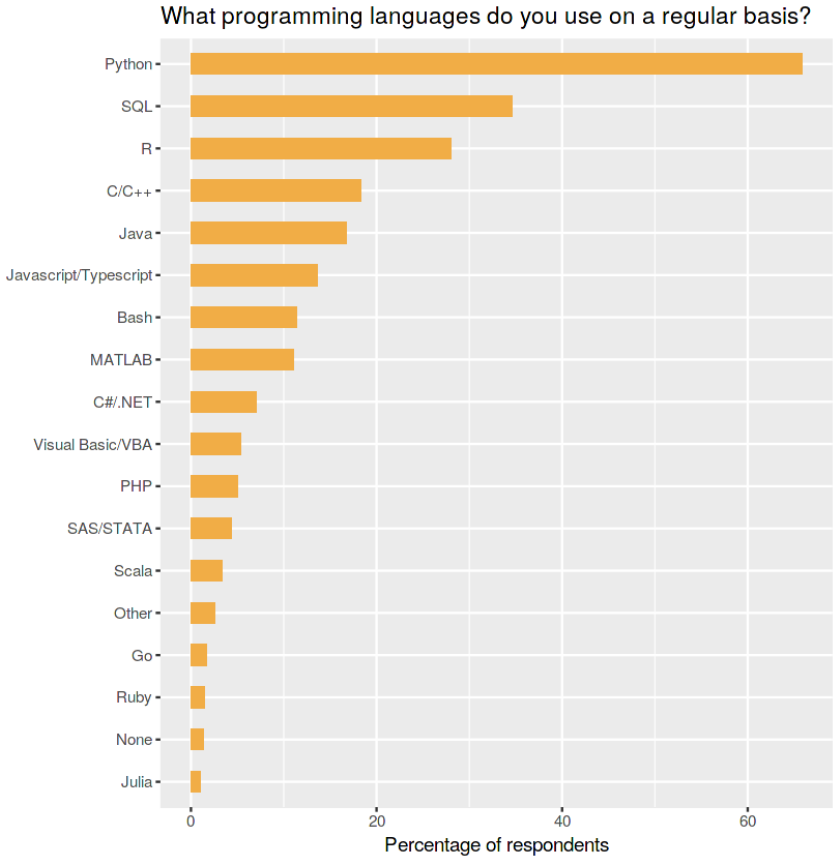

Fig. 2.6 "Most common programming languages according to 2018 Kaggle ML & DS Survey" [27]

Python is an interpreted, high-level, general-purpose programming language that was first released in 1991 and has more than 8.2 million developers worldwide according to SlashData [28][29]. Furthermore, Python counts with a well maintained documentation, a very active community in sites such as Stack Overflow, created to ask questions about programming problems, and according to The Python Package Index (PyPI) more than 237,172 available packages developed by the community for multiple different purposes [30][31].

2.1.4 Python packages

In the field of machine learning, one of the big advantages that provides Python, as stated by Angela Beklemysheva in her article "*Why Use Python for AI and Machine Learning?*" [32], in addition to its simplicity and consistency is the extensive selection of libraries and frameworks that it has related to the field. Python offers well maintained and documented packages, which reduce the time and effort of tackling machine learning tasks from scratch. Among those packages, the most common are TensorFlow, Keras, Pytorch and Scikit-learn.

2.1.4.1 TensorFlow

TensorFlow is an end-to-end, open source library focused on machine learning tasks. It was originally developed by the Google Brain team before being published under open source

licensing in november 2015 [33]. TensorFlow is mainly focused for more experienced machine learning developers, thus is commonly used via some high level APIs in many cases.

2.1.4.2 Keras

Keras is a high level deep learning API written in Python that runs on top of TensorFlow. Its main focus is enabling easier and faster implementation of deep learning models, as well as allowing the user to export them and use them in the browser or on a mobile phone [34].

2.1.4.3 LibROSA

LibROSA is a python package for audio analysis, it provides the necessary tools for sound feature extraction as well as audio modification [35].

2.1.4.4 sounddevice

Sounddevice is a Python module that provides tools for playing and recording audio directly from a Python program [36].

2.2 Raspberry Pi

Raspberry Pi is a well known brand of single-board computers (SBC), which refer to a complete computer built on a single circuit board.

Raspberry Pi was developed by the Raspberry Pi Foundation in February 2012. Due to its low cost, portability and specifications it is widely used in projects of many different kinds, such as robotics, web servers or education.

Since the original model, many different versions have been developed, but in this case the main focus will be the Raspberry Pi 3 model B+ as is the one that will be used in this project [37][38].

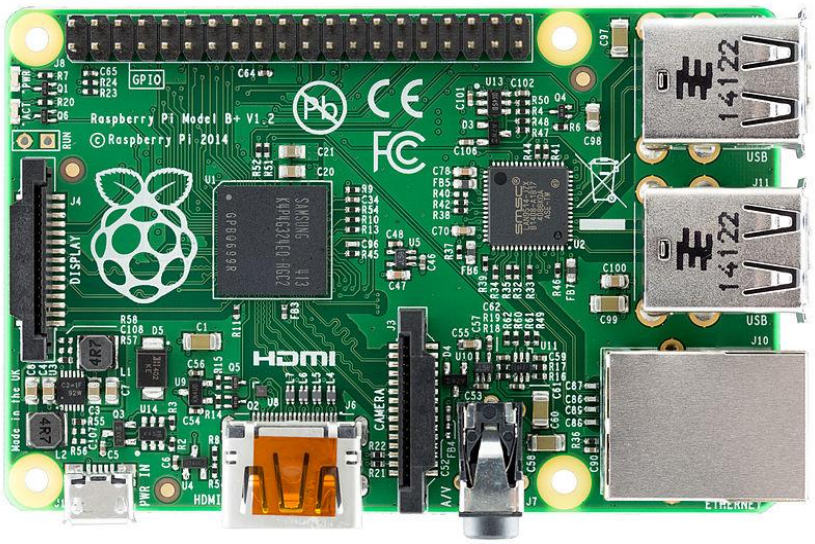

Fig. 2.7 Raspberry Pi 3 model B+ top view [39]

The circuit board of the Raspberry Pi 3 model B+ is 85 mm x 56 mm [40]. The hardware can be divided in 6 different components as shown in the following *Fig. 2.8*.

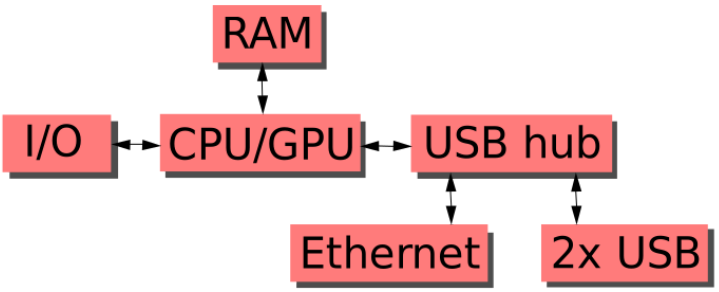

Fig. 2.8 Raspberry Pi 3 model B+ hardware block diagram [41]

For the project at hand, the most relevant features in the hardware side are related to performance, input/output and connectivity.

- **Performance:** in this sense, the Raspberry Pi model B+ has limited processing power, and even though it is capable of performing inference using a trained machine learning model, this process will be slow [42].

- **Input/Output:** in addition to the different USB 2.0 ports and 40-pin GPIO headers, Raspberry Pi models count with a camera port for which the company offers a 5MP HD official camera that costs around 14 euros [43][44].

- **Connectivity:** in relation to connectivity, this model counts with Bluetooth 4.2 as well as with wireless and cable internet connection [44].

When looking at the software and OS of the Raspberry Pi, the recommended one by the company is called Raspberry Pi OS, which was previously called Raspbian. This OS is Debian-based and optimized for the Raspberry Pi hardware, this means that as Debian is a Linux distribution, many of the Linux packages are available and compatible with it [45].

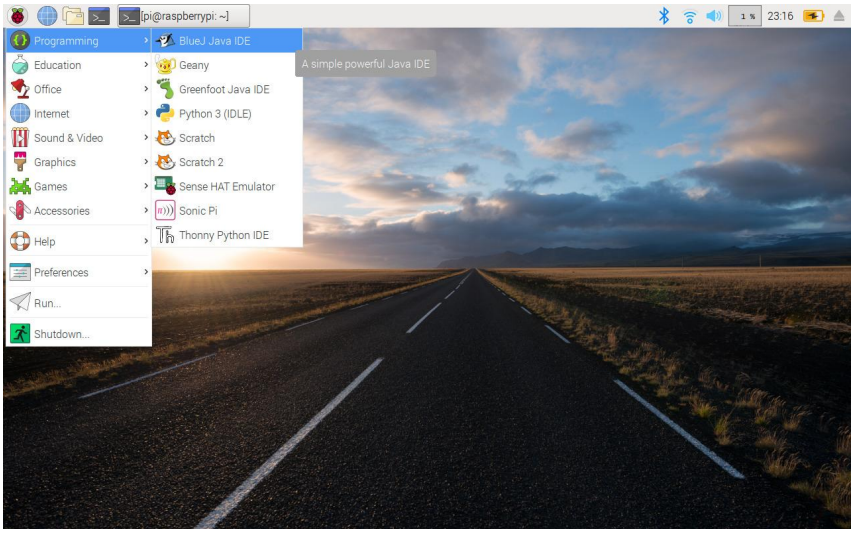

Fig. 2.9 Raspberry Pi OS screenshot [46]

2.3 Real time communication

Real time communications (RTC) are any type of telecommunications that happen live and without transmission delays or latency. There are two modes RTC can take place:

- Half-duplex: allows data to travel bidirectionally on a single carrier but not at the same time.

- Full-duplex: allows data to travel bidirectionally on a single carrier and at the same time [47].

The main difference between RTC and other types of communications is that rather than focusing on ensuring that all the data arrived well and in the right order despite making the process slower, it focuses on sending the data as fast as possible without caring much about some data being lost in the process.

There are many examples of RTC uses, but in this case our main focus will be in web RTC for video and audio sharing.

2.3.1 WebRTC

As defined by the official WebRTC page, "*WebRTC is a free, open project that provides browsers and mobile applications with Real-Time Communications (RTC) capabilities via simple application programming interfaces (APIs)*" [48].

WebRTC APIs are programmed in JavaScript and have different purposes, the one that is most related to this project is called *RTCPeerConnection*, which allows streaming audio and video between users, also called *peers*, as the protocol it uses is peer-to-peer (P2P), which refers to a connection between users without intermediaries such as servers or stable hosts.

WebRTC is currently being standardized by the World Wide Web Consortium (W3C) and the Internet Engineering Task Force (IETF) in order to make this technology easier to use in different web platforms. For the moment, WebRTC is available in most of the commonly used browsers and is being actively developed [49][50][51].

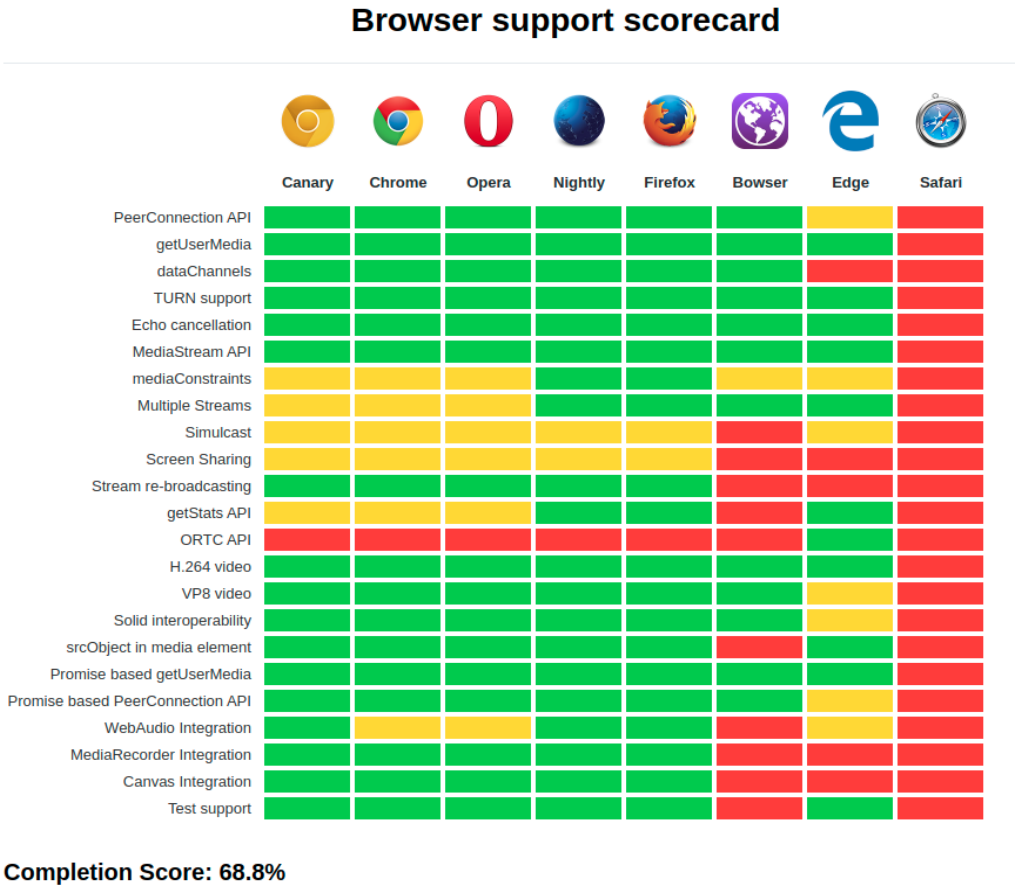

Fig. 2.10 WebRTC browser support scoreboard [51]

2.4 Mobile application development tools

During the last years, the number of mobile applications has increased considerably. Along with it, many development platforms have arised or improved their capabilities in order to make the process of creating mobile applications easier and faster.

One of the main problems that existed with many of these tools was cross-platform development, which refers to developing the same mobile application for different operating systems (OS) such as Android, iOS or Windows, as this process was different for each of them and as shown in *Fig. 2.11* choosing only one operating system would mean reducing the number of possible users by at least a 10%.

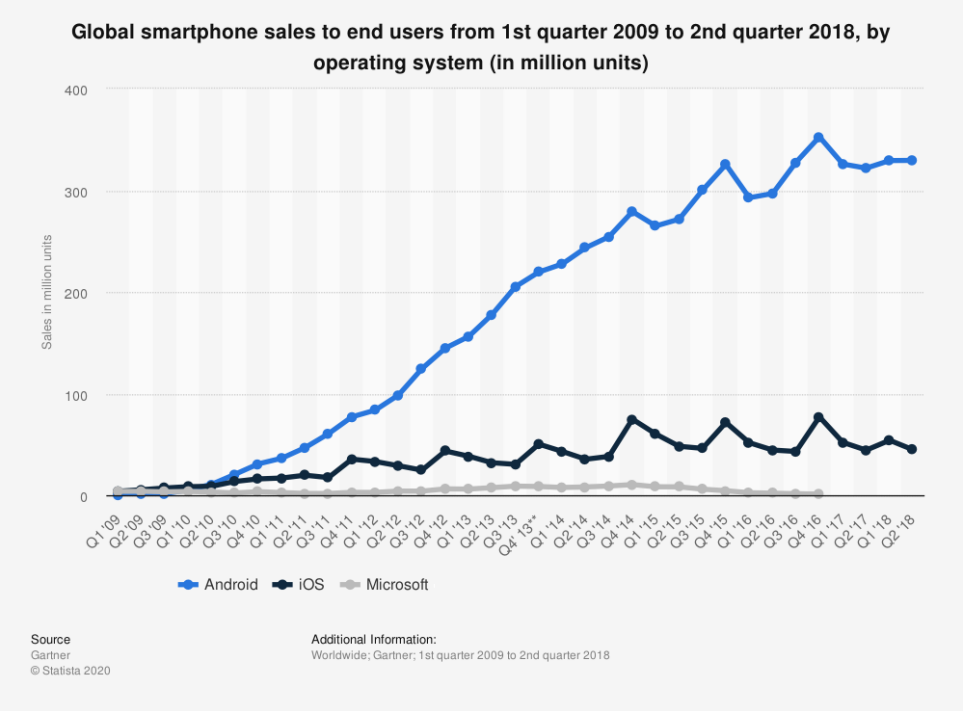

Fig. 2.11 Number of apps available in leading app stores as of 1st quarter 2020 [52]

This problem was solved with cross-platform development tools, which allow creating a mobile application only once and then translate it to the desired OS among the ones supported when compiled.

There are many different cross-platform development tools such as React Native, Xamarin, Flutter, Ionic, etc. In this case we will be focusing in Ionic, as is the one chosen for this project.

2.4.1 Ionic

Ionic is an open source framework that allows developing cross-platform mobile and desktop applications by using web technologies such as HTML, JavaScript or CSS. It integrates with

other frameworks such as Angular, React or Vue and it is mainly focused on the UX and UI interaction [53].

2.4.1 React Native

React Native is an open source framework that allows developing cross-platform mobile and desktop applications by using native platform capabilities.

React Native was created by Facebook in 2012 in order to improve performance of their apps in different platforms [54].

3. RELATED WORK

In this chapter, the current solutions that exist to solve the problem previously mentioned will be presented and discussed.

There are many different approaches that have been used to create this kind of systems, but the main ones could be divided in two, which main difference lies in the monitoring side.

3.1 Systems based on a monitoring device with a mobile application

This kind of systems, as stated in the title, are based on a monitoring device which has a mobile application that allows the user to connect to it and carry out different actions depending on the system. Some common features are connecting to a camera in real-time, interacting with the pet in different ways, such as giving it a treat, talking to it or playing with it, mainly by using an attached laser.

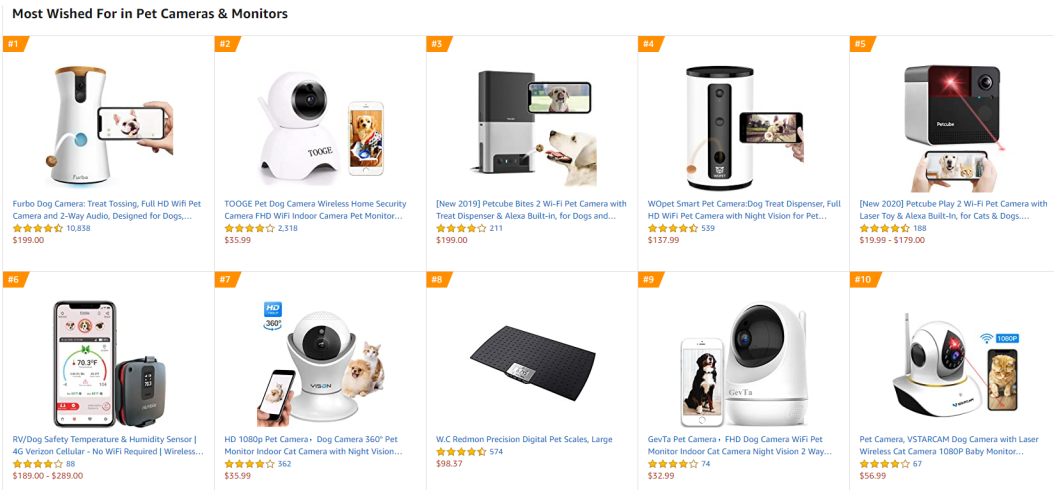

Fig. 2.12 Top 10 wished pet cameras & monitors in Amazon [55]

As shown in *Fig. 2.12*, the cost of the most sold systems of this kind in Amazon vary noticeably, ranging from 33 $ to around 200 $. This difference in price can be explained due to the capabilities of the system, as the most expensive ones have more available features, being the treat dispenser their main differentiator.

From the notification side, most of these systems have a motion detection sensor in order to notify the user about the pet's activity. Some of them also have a sound detector that notifies the user when a sound surpases certain threshold.

To sum up, these systems have the advantage of having many different capabilities focused on monitoring the pet as they are a dedicated system, but as these capabilities increase so does the price, making some of these systems very expensive.

3.2 Systems based on a mobile application used in two devices

This kind of systems are based on a mobile application that has to be installed in two different devices so that one of the devices is the one that the user usually carries and the other one acts as a monitoring camera that allows the user to monitor the pet.

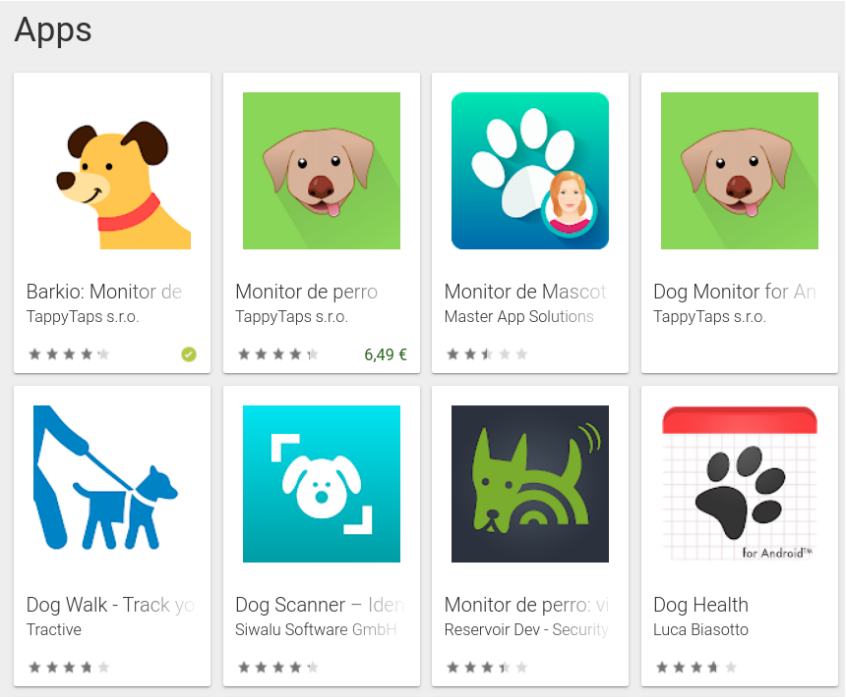

Fig. 2.13 Top dog monitoring apps in Google Play [56]

These apps are usually free or have a low cost, among the most famous ones in Google Play as shown in *Fig. 2.13* there is Barkio or Dog monitor app. The main features of this kind of apps are very similar, being these, among others, the possibility to watch live video of the dog, speak to the dog and receive a notification when a sound is detected.

The main disadvantage of this kind of system is the need of two devices in order to use them, since these devices are not made for this kind of prolonged usage, they run into different problems such as low battery life, low quality video or bad sound detection.

To sum up, these systems have the advantage of having a low cost from the mobile application point of view, but as two devices are needed in order to use them, this price can increase considerably without guaranteeing important technical specifications such as high battery life, high quality video or good sound detection.

3.3 Comparison with the developed system

In this section, a comparison between the different systems commented previously and the one developed in this thesis will be carried out.

In order to make this comparison, the most sold product in Amazon for the first kind of system, the Furbo dog camera, and the most downloaded app in Google Play for the second, Barkio, will be chosen. In the following *Table 2.1*, both these products' most relevant characteristics will be compared to the system developed in the project [57][58][59].

	Furbo dog camera	Barkio	Developed system
Price	169 €	Free, the price comes from the device used to monitor the dog	70 €
Features	- Treat tossing. - Barking alert using sound threshold. - Real-time video and audio. - Talk to the dog. - Movement detection.	- Barking alert using sound threshold. - Real-time video and audio. - Talk to the dog. - Available for multiple devices.	- Barking alert using AI recognition system. - Real-time video and audio. - Autoplay music when a bark is detected.
Camera specifications	1080p Full HD, 4x zoom, 160° wide-angle and night vision	Depends on the device used, commonly around 72º field of view and Full HD.	1080p Full HD and 54° field of view

TABLE. 2.1 SIMILAR SYSTEMS COMPARISON

When looking at the comparison, it is clear that the system developed is more similar to the Furbo, but cheaper and, thus, with less features. Also, the main differentiator between the developed system and the other two is the usage of AI in order to detect barks more accurately. Lastly, the developed system does not have the feature to talk to the dog, but in exchange the feature of playing music when a bark is detected is added.

4. ANALYSIS

In this chapter, a description of the project and its capabilities are given. Also, the use cases of the system as well as the requirements are presented. Lastly, a brief study of the legislation related to the project is conducted.

4.1 Description

As stated before, the purpose of this thesis is to develop a dog monitoring system, this system is divided into a mobile application and a monitoring device, which in this case is a Raspberry Pi with a camera, a microphone and a speaker attached.

The system detects dog barks and differentiates them from other sounds in the house by using a deep learning model. Sounds are recorded in real time via the microphone attached to the Raspberry Pi and analysed by the model in order to determine whether the recorded sound is a bark or not.

Every time a bark is detected, information about the time when it was recorded and the duration of continuous barking is saved in the Raspberry Pi, also if the user has an action programmed when a bark is detected, the action is carried out.

When using the app, there are three different functionalities the user can use:

1. Check a record of the barks registered by the system along with the duration, time and date of each of them.

2. Connect to the camera and microphone attached to the Raspberry Pi in real time.

3. Set an action when a bark is detected. In this case, the only available action implemented is playing music.

The focus of the system is not to be a commercial product but more of a DIY engineering project, even though by simplifying the Raspberry Pi installation and adding more available actions, it could be converted into one.

4.2 Design alternatives

The design of the system developed in the thesis is divided into a mobile application and a monitoring device. In this case, the monitoring device also acts as a server to which the app connects in order to retrieve the obtained data.

An alternative design to this one could be having a separated server to which the app and the monitoring system connect in order to share data. But as one of the goals of the system is to keep a low cost, running a server on another machine would increase the cost of development considerably.

Another change that could be made to the project is choosing a different technology for the monitoring side. The one used in this project is a Raspberry Pi as explained previously and the reason why this option was better than others, like another mobile device, is the processing power as the monitoring side has to be constantly running an AI model that analyses recorded sounds while sharing real-time and non-real-time data. Furthermore, the monitoring system is designed to be static, and kept in a house for long periods of time, which reduces the need of making it portable.

4.3 Capabilities

The system will have the following capabilities:

- Detect barks in real time.
- Displaying a record of the time, date and duration of the barks detected.
- Filter barks from the record by time, date or duration ranges.
- Stream video and audio from the Raspberry Pi to the app.
- Upload a song from the app to the Raspberry Pi.
- Autoplay music when a bark is detected.
- Multiple device access to the bark record.

4.4 Use cases

In this section the different use cases of the system are presented. Every use case is presented using the table shown in *TABLE 4.1*.

UC-X	
Name	
Actor	
Objective	

Flow	
Alternative Flow 1	
...	
Alternative Flow N	

TABLE. 4.1 TEMPLATE TABLE FOR USE CASES

Each element of the previous table has the following meaning:

- **Identifier**: each use case has a unique identifier constituted by the letters "*UC-*" followed by a number that represents an enumeration starting from 1.
- **Name**: descriptive name of the use case.
- **Actor**: the person or element that interacts with the system. In this case, the main actor is the user of the application.
- **Objective**: refers to the goal that the actor is trying to accomplish.
- **Flow**: is the sequence of steps the actor should follow to fulfill the objective of the use case. It represents the basic flow where nothing goes wrong.
- **Alternative flow**: there can be multiple alternative flows. They represent the alternative steps when something goes wrong.

In every use case where the actor is the user, in order to avoid redundancy, the first steps of the flow in all of them are specified now:

1. Download and install the mobile application developed in this thesis.
2. Open the mobile application.

Also, it is assumed that the Raspberry Pi with its corresponding software is also installed and ready to use.

UC-1	
Name	Connect to Raspberry Pi
Actor	User
Objective	Connect to the Raspberry Pi in order to interact with it using the app.

Flow	1. Open the settings menu 2. Fulfil the required fields 3. Press the save button
Alternative Flow 1	If it is the first time that the user opens the app the fields from the settings menu will appear straight away.
Alternative Flow 2	If the save button is not pressed and the app closes the changes made will be lost.
Alternative Flow 3	If one of the required fields is blank the app will not connect to the Raspberry Pi.

TABLE. 4.2 USE CASE 1

UC-2	
Name	Connect to camera
Actor	User
Objective	Watch in real time the video and audio captured by the camera and microphone attached to the Raspberry Pi
Flow	1. Press the camera button in the app 2. Watch real time video and audio from the camera and microphone
Alternative Flow 1	If the app is not connected to the Raspberry Pi an error will be shown and no video or audio will be displayed.
Alternative Flow 2	If the camera is being used by another device, a message is shown to warn the user and no video is displayed.
Alternative Flow 3	If no microphone is attached, the video is shown without audio.

TABLE. 4.3 USE CASE 2

UC-3	
Name	Check the record
Actor	User
Objective	Check all the registered barks with the

	corresponding information about them.
Flow	1. Press the record button 2. A list with all the recorded barks is shown
Alternative Flow 1	If the app is not connected to the Raspberry Pi an error will be shown and no data will be displayed.

TABLE. 4.4 USE CASE 3

UC-4	
Name	Search in the record
Actor	User
Objective	Search for a bark in the record depending on the time, date and duration.
Flow	1. Press the record button 2. Press the filter button 3. Modify the ranges of the filters 4. Press the “OK” button 5. A list of the filtered barks is shown
Alternative Flow 1	If the app is not connected to the Raspberry Pi an error will be shown and no data will be displayed.
Alternative Flow 2	If the “OK” button is not pressed the filter will not be applied.

TABLE. 4.5 USE CASE 4

UC-5	
Name	Upload a song
Actor	User
Objective	Upload the song to be played when a bark is detected.
Flow	1. Open the settings menu 2. Press the “Upload a song” button 3. Choose a song from the device 4. Press the “OK” button

Alternative Flow 1	If the app is not connected to the Raspberry Pi an error will be shown and the song will not be saved.
Alternative Flow 2	If the "OK" button is not pressed the song will not be saved.

TABLE. 4.6 USE CASE 5

UC-6	
Name	Activate/Deactivate the autoplay option
Actor	User
Objective	Activate/Deactivate the option to autoplay music when a bark is detected.
Flow	1. Open the settings menu 2. Press the toggle button "Autoplay music" 3. Press the save button
Alternative Flow 1	If the app is not connected to the Raspberry Pi an error will be shown and the data will not be saved.
Alternative Flow 2	If no song is uploaded to the Raspberry Pi, a warn message will appear to ask the user to upload a song before activating the autoplay option.
Alternative Flow 3	If the save button is not pressed and the app closes the changes made will be lost.

TABLE. 4.7 USE CASE 6

4.5 Requirements

In this section, the requirements that the system must fulfill in order to have the expected functionality are defined.

Each requirement is presented using the table shown in *TABLE 4.8*.

RXX-X	
Name	

Priority	
Description	

TABLE. 4.8 TEMPLATE TABLE FOR REQUIREMENTS

Each element of the previous table has the following meaning:

- **Identifier**: each requirement has a unique identifier constituted by the letter "*R*" followed by "*F*" if the requirement is functional or "*NF*" if it is non functional. After those letters, there is a hyphen followed by a number that represents an enumeration starting from 1.

- **Name**: descriptive name of the requirement.

- **Priority**: represents the relevance of the requirement in the development of the project and in the fulfillment of user expectations. Has three possible values:

    - **High**: the requirement is essential for the project development.
    - **Medium**: the requirement is desired for the project development.
    - **Low**: the requirement is optional for the project development.

- **Description**: brief description that summarizes the requirement.

4.5.1 Functional

RF-1	
Name	Detect barks
Priority	High
Description	The system must be able to detect barks in real time and record the time, date and duration of them.

TABLE. 4.9 FUNCTIONAL REQUIREMENT 1

RF-2	
Name	Connect to camera
Priority	High
Description	The system must be able to connect to the

	camera attached to the Raspberry Pi and display real-time video and audio from it.

TABLE. 4.10 FUNCTIONAL REQUIREMENT 2

RF-3	
Name	Autoplay music
Priority	High
Description	The system must be able to autoplay the uploaded song by the user when a bark is detected.

TABLE. 4.11 FUNCTIONAL REQUIREMENT 3

RF-4	
Name	Upload a song
Priority	High
Description	The system must allow the user to upload a song.

TABLE. 4.12 FUNCTIONAL REQUIREMENT 4

RF-5	
Name	Filter the record
Priority	Medium
Description	The system should allow the user to filter the barks record.

TABLE. 4.13 FUNCTIONAL REQUIREMENT 5

RF-6	
Name	Change the settings
Priority	High
Description	The system must allow the user to change

	the IP address of the Raspberry Pi, as well as the username and password of the server and WebRTC connection.

TABLE. 4.14 FUNCTIONAL REQUIREMENT 6

RF-7	
Name	App navigability
Priority	High
Description	The system must have a menu that allows the user to access all the pages of the app.

TABLE. 4.15 FUNCTIONAL REQUIREMENT 7

4.5.1 Non functional

RNF-1	
Name	IP address format
Priority	High
Description	The system will only allow IP addresses of the format “X.X.X.X” where each “X” is any number from 0-255

TABLE. 4.16 NON FUNCTIONAL REQUIREMENT 1

RNF-2	
Name	Song format
Priority	High
Description	The system will only allow the user to upload songs with “mp3” format

TABLE. 4.17 NON FUNCTIONAL REQUIREMENT 2

RNF-3	
Name	Internet access
Priority	High

Description	The system will require internet access when the app is being used.

TABLE. 4.18 NON FUNCTIONAL REQUIREMENT 3

RNF-4	
Name	Operating system compatibility
Priority	High
Description	The system will be available for any Android or iOS device.

TABLE. 4.19 NON FUNCTIONAL REQUIREMENT 4

RNF-5	
Name	Maximum request time
Priority	High
Description	The maximum answer time to a request to the server must be under 20 seconds.

TABLE. 4.20 NON FUNCTIONAL REQUIREMENT 5

RNF-6	
Name	Secure connections
Priority	High
Description	The system must use secure protocols for data transfer such as HTTPS.

TABLE. 4.21 NON FUNCTIONAL REQUIREMENT 6

RNF-7	
Name	Multiple connections
Priority	High
Description	The system must support at least 5

	connections to the server at the same time.

TABLE. 4.22 NON FUNCTIONAL REQUIREMENT 6

4.6 Legislation

When developing a system that could potentially be sold as a product, it is important to consider all the legal aspects established that relate to it and that the system must obey.

Generally, when talking about mobile applications, these restrictions come from the user's privacy side. In the European Union these restrictions are defined in the *General Data Protection Regulation (GDPR)* [60], which came into force in May 2016. Also, in Spain two other laws must be considered when developing an app, which are the *Organic Law on Personal Data Protection and guarantee of digital rights (LOPD)* [61] and the *Law of Services of the Information Society and Electronic Commerce (LSSI)* [62].

According to the GDPR, the LOPD and the LSSI, in relation with the project at hand, the following restrictions must be taken into account [63]:

- **User consent:** it is necessary to inform the user about the data that can be obtained through the app and the purpose of it, as well as, asking for permissions when a device functionality is used. In order to achieve this, when accessing the device storage for uploading a song, the user is asked to give permissions.

- **Data minimisation:** refers to obtaining just the necessary data from the user in order to make the system work properly. In this case, all critical data stored is the Raspberry Pi's IP and the credentials of the server in order to connect to it.

- **Data protection:** all personal data must be properly processed in order to ensure the protection of the integrity and confidentiality of the user in case of unauthorised or unlawful processing or accidental loss, destruction or damage. In the system developed, all data is transferred and stored using standardized encryption methods that ensure the security of it.

- **General information:** The identity of the person responsible for the app as well as information related to their contact details must be given to the user. In this case, the responsible for the app is the developer.

Regarding the real-time video, as it is streamed and not recorded, the main legal issue would be transfering the data in a safe manner, which in this case is done by using encrypted methods.

5. DESIGN AND IMPLEMENTATION

In this chapter, the chosen technologies for the development of the system are defined and justified. Also, the process followed for the design of the system, as well as its implementation are given.

4.1 Chosen technologies

In order to develop the system defined in the previous chapter, first it is necessary to choose which technologies among the ones discussed in the background chapter will be used for each task.

As commented before, the system can be divided into three parts that require different technologies to be implemented.

4.1.1 Bark recognition system

This part of the system requires developing an AI model capable of recognising bark sounds in real time, in order to achieve this, there exist many different programming languages and tools that simplify the making and use of this model. In this case, the ones chosen are:

- **Python:** as stated in *section 2.1.3*, Python is the most common programming language used in this kind of tasks according to Kaggle. Furthermore, Python has many different libraries dedicated to machine learning such as Tensorflow, Pytorch, Keras, etc., as well as other libraries for sound feature extraction or sound recording. For these reasons and the previous experience I had working with it, it was chosen as the programming language for this task.

- **Tensorflow and Keras:** There are many different libraries in Python dedicated to machine learning, but when talking about deep learning the main ones that show up are Tensorflow and Pytorch. These tools are very similar to each other, but in our case thanks to Keras, a library that notably simplifies the process of developing AI models and works on top of Tensorflow, the latter one was chosen to develop the model.

- **LibROSA and sounddevice:** In order to extract the features of the sound and record sound using Python, these two libraries were used as they are well documented, have the tools needed for the project and are easy to use.

Apart from the software, the hardware for developing the monitoring system is also included in this section, as the bark recognition system is run on top of it.

One of the objectives of this project is developing a low cost system. In order to do that, the monitoring device should have enough technical requirements to run a sound classification deep learning model as well as a small server while being relatively cheap. The best option in order to fulfill these requisites is using a single-board computer.

There are many different SBCs in the market, but when looking for cheap, well equipped and, specially, with a big community support in order to facilitate solving the problems that you may encounter along the way, the Raspberry Pi is the one to go.

In this case, the Raspberry Pi chosen to develop the system has been the Raspberry Pi 3 Model B+, which specifications are defined in *section 2.2*, as I already had it bought before starting the project and the specifications were good enough.

Lastly, in relation to the hardware, in order to develop some of the capabilities of the system, three different external accessories are needed, which are a camera, a microphone and a speaker. Luckily, this hardware can be bought cheaply while having good specifications regardless of the brand.

4.1.2 Communication between Raspberry Pi and mobile application

As stated before, one of the main requirements for data transfering is encryption. The data must be always transferred in a secure way in order to ensure the user's privacy.

There are two types of data to be communicated between the Raspberry Pi and the app. These are the data from the server and the real-time video and audio.

For the data retrieved from the server, the most common protocol used is HTTP, as it has been standardized by the *World Wide Web Consortium (W3C)* [64], which is one of the most important communities to develop web standards in the world.

But as commented before data must be sent securely, and HTTP is not a secure protocol, that is the reason why the protocol chosen in this case has been HTTPS, which is an extension of HTTP that ensures a secure communication over the network.

In order to implement the server in the Raspberry Pi that uses this protocol, a Python library called *sauth* [65] will be used due to its simplicity and ease of use.

When talking about the communication of real-time video and audio, HTTPS is not a suitable protocol as it is run over TCP which is a much slower but reliable transport layer protocol. Instead, for video and audio transmission, protocols that run over UDP are much commonly used due to its speed, in exchange for its unreliability in terms of data loss.

In this case, the protocol chosen for this task has been WebRTC as it is a state of the art protocol being standardized by the W3C and compatible with most of the commonly used web browsers as shown in *Fig 2.10* of the background chapter.

In order to implement the WebRTC protocol to share the video and audio captured from the Raspberry Pi a collection of drivers called User space Video4Linux (UV4L) [66] will be used, as they provide the necessary tools to create the connection fast and automatically when booting up the Raspberry Pi.

4.1.3 Mobile application

There are many different frameworks that can be used to develop cross-platform mobile applications, such as Ionic, React Native, Flutter, etc.

Among them, the one chosen to develop the mobile application of this project has been Ionic. As all of these frameworks offer similar capabilities, the main reason for choosing Ionic has been that I have experience working with it along with the Angular framework, which would be translated into a better development of the app.

4.2 System architecture

The system developed in this thesis follows a centralized architecture in which the monitoring component acts as the central server and the user's device connects directly to it without the intervention of external elements.

Fig. 4.1 Simplified system architecture diagram

The *Fig. 4.1* presents a simplified diagram of the system architecture in order to show the data flow and basic architecture. When replacing each component with the chosen technology used to implement it, the following *Fig 4.2* is obtained.

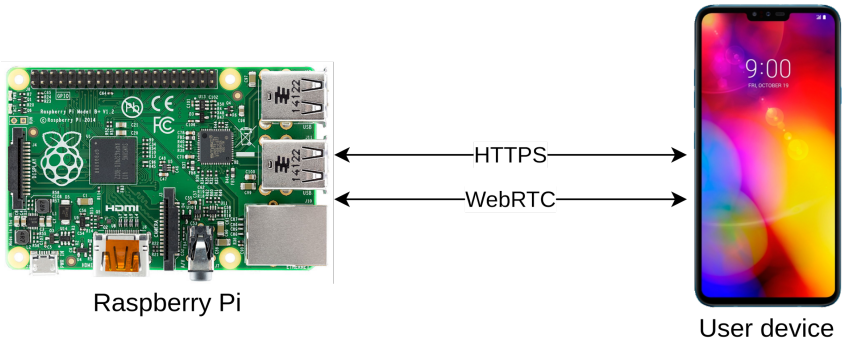

Fig. 4.2 System architecture diagram

Depending on the data type different protocols can be used in order to make the communication easier, faster and more secure. In this case, two different protocols are used, WebRTC for real time audio and video transfer and HTTPS for non-real-time data transfer.

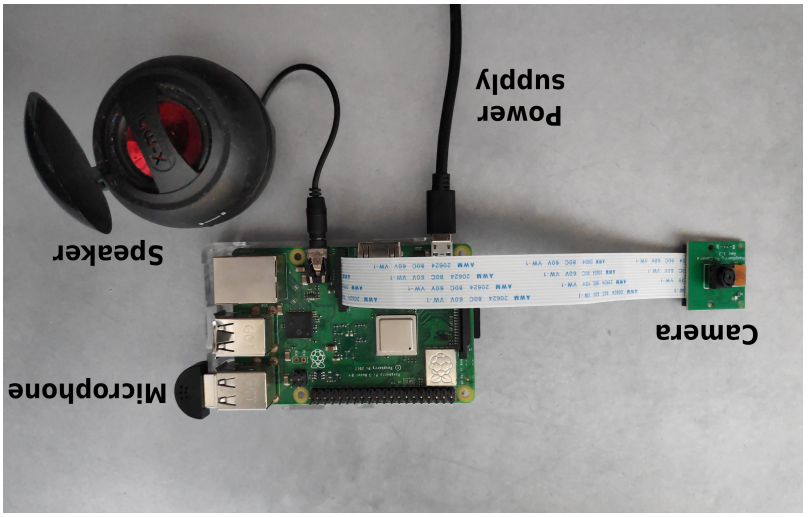

Fig. 4.3 Monitoring device

Also, as shown in *Fig. 4.3*, the Raspberry Pi has an attached camera, microphone and a speaker in order to record video and audio and play the song uploaded by the user.

4.3 Bark recognition system

One of the core features of the project to develop is bark recognition, in order to implement it using machine learning, the steps described in *section 2.1.2* of this thesis will be followed.

1. **Gathering data:** the data needed for this project is divided into recorded barks and other sounds that may take place in a house. In order to gather as much labeled data as possible without having to label it manually, this data has been obtained from the UrbanSound8K dataset, which provides 1,000 recorded barks among other sounds [67]. In the case of other sounds recorded in a house, we have used the microphone attached to the Raspberry Pi to record 1,000 recordings of 4 seconds length, as the

ones from the dataset are at most that long. To sum up, the final dataset used is made of 2,000 recordings of at most 4 seconds length, in which half of them are barks and the other half random sounds in a house.

2. **Data preparation:** in order to prepare the data obtained in the previous step, 5 different transformations have been used to extract the main features of the recordings. These transformations are:

    1. **Mel-frequency cepstral coefficients:** are coefficients for sound representation based on human auditory perception. These coefficients allow a better representation of sound for identifying relevant content.
    2. **Chroma feature:** relates to the twelve different pitch classes and are commonly used to capture harmonic and melodic characteristics of sound, while being robust to changes in timbre and instrumentation.
    3. **Mel-scaled spectrogram:** is a spectrogram where the frequencies are converted to the mel scale, which is a perceptual scale of pitches judged by listeners to be equal in distance from one another.
    4. **Spectral contrast:** each frame of a spectrogram is divided into sub-bands. For each sub-band, the energy contrast is estimated by comparing the mean energy in the top quantile (peak energy) to that of the bottom quantile (valley energy).
    5. **Tonal centroid features:** is a conceptual lattice diagram representing tonal space.

    In order to apply these transformations to the sounds, the Python package LibROSA has been used. After applying them, a new dataset is obtained with 161 data points per sound sample.

3. **Choosing a model:** firstly, we have to establish which type of problem we are dealing with and, therefore, the machine learning technique that could be used to solve it. In order to do that, we need to know whether we have a set of input-output pairs or if creating one is possible, as it is the main difference between machine learning technique families as shown in *Fig 2.2*. In this case the input is a sound and the output is the corresponding classification of that sound into a bark or another sound. Therefore, the problem at hand is a classification problem, which belongs to the supervised machine learning techniques family.

    In order to create a model that solves this problem a DNN will be used, as this type of NNs are easy to implement and suitable for this kind of tasks. The technologies chosen to create this model are Keras and Tensorflow.

The last 4 steps which are training, evaluation, hyperparameter tuning and prediction, are summarized in the following *Table 4.1*, in which the best results of different architectures are

compared by the training and test accuracies in order to find the model that fits better the data without overfitting.

In every test the data is divided into 70% for the training set and 30% for the test set, also the number of barks and other sounds in each set is identical in order to avoid a prediction bias due to the probability of each class appearance.

The basic architecture of the DNN has an input layer of size 161, which is the amount of features extracted per recording, as well as an output layer of size 2, which represents the two possible output labels, “bark” or “other” encoded as 0 or 1.

Regarding the activation functions, the input layers as well as the hidden layers use the *Rectified Linear Units (ReLU)* function, while the output layer uses the *Hyperbolic tangent* function instead of *Softmax* as the inputs to this layer are positive and negative.

Lastly, after each layer except for the output layer, a dropout is applied in order to reduce overfitting. On the input layer the dropout has a frequency of 0.2 and on the hidden layers a frequency of 0.1. Also, after each layer a batch normalization is done in order to standardize the inputs of the layer, accelerating the training process.

For each architecture 10 runs have been done as the dropout adds a random component that makes each run different.

In order to measure the precision of each model, the following metrics have been used:

- **Accuracy:** represents the amount of correct classifications divided by the total amount of classifications.

- **Loss**: represents the average of how uncertain is the model when predicting the class of each datapoint. In order to measure it, the function *‘SparseCategoricalCrossentropy’* from Keras has been used.

The following *Table 4.1* contains the best result for each architecture.

Hyperparemeters and architecture	**Training accuracy**	**Training loss**	**Test accuracy**	**Test loss**
Learning rate : 0.001 Number of hidden layers: 2 Number of neurons per layer 1: 128 Number of neurons per layer 2: 128 Number of epochs: 10	**0.9293**	**0.3317**	**0.8702**	**0.3832**

Learning rate : 0.001 Number of hidden layers: 2 Number of neurons per layer 1: 128 Number of neurons per layer 2: 256 Number of epochs: 10	**0.7717**	**0.3472**	**0.7836**	**0.4097**
Learning rate : 0.001 Number of hidden layers: 2 Number of neurons per layer 1: 256 Number of neurons per layer 2: 128 Number of epochs: 10	**0.8629**	**0.3478**	**0.8818**	**0.4759**
Learning rate : 0.001 Number of hidden layers: 2 Number of neurons per layer 1: 256 Number of neurons per layer 2: 256 Number of epochs: 10	**0.8269**	**0.3486**	**0.7649**	**0.4309**

TABLE. 4.1 ACCURACY RESULTS FOR DIFFERENT DNN ARCHITECTURES

Looking at the results obtained, the best model is the one obtained with the first architecture. In *Fig 4.4* the confusion matrix of this model is shown. As expected, the data is balanced and most of the predictions are correct. Also, it should be noted that most of the errors are false negatives, meaning that the system may not notify some barks but when it does the probability of being right is very high.

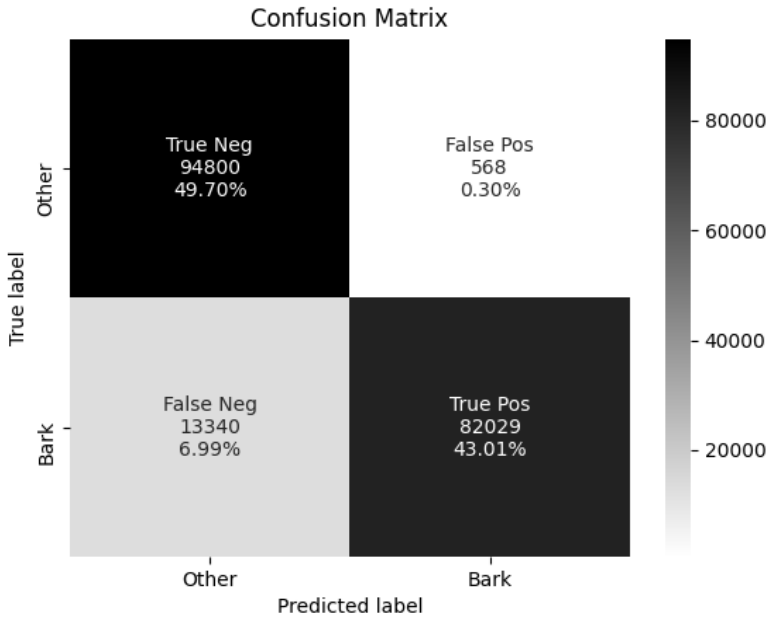

Fig. 4.4 Confusion matrix

In order to use the model, we copy it to the Raspberry Pi and run it using a Python script. The script uses the Python library *sounddevice* in order to record 4 seconds long chunks of sound

repeatedly, which are then fed to the model in order to predict if a bark was recorded or not in real time.

When a bark is detected, the Python script writes in a JSON file the duration, time and date of the recorded bark.

4.4 Communication between Raspberry Pi and mobile application

There are two different types of data that the Raspberry Pi must be able to transfer to the user's device and vice versa, the first one is non-real-time data, regarding the barks record or a song, for which the best option to use is considered the protocol HTTPS and the second one is real time video and audio, in which case WebRTC is a faster and more reliable protocol.

In order to use the HTTPS protocol to share the non-real-time data between the server and the user's device, we create a designated folder in the Raspberry Pi that will act as the server's database. In this case, the only non-real-time data that has to be sent is the bark record and the uploaded song from the user's device which will both be stored in that folder.

Now, the only step left is creating an HTTPS server in that folder, which can be done by using a Python package called "sauth" that allows serving a directory via HTTPS with a username and password. This server is accessible through the Raspberry's IP but requires an authentication in order to access the data.

As for the real-time data communication, the chosen protocol is WebRTC. In order to use this protocol from the server side, a collection of drivers called User space Video4Linux (UV4L) has to be installed. Once the drivers are installed and configured the video and audio captured in real-time by the camera and microphone attached to the Raspberry Pi will be accessible by using the WebSocket protocol to communicate with the Raspberry's IP using the configured username and password.

Once both all the necessary data can be accessed by knowing the Raspberry's IP address and the authentication credentials, the only thing left is making sure that the Raspberry Pi's address is accessible from the internet and not just the local network. In order to do that, one option is to set up a port forward in the local network's router and use the router's WAN IP as the Raspberry's.

4.5 Mobile application

Another core feature of the project developed is the mobile application. In order to create it, the framework Ionic along with the programming language Angular have been used.

The main goal of the app is making the monitoring system easy to install and all the available features intuitive and easy to use. In order to achieve that, the app has been divided into different sections, which match each of the pages of the app.

4.5.1 Home page

The home page is the main page of the app and it is always shown when the app is opened, except for the first time, in which case the app requests the user to introduce the Raspberry Pi's IP and the credentials established when setting up the server in order to start working.

The home page is composed of three buttons that give access to the rest of the pages that the app has. As shown in *Fig 4.3.*, these pages are the settings menu, the bark record and the camera. By clicking on a button the corresponding page is opened.

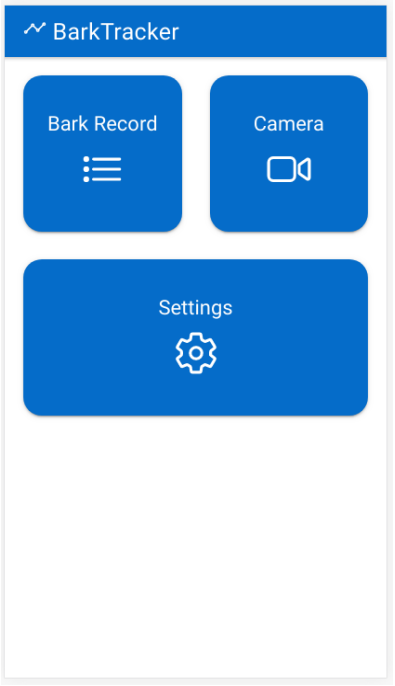

Fig. 4.5 Home page

4.5.2 Settings page

The settings page collects all the required fields in order to use the system as well as some others for extra features like the remote camera access or the song autoplay when a bark is detected.

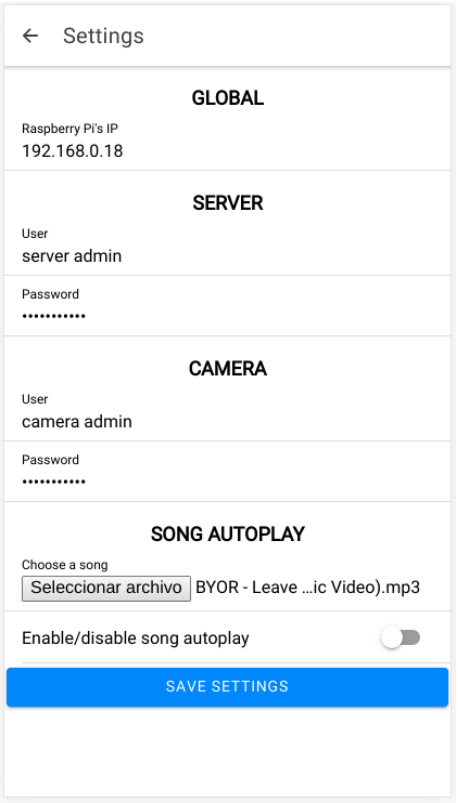

Fig. 4.6 Settings page

As shown in *Fig 4.5*, the required fields for the app to work are the Raspberry Pi's IP address and the server credentials. Other optional fields are the live video credentials and the song uploader.

The data collected in this page is encrypted and stored in the device, except for the song uploaded by the user in which case it is transferred to the Raspberry Pi's server and stored there, as the song is played remotely.

4.5.3 Bark record page

The bark record page keeps the record of the dog barks and displays the time, date and duration of each of them. It also has a filter that allows the user to search according to the bark duration, date and time.

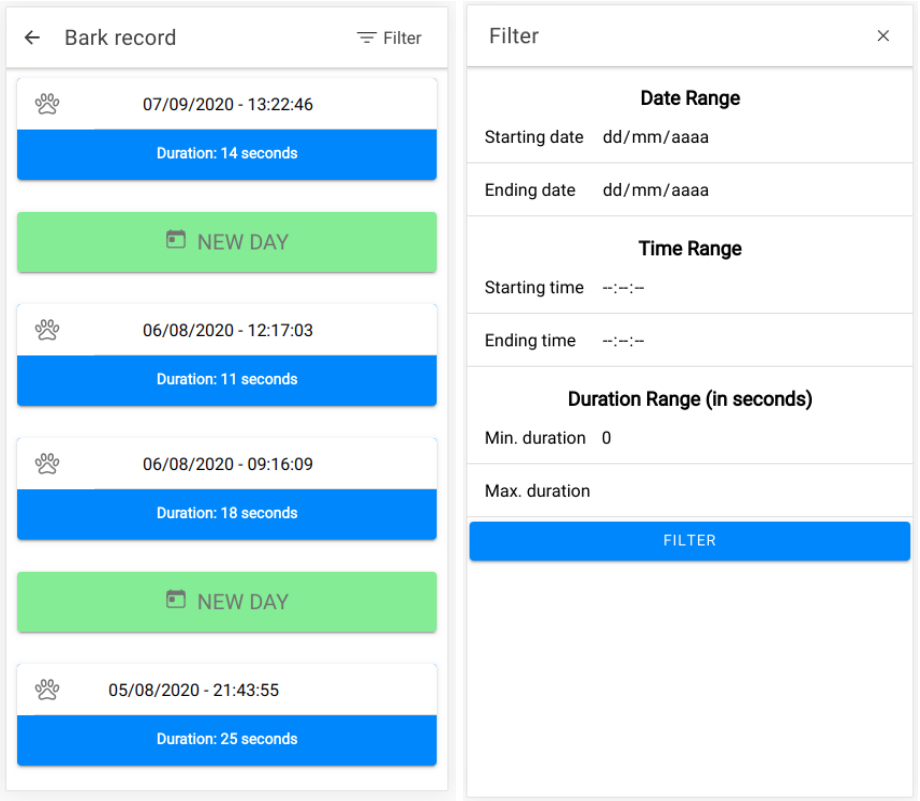

Fig. 4.7 Bark record page

The data of the record is retrieved from the server when the page is opened. This data is loaded dynamically in order to avoid delays caused by the data transfering.

4.5.4 Camera page

The camera page allows the user to receive real-time video and audio from the camera and microphone attached to the monitoring device. It is composed of two buttons that start and end the connection and a video interface that displays the video stream.

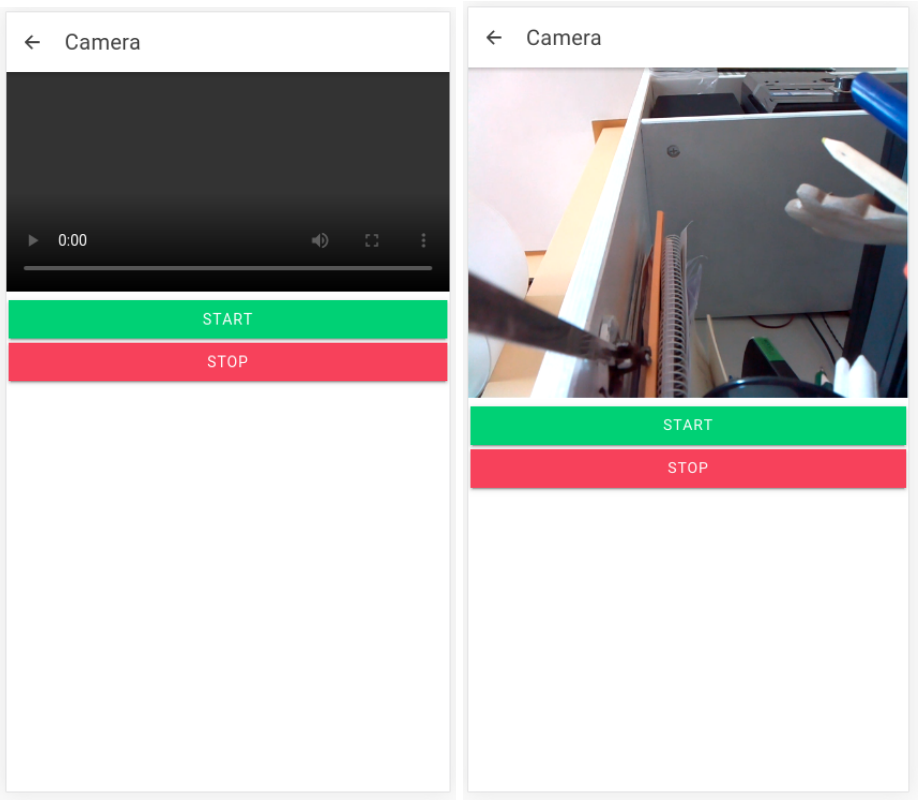

Fig. 4.8 Camera page

Both the bark record page and the camera page, require internet access and the required fields of the settings page correctly filled in along with the camera access credentials.

5. TESTING

In this chapter a set of tests will be performed in order to ensure that the system meets all the requirements specified in the analysis chapter.

5.1 Testing template

In order to document the results of the performed test, the following template in *Table 5.1* will be used.

T-X	
Name	
Steps	
Expected output	
Obtained output	
Related requirements	

TABLE. 5.1 TEMPLATE TABLE FOR TESTING

Each element of the previous table has the following meaning:

- **Identifier**: each test has a unique identifier constituted by the letter "*T-*" followed by a number that represents an enumeration starting from 1.
- **Name**: descriptive name of the test.
- **Steps**: the steps that have been followed to perform the test.
- **Expected output**: refers to the output that the system should give after performing the test.
- **Obtained output**: refers to the output that the system gives after performing the test.
- **Related requirements**: the requirements that have been checked in the test.

5.2 Tests performed

T-1	
Name	Bark detection
Steps	1. Play a bark sound recording near the monitoring system 2. Open the app 3. Go to the “Bark record” page
Expected output	The system records the bark sound and stores the date, time and duration. Then, navigates to the “Bark record” page and displays it.
Obtained output	Correct
Related requirements	RF-1, RF-7, RNF-3

TABLE. 5.2 TEST 1

T-2	
Name	Camera connection
Steps	1. Open the app 2. Go to the “Camera” page 3. Press the “start” button
Expected output	The system navigates to the “Camera” page and streams live video and audio from the monitoring system.
Obtained output	Correct
Related requirements	RF-2, RF-7, RNF-3

TABLE. 5.3 TEST 2

T-3	
Name	Update the settings
Steps	1. Open the app 2. Go to the “Settings” page 3. Fill all the fields

	4. Press the “SAVE SETTINGS” button
Expected output	The system navigates to the “Settings” page and warns the user about the required or wrongly written inputs. Then, it stores the settings on the device's storage.
Obtained output	Correct
Related requirements	RF-4, RF-6, RF-7, RNF-1, RNF-2

TABLE. 5.4 TEST 3

T-4	
Name	Autoplay music
Steps	1. Open the app 2. Go to the “Settings” page 3. Press the “Enable/disable song autoplay” toggle button 4. Press the “Choose file” button 5. Upload a song 6. Press the “SAVE SETTINGS” button 7. Play a bark sound recording near the monitoring system
Expected output	The system navigates to the “Settings” page, enables the autoplay option and uploads the song to the server, only allowing files with “mp3” extension. When the monitoring system detects the bark sound, it starts playing the uploaded song.
Obtained output	Correct
Related requirements	RF-3, RF-4, RF-6, RF-7, RNF-2, RNF-3

TABLE. 5.5 TEST 4

T-5	
Name	Bark record filter
Steps	1. Open the app 2. Got to the “Bark record” page

	3. Press the “Filter” button on the upper right corner 4. Fill the fields 5. Press the “FILTER” button
Expected output	The system navigates to the “Bark record” page, opens the “Filter” modal and displays the bark records that fulfill the introduced inputs in the filter.
Obtained output	Correct
Related requirements	RF-5, RF-7, RNF-3

TABLE. 5.6 TEST 5

T-6	
Name	Maximum request time
Steps	1. Open the app 2. Open the “Bark record” page 3. Time how long it takes to display the bark recordings
Expected output	The system navigates to the “Bark record” page and displays the bark recordings in 20 seconds or less.
Obtained output	Correct
Related requirements	RNF-3, RNF-5

TABLE. 5.7 TEST 6

T-7	
Name	Operating system compatibility
Steps	1. Install the app on an Android device 2. Open the app on the Android device 3. Install the app on an iOS device simulator 4. Open the app on the iOS device simulator
Expected output	The system opens the app correctly in both devices.

Obtained output	Correct
Related requirements	RNF-4

TABLE. 5.8 TEST 7

T-8	
Name	Multiple devices connected
Steps	1. Open the app on 5 different devices 2. Go to the "Bark record" page in every device
Expected output	The system displays the bark records in all of the devices.
Obtained output	Correct
Related requirements	RNF-3, RNF-7

TABLE. 5.9 TEST 8

6. PROJECT MANAGEMENT

In the following chapter the project schedule is carried out, to do so, the project is divided into tasks, to which an estimated solving time is given. Also, the personnel and resources cost is calculated according to the project duration. Lastly, the socio-economic environment in which the project is carried out is studied and explained.

6.1 Project schedule

In order to make the project schedule, firstly is necessary to divide it into tasks:

- **Investigation:** this task involves investigating which solutions exist to the problem at hand and in which way those solutions could be improved.

- **Objective definition:** once the possible improvements over already existing solutions are clear, then the main objectives of the system have to be defined, these objectives can also be seen as tasks to solve during the project schedule.

- **Design of the system:** in this task the goal is to research about the technologies that exist and choose how to combine them in order to fulfil each of the objectives defined previously. Also, the requirements and use cases must be defined in this step.

- **Implementation of the system:** this task involves creating all the components of the system and joining them together following the design defined previously. This task is divided into 3 subtasks, which are creating the AI model for recognising the barks, implementing the protocols to communicate between the Raspberry Pi and the user's device and creating the mobile application to interact with the system.

- **Testing:** check that the defined requirements are met and the system works as expected.

- **Documentation:** document the process followed until the achievement of the project.

In the following *Table 5.1* every task is shown along with the duration in hours and the beginning and ending dates of each of them.

Task	Days	Beginning date	Ending Date
Investigation	10	20/01/20	31/01/20
Objective definition	5	03/02/20	07/02/20

Design of the system	25	10/02/20	13/03/20
Bark recognition system	45	16/03/20	15/05/20
Communication protocols	20	16/03/20	05/06/20
Mobile application	60	11/05/50	05/06/20
Testing	7	08/06/20	16/06/20
Documentation	31	18/05/20	29/06/20
Total of days	203		

TABLE. 5.1 TIME ESTIMATION FOR PROJECT TASKS

The total number of estimated work days is 203, taking into account a working time of 4 hours per day. However, there are some tasks that can be solved in parallel, such as the documentation and the implementation of the system, which involves the tasks of implementing the bark recognition system, the communication protocols and the mobile application. Taking this into account, the following Gantt diagram in *Fig. 5.1* is obtained.

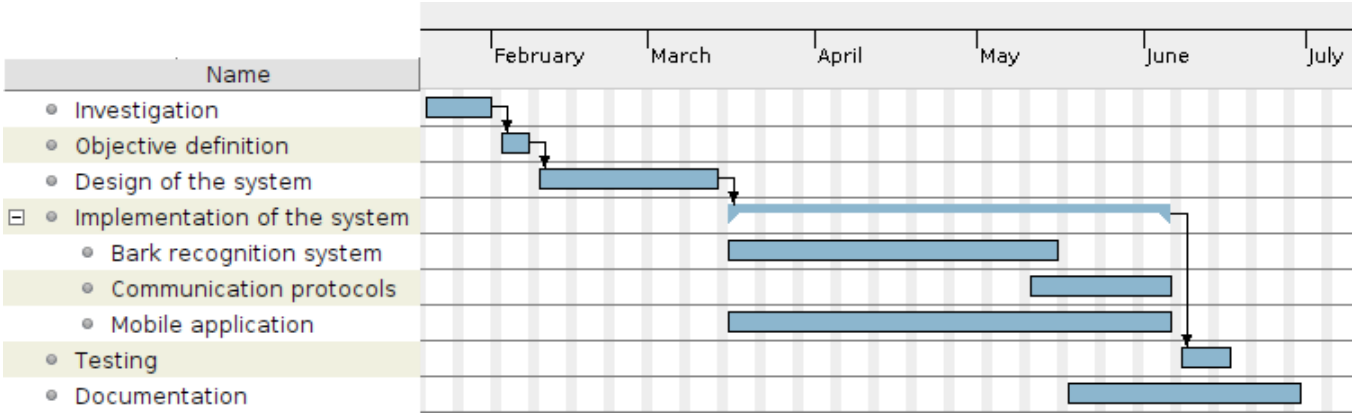

Fig. 5.1 Gantt diagram

In the diagram, it can be observed, in a more visual way, how the tasks are distributed and how much time each takes compared to the others.

When parallelizing tasks the final amount of workdays is 116 days, but the amount of daily hours spent in that period increases to 6 in order to deal with all of them, which gives a total of 584 hours of work.

6.2 Budget estimation

In this section the cost of developing the project will be presented. In order to do it, the previous planification will be taken into account so that an estimated personnel cost can be

carried out. Also, the cost of all the resources used for developing the project will be computed.

6.2.1 Personnel costs

This section refers to the cost of hiring the qualified personnel to carry out the project. In order to estimate this cost, the project schedule defined in the previous section will be used.

According to the website Glassdoor, the average salary for a junior software developer in Spain is 19,000 euros per year [68] From this figure and taking into account that only the author of this thesis took part in the development of the project, we can make the following calculations in order to estimate the cost of the personnel involved.

$$\frac{19000\ €}{12\ months} = 1583€/month\ ;\ \frac{1583\ €/month}{22\ days} = 71,95\ €/day\ ;\ \frac{71{,}95\ €/day}{8\ hours} = 9\ €/hou$$

Profile	Time (hours)	Hourly cost (€)	Total cost (€)
Junior Software Developer	584	9	5256

TABLE. 5.2 PERSONNEL COST

6.2.2 Resources costs

In this section the cost of the resources used for the development of the project are estimated. In order to calculate that cost, only the hardware devices will be taken into account, as all the software used is open-source and, thus, free.

In order to develop this thesis an MSI computer has been used for documenting the process as well as training and testing the deep neural network and developing the mobile application. As this computer was bought in september of 2019 and it is a resource that will be used beyond the scope of this project, the estimated cost that will be taken into account for the resource cost of this project will be calculated with an estimated amortization of use of 5 years as shown in *Table 5.3*.

Resource	Price (€)	Usage time (days)	Amortization period (days)	Total cost (€)
MSI computer	1649	116	1825	105

TABLE. 5.3 RESOURCES COST WITH AMORTIZATION

Regarding the hardware resources that are necessary for the development of the monitoring system, the price for each of them as well as the computed total price with taxes is taken into account in the following *Table 5.3*.

Resource	Cost (€)
Raspberry Pi	40
Raspberry Pi Camera	13
Microphone	8
Speaker	9.5
MSI computer with amortization	105
Total cost	175.5

TABLE. 5.4 RESOURCES COST

Finally, after calculating the personnel cost and the hardware cost, the final cost can be computed in order to have an estimation of the total cost of the project, which is shown in the following *Table 5.5*.

Name	Cost (€)
Personnel cost	5256
Resources cost	175.5
Project total cost	5431.5

TABLE. 5.5 PROJECT TOTAL COST

The total cost of developing and implementing the pet monitoring system described in this thesis is 5431.5 euros. To this quantity we should apply the corresponding taxes, which in Spain would be the IVA and subtract the estimated profit of the product.

6.3 Socio-economic environment

In this section a study of the socio-economic environment in which the project is framed will be carried out in order to define the possible market for the product as well as the way to make it profitable.

The pet monitoring system that has been developed during this thesis is composed of two well differentiated parts, the software which is the mobile application and the hardware which is the monitoring system itself.

From the app perspective, as explained in previous sections, the market of this kind of software has increased noticeably during the last years. This means that even though the amount of competitors grows, so do the customers. In most cases, the way in which these apps get revenue is by in-app purchases, advertisements or pricing them. But, in the case of the app developed in this project, none of these methods would be used, as the app is useless without the required hardware.

Therefore, all the revenue that could be obtained by producing the pet monitoring system would come from the hardware side. As explained along the thesis, the main goal of the system developed is improving the method of detecting dog barks by using AI as well as reducing the cost of this kind of hardware based systems. Even though the developed system meets these requirements, simplifying the installation process and reducing the production cost by buying the components in bulk or mass producing the product, would be necessary in order to compete in such a small market niche.

Finally, a good approximation for the socio-economic impact that the system could have can be obtained by observing the reach of other similar products. In this case, when searching in Amazon, the largest e-commerce retailer by online revenue in the world in 2018 according to Forbes [69], for the best seller product of this kind, the Furbo Dog camera, has more than 11,500 ratings and around 3,500 monthly sales [70][71].

Taking into account these figures belong to the most sold product of this kind, we can conclude that the socio-economic impact of the system would be low.

7. CONCLUSIONS

This chapter collects the conclusions obtained after the completion of the project through the revision of the accomplished objectives that were established at the beginning of the thesis and the proposal of improvements that could be included in future developments.

7.1 Accomplished objectives

The main objective of this thesis was designing and implementing a pet monitoring system that would improve the already existing ones by using AI to detect barks while maintaining the most relevant functionalities of these systems. In order to achieve this objective four subgoals were set at the beginning of this thesis:

1. Choosing suitable technologies in order to make the system inexpensive while meeting the requirements established.

2. Developing an artificial intelligence model capable of recognising bark sounds in real time from a portable device.

3. Implementing a protocol that allows two devices to send video and audio to each other in real time.

4. Developing an application that receives notifications when a bark is detected and allows the user to make different actions.

After the development of the project we can conclude that all these subgoals have been properly accomplished as has been documented along the thesis.

7.2 Future work lines

In relation to the future work that could be done in relation to the project at hand, the following ideas proposed are the most remarkable ones:

- **Sending a notification when a bark is detected:** At the moment the system only retrieves the data from the server when the app is opened, but using push notifications a message could be sent to the user's device to warn about a bark.

- **Moving the camera remotely:** in the current design, the camera attached to the monitoring device is static. In order to make it movable, a servo could be attached to the GPIO of the Raspberry and be controlled remotely through an interface implemented in the app.

- **Improving the AI model:** making the AI model capable of distinguishing between barks, growls, whines, etc. In order to know how the dog is feeling.

- **Talking to the dog:** give the user the possibility of talking to the dog in real-time through the app.

- **Create a bark prevention system:** by analysing the time at which the dog usually barks, the system could automatically alert the user in advance or play music to the dog before it barks.

- **Simplifying the installation process:** at the moment the installation process is long and unintuitive, as all the steps for installing the system have to be done manually. This process could be greatly simplified by creating a script that installed all the required software directly.

8. BIBLIOGRAPHY

[1] "Informe Sectorial". AMVAC, 2017 [Online]. Available: https://www.diagnosticoveterinario.com/wp-content/uploads/2018/11/Estudio-sectorial.pdf. [Accessed: 19- Mar- 2020]

[2] "Dogged By Guilt", Legalandgeneral.com, 2020. [Online]. Available: https://www.legalandgeneral.com/pet-insurance/dogged-by-guilt/. [Accessed: 19- Mar- 2020]

[3] "artificial intelligence | Definition, Examples, and Applications", Encyclopedia Britannica, 2020. [Online]. Available: https://www.britannica.com/technology/artificial-intelligence. [Accessed: 04- Aug- 2020]

[4] "Artificial intelligence", En.wikipedia.org, 2020. [Online]. Available: https://en.wikipedia.org/wiki/Artificial_intelligence. [Accessed: 02- Apr- 2020]

[5] D. Pickell, "The Complete Guide to Machine Learning in 2020", Learn.g2.com, 2020. [Online Image]. Available: https://learn.g2.com/machine-learning. [Accessed: 04- Apr- 2020]

[6] "Machine learning", En.wikipedia.org, 2020. [Online]. Available: https://en.wikipedia.org/wiki/Machine_learning. [Accessed: 06- Apr- 2020]

[7] "What Is Machine Learning? | How It Works, Techniques & Applications", Mathworks.com, 2020. [Online]. Available: https://www.mathworks.com/discovery/machine-learning.html. [Accessed: 07- Apr- 2020]

[8] "Supervised learning", En.wikipedia.org, 2020. [Online]. Available: https://en.wikipedia.org/wiki/Supervised_learning. [Accessed: 07- Apr- 2020]

[9] "Unsupervised learning", En.wikipedia.org, 2020. [Online]. Available: https://en.wikipedia.org/wiki/Unsupervised_learning. [Accessed: 07- Apr- 2020]

[10] "A few useful things to know about machine learning", Homes.cs.washington.edu, 2012. [Online]. Available: https://homes.cs.washington.edu/~pedrod/papers/cacm12.pdf. [Accessed: 10- Apr- 2020]

[11] "Types of Machine Learning Algorithms You Should Know", Medium, 2017. [Online]. Available: https://towardsdatascience.com/types-of-machine-learning-algorithms-you-should-know-953a08248861?gi=9a6c714e4f02. [Accessed: 11- Apr- 2020]

[12] "Artificial neural network", En.wikipedia.org, 2020. [Online]. Available: https://en.wikipedia.org/wiki/Artificial_neural_network. [Accessed: 11- Apr- 2020]

[13] "Fundamentals of Artificial Neural Networks", Google Books, 2020. [Online]. Available: https://books.google.es/books?hl=es&lr=&id=Otk32Y3QkxQC&oi=fnd&pg=PR13&dq=artificial+neural+networks&ots=dc8OiGzaX5&sig=BBYZ-8UXp-o1yssS3bRhvB3vRMo#v=onepage&q=artificial%20neural%20networks&f=false. [Accessed: 13- Apr- 2020]

[14] Glosser.ca, “Colored neural network”. 2013. [Online Image]. Available: https://commons.wikimedia.org/wiki/File:Colored_neural_network.svg. [Accessed: 13- Apr- 2020]

[15] "Deep learning", En.wikipedia.org, 2020. [Online]. Available: https://en.wikipedia.org/wiki/Deep_learning. [Accessed: 16- Apr- 2020]

[16] J. Brownlee, "What is Deep Learning?", Machine Learning Mastery, 2020. [Online]. Available: https://machinelearningmastery.com/what-is-deep-learning/. [Accessed: 20- Apr- 2020]

[17] "Prediction of wind pressure coefficients on building surfaces using Artificial Neural Networks", ResearchGate, 2017. [Online Image]. Available: https://www.researchgate.net/publication/321259051_Prediction_of_wind_pressure_coefficients_on_building_surfaces_using_Artificial_Neural_Networks. [Accessed: 14- Apr- 2020]

[18] "The 7 Steps of Machine Learning", Medium, 2020. [Online]. Available: https://towardsdatascience.com/the-7-steps-of-machine-learning-2877d7e5548e. [Accessed: 20- Apr- 2020]

[19] J. Brownlee, "Impact of Dataset Size on Deep Learning Model Skill And Performance Estimates", Machine Learning Mastery, 2020. [Online]. Available: https://machinelearningmastery.com/impact-of-dataset-size-on-deep-learning-model-skill-and-performance-estimates/. [Accessed: 22- Apr- 2020]

[20] "Mel-frequency cepstrum", En.wikipedia.org, 2020. [Online]. Available: https://en.wikipedia.org/wiki/Mel-frequency_cepstrum. [Accessed: 25- Apr- 2020]

[21] S. Tjoa, "Mel Frequency Cepstral Coefficients", Musicinformationretrieval.com, 2020. [Online]. Available: https://musicinformationretrieval.com/mfcc.html. [Accessed: 27- Apr- 2020]

[22] D. Ellis, "Chroma Feature Analysis and Synthesis", Labrosa.ee.columbia.edu, 2007. [Online]. Available: https://labrosa.ee.columbia.edu/matlab/chroma-ansyn/. [Accessed: 02- May- 2020]

[23] M. Smales, "Sound Classification using Deep Learning", Medium, 2019. [Online]. Available: https://medium.com/@mikesmales/sound-classification-using-deep-learning-8bc2aa1990b7. [Accessed: 03- May- 2020]

[24] N. Srinivasan, "Building an Audio Classifier using Deep Neural Networks", KDnuggets, 2017. [Online]. Available: https://www.kdnuggets.com/2017/12/audio-classifier-deep-neural-networks.html. [Accessed: 05- May- 2020]

[25] J. Brownlee, "Overfitting and Underfitting With Machine Learning Algorithms", Machine Learning Mastery, 2019. [Online]. Available: https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/. [Accessed: 07- May- 2020]

[26] "2018 Kaggle ML & DS Survey", Kaggle.com, 2018. [Online]. Available: https://www.kaggle.com/kaggle/kaggle-survey-2018. [Accessed: 12- May- 2020]

[27] "Top languages used", Kaggle.com, 2019. [Online Chart]. Available: https://www.kaggle.com/tvirot/top-languages-used. [Accessed: 16- May- 2020]

[28] "Python (programming language)", En.wikipedia.org, 2020. [Online]. Available: https://en.wikipedia.org/wiki/Python_(programming_language). [Accessed: 23- May- 2020]

[29] "The Global Developer Population 2019", SlashData, 2020. [Online]. Available: https://slashdata-website-cms.s3.amazonaws.com/sample_reports/EiWEyM5bfZe1Kug_.pdf. [Accessed: 27- May- 2020]

[30] "Stack Overflow Developer Survey 2018", Stack Overflow, 2018. [Online]. Available: https://insights.stackoverflow.com/survey/2018#most-loved-dreaded-and-wanted. [Accessed: 28- May- 2020]

[31] "The Python Package Index", PyPI, 2020. [Online]. Available: https://pypi.org/. [Accessed: 28- Apr- 2020]

[32] A. Beklemysheva, "Why Use Python for AI and Machine Learning?", Steelkiwi.com, 2020. [Online]. Available: https://steelkiwi.com/blog/python-for-ai-and-machine-learning/. [Accessed: 01- Jun- 2020]

[33] "TensorFlow", Es.wikipedia.org, 2020. [Online]. Available: https://es.wikipedia.org/wiki/TensorFlow. [Accessed: 02- Jun- 2020]

[34] "Keras documentation: About Keras", Keras.io, 2020. [Online]. Available: https://keras.io/about/. [Accessed: 04- Jun- 2020]

[35] "librosa — librosa 0.8.0 documentation", Librosa.org, 2020. [Online]. Available: https://librosa.org/doc/latest/index.html#. [Accessed: 04- Jun- 2020]

[36] "Play and Record Sound with Python — python-sounddevice, version 0.4.0", Python-sounddevice.readthedocs.io, 2020. [Online]. Available: https://python-sounddevice.readthedocs.io/en/0.4.0/. [Accessed: 04- Jun- 2020].

[37] "Raspberry Pi", En.wikipedia.org, 2020. [Online]. Available: https://en.wikipedia.org/wiki/Raspberry_Pi. [Accessed: 06- Jun- 2020]

[38] "Single-board computer", En.wikipedia.org, 2020. [Online]. Available: https://en.wikipedia.org/wiki/Single-board_computer. [Accessed: 06- Jun- 2020]

[39] L. Bosch, “Raspberry Pi B+ top”. 2014. [Online Image]. Available: https://commons.wikimedia.org/wiki/File:Raspberry_Pi_B%2B_top.jpg. [Accessed: 06-Jun-2020]

[40] "Raspberry Pi 3 B+ mechanical drawing", Raspberrypi.org, 2020. [Online]. Available: https://www.raspberrypi.org/documentation/hardware/raspberrypi/mechanical/rpi_MECH_3bplus.pdf. [Accessed: 06- Jun- 2020]

[41] Wirepath, “Raspberry Pi block function”. 2012. [Online Image]. Available: https://commons.wikimedia.org/w/index.php?curid=19277124. [Accessed: 06- Jun- 2020].

[42] N. Heath, "Raspberry Pi and machine learning: How to get started", TechRepublic, 2020. [Online]. Available: https://www.techrepublic.com/article/raspberry-pi-and-machine-learning-how-to-get-started/. [Accessed: 06- Jun- 2020]

[43] "Oficial Raspberry Pi’s camera", Amazon.es, 2020. [Online]. Available: https://www.amazon.es/gp/product/B07TXGGJMT/ref=ppx_yo_dt_b_asin_title_o01_s00?ie=UTF8&psc=1. [Accessed: 06- Jun- 2020]

[44] "Raspberry Pi 3 Model B+", Static.raspberrypi.org, 2020. [Online]. Available: https://static.raspberrypi.org/files/product-briefs/200206+Raspberry+Pi+3+Model+B+plus+Product+Brief+PRINT&DIGITAL.pdf. [Accessed: 06- Jun- 2020]

[45] "Raspbian Home Page", Raspbian.org, 2020. [Online]. Available: https://www.raspbian.org/. [Accessed: 06- Jun- 2020]

[46] Raspbian Project, “Raspbian 2019.04 application menu”. 2019. [Online Image]. Available: https://commons.wikimedia.org/wiki/File:Raspbian_2019.04_application_menu.jpg. [Accessed: 06- Jun- 2020].

[47] M. Rouse, "What is Real-Time Communications (RTC)?", SearchUnifiedCommunications, 2020. [Online]. Available: https://searchunifiedcommunications.techtarget.com/definition/real-time-communications. [Accessed: 10- Jun- 2020]

[48] "WebRTC Home Page", Webrtc.github.io, 2020. [Online]. Available: https://webrtc.github.io/webrtc-org/. [Accessed: 10- Jun- 2020]

[49] "WebRTC 1.0: Real-time Communication Between Browsers", W3.org, 2020. [Online]. Available: https://www.w3.org/TR/webrtc/. [Accessed: 10- Jun- 2020]

[50] S. Dutton, "Getting Started with WebRTC", HTML5 Rocks - A resource for open web HTML5 developers, 2014. [Online]. Available: https://www.html5rocks.com/en/tutorials/webrtc/basics/. [Accessed: 10- Jun- 2020].

[51] "Is WebRTC ready yet?", Iswebrtcreadyyet.com, 2020. [Online]. Available: http://iswebrtcreadyyet.com/legacy.html. [Accessed: 10- Jun- 2020].

[52] Gartner, "Global smartphone sales to end users from 1st quarter 2009 to 2nd quarter 2018, by operating system (in million units)", Statista, 2018. [Online Chart]. Available: https://www-statista-com.biblioteca5.uc3m.es/statistics/266219/global-smartphone-sales-since-1st-quarter-2009-by-operating-system/. [Accessed: 11- Jun- 2020].

[53] "Ionic Framework", ionicframework, 2020. [Online]. Available: https://ionicframework.com/docs. [Accessed: 13- Jun- 2020]

[54] "React Native", reactnative, 2020. [Online]. Available: https://reactnative.dev/. [Accessed: 13- Jun- 2020]

[55] "Most Wished For in Pet Cameras & Monitors", Amazon.com, 2020. [Online]. Available: https://www.amazon.com/-/es/gp/most-wished-for/pet-supplies/17440052011/ref=zg_bs_tab_t_mw?language=en_US. [Accessed: 15- Jun- 2020].

[56] "Top dog monitoring apps in Google Play", Play.google.com, 2020. [Online]. Available: https://play.google.com/store/search?q=dog%20monitoring%20app&c=apps. [Accessed: 15- Jun- 2020].

[57] "Furbo Dog Camera", Shopus.furbo.com, 2020. [Online]. Available: https://shopus.furbo.com/products/furbo-dog-camera. [Accessed: 25- Jun- 2020].

[58] "Barkio: Dog Monitor", Barkio: Dog Monitor for iOS, Android, macOS, and Windows, 2020. [Online]. Available: https://barkio.com/. [Accessed: 25- Jun- 2020].

[59] "Camera Module - Raspberry Pi Documentation", Raspberrypi.org, 2020. [Online]. Available: https://www.raspberrypi.org/documentation/hardware/camera/. [Accessed: 03- Jul- 2020].

[60] "Regulation (EU) 2016/679 of the European Parliament and of the Council", Eur-lex.europa.eu, 2016. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679. [Accessed: 06- Jul- 2020].

[61] "Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales.", Boe.es, 2018. [Online]. Available: https://www.boe.es/buscar/pdf/2018/BOE-A-2018-16673-consolidado.pdf. [Accessed: 06- Jul- 2020].

[62] "Ley 34/2002, de 11 de julio, de servicios de la sociedad de la información y de comercio electrónico.", boe.es, 2002. [Online]. Available: https://www.boe.es/buscar/pdf/2002/BOE-A-2002-13758-consolidado.pdf. [Accessed: 06- Jul- 2020].

[63] J. Sánchez and C. Sandoval, "Applicable Law to mobile applications in Spain", Mariscal-abogados.com. [Online]. Available: https://www.mariscal-abogados.com/applicable-law-to-mobile-applications-in-spain/. [Accessed: 06- Jul- 2020].

[64] "About W3C", W3.org, 2020. [Online]. Available: https://www.w3.org/Consortium/. [Accessed: 10- Jul- 2020].

[65] "sauth", PyPI, 2018. [Online]. Available: https://pypi.org/project/sauth/. [Accessed: 10- Jul- 2020].

[66] "UV4L", Linux-projects.org, 2020. [Online]. Available: http://www.linux-projects.org/uv4l/. [Accessed: 10- Jul- 2020].

[67] "UrbanSound8K", Urban Sound Datasets. [Online]. Available: https://urbansounddataset.weebly.com/urbansound8k.html. [Accessed: 14- Jul- 2020].

[68] "Sueldo: Junior Software Developer", Glassdoor, 2020. [Online]. Available: https://www.glassdoor.es/Sueldos/junior-software-developer-sueldo-SRCH_KO0,25.htm?countryRedirect=true. [Accessed: 16- Jul- 2020].

[69] N. Angelovska, "Top 5 Online Retailers: 'Electronics And Media' Is The Star Of E-commerce Worldwide", Forbes, 2019. [Online]. Available: https://www.forbes.com/sites/ninaangelovska/2019/05/20/top-5-online-retailers-electronics-and-media-is-the-star-of-e-commerce-worldwide/#49f8fe8a1cd9. [Accessed: 18- Jul- 2020].

[70] "Amazon Sales Estimates", My Dog Is A Robot, 2019. [Online]. Available: https://www.mydogisarobot.com/amazon-sales-estimates/. [Accessed: 18- Jul- 2020].

[71] "Furbo Dog Camera", Amazon.com, 2020. [Online]. Available: https://www.amazon.com/Furbo-Dog-Camera-Designed-Compatible/dp/B01FXC7JWQ/ref=zg_mw_17440052011_1?_encoding=UTF8&refRID=03GAD037SQZCW3ZQPVNY&th=1. [Accessed: 18- Jul- 2020].

9. ACRONYMS

AMVAC	Asociación Madrileña de Veterinarios de Animales de Compañía
U.K.	United Kingdom
AI	Artificial Intelligence
ANN	Artificial Neural Network
DNN	Deep Neural Network
MFCCs	Mel-Frequency Cepstral Coefficients
UX	User Experience
UI	User Interface
SBC	Single-Board Computer
MP	Megapixels
GPIO	General Purpose Input/Output
HD	High Definition
RTC	Real Time Communication
API	Application Programming Interface
P2P	Peer-To-Peer
OS	Operating System
DIY	Do It Yourself
EU	European Union
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure

TCP	Transmission Control Protocol
UDP	User Datagram Protocol

Appendix I. Application user manual

The following appendix has the objective of explaining how to install and use the system developed in this thesis.

1. System installation

In order to use the system, first the user needs to install it in the room of the house where the dog will be monitored.

The system installation can be divided in two parts, the app and the monitoring system.

1.1 Monitoring system installation

Firstly, the user must install the monitoring system, as the app functionalities depend on it. In order to do so, the following hardware is required:

1. **Raspberry Pi:** this Raspberry Pi must be the model 3 B+ or higher to ensure that the system works properly.

2. **Microphone:** the microphone must be attached to the Raspberry Pi via USB. It is recommended to use the microphone[1] that has been chosen in this project for a better experience, as the AI model has been developed using it, which means that it will be more accurate.

3. **Camera:** the camera used in the development of this project has been the Raspberry Pi's official camera[2], as it comes with the necessary connectors to attach it to Raspberry Pi.

4. **Speaker:** lastly, a speaker that can be used while charging is required as it must always be on and connected to the Raspberry Pi via the audio jack, the USB port, Bluetooth connection or any other valid method. In this case, the speaker used in the development of the system can be bought in the following link[3].

[1] SunFounder USB 2.0 Mini Microphone purchase link
[2] Raspberry Pi's official camera purchase link
[3] XMI X-mini II Mini Speaker purchase link

Once having all the necessary components, the installation process is as follows:

1. Setting up the Raspberry Pi with Raspbian Stretch OS. In order to do so, the tutorial in the following link https://projects.raspberrypi.org/en/projects/raspberry-pi-setting-up explains the process in detail.

2. Once the Raspberry is ready to be used, the following software must be installed on it:

   - **Python3.6:** to do so, open the terminal and type the following command *'sudo apt-get install libssl-dev; wget https://www.python.org/ftp/python/3.6.5/Python-3.6.5.tgz; tar -xvf Python-3.6.5.tgz; cd Python-3.6.5/; ./configure; sudo make; sudo make install'*. Once Python3.6 is installed, the pip3.6 must be install with the following command *'curl https://bootstrap.pypa.io/get-pip.py | sudo -H python3.6'*. After that, the following libraries can be installed:

     ✓ **LibROSA**
     ✓ **Numpy**
     ✓ **TensorFlow**
     ✓ **Sounddevice**
     ✓ **Joblib**
     ✓ **Sauth**

     In order to install them, type the following command in the terminal *'pip3.6 install librosa numpy tensorflow sounddevice joblib sauth'*.

   - **User space Video4Linux (UV4L):** In order to install this set of drivers the following steps must be followed:

     1. Open the terminal

     2. Type *'curl http://www.linux-projects.org/listing/uv4l_repo/lpkey.asc | sudo apt-key add -'* in the terminal.

     3. Add the following line *'deb http://www.linux-projects.org/listing/uv4l_repo/raspbian/stretch stretch main'* to the file *'/etc/apt/sources.list'*.

     4. Type *'sudo apt-get update; sudo apt-get install uv4l uv4l-raspicam'* in the terminal. At this point the drivers are already installed.

5. In order to make it load at boot, type the following *'sudo apt-get install uv4l-raspicam-extras'* in the terminal.

6. Go to the following file *'/etc/uv4l/uv4l-raspicam.conf'* and replace it with the one with the same name uploaded to the *'user-installation'* folder in the following GitHub repository https://github.com/AlejandroMarchan/BarkTracker.

7. Then go to line 177 of the *'uv4l-raspicam.conf'* file and change the server admin password *'*PASSWORD*'* to the desired one. Remember the password as you will need to introduce it in the camera settings of the app along with the user *'admin'* later.

```
173 server-option = --port=4200
174 # server-option = --bind-host-address=localhost
175 # server-option = --md5-passwords=no
176 # server-option = --user-password=*PASSWORD*
177 server-option = --admin-password=*PASSWORD*
178 ### To enable 'config' user authentication
179 # server-option = --config-password=*PASSWORD*
```

Fig. I.1 Camera credentials

In order to check the status of the driver or restart it, the following command can be typed in the terminal *'sudo service uv4l_raspicam status'* or *'sudo service uv4l_raspicam restart'* respectively.

3. At this point all the software has been installed. Now, the AI model and the python script to execute have to be copied to the Raspberry Pi. In order to do that, copy the folder '*bark-detection*' from the GitHub repository previously commented to the Raspberry Pi desktop.

4. Open a terminal in the '*bark-detection'* folder location and run the following command *'python3.6 run_model.py'*. It is important after doing it to leave the terminal open.

5. Open a terminal in the *'server'* folder that is inside the '*bark-detection'* folder and run the following command *'sauth --https *USER* *PASSWORD*'*, where the user and password are the server credentials that need to be filled in the app's *settings* page. As well as in the previous step, it is important to leave the terminal open.

6. Now the only step left is forwarding the Raspberry Pi's port in order to make it accessible from the internet and not just the local network. In order to do that, in the following link https://portforward.com/router.htm, there is a guide on how to port forward the most common routers. In order to know the Raspberry Pi's IP open a

terminal and type the command *'ifconfig'*, then check the number highlighted in *Fig I.2*. Lastly, be sure to introduce por *4200* when forwarding the port and check for the router's IP, as is the one that must be introduced in the Raspberry's IP section of the app's *settings* page.

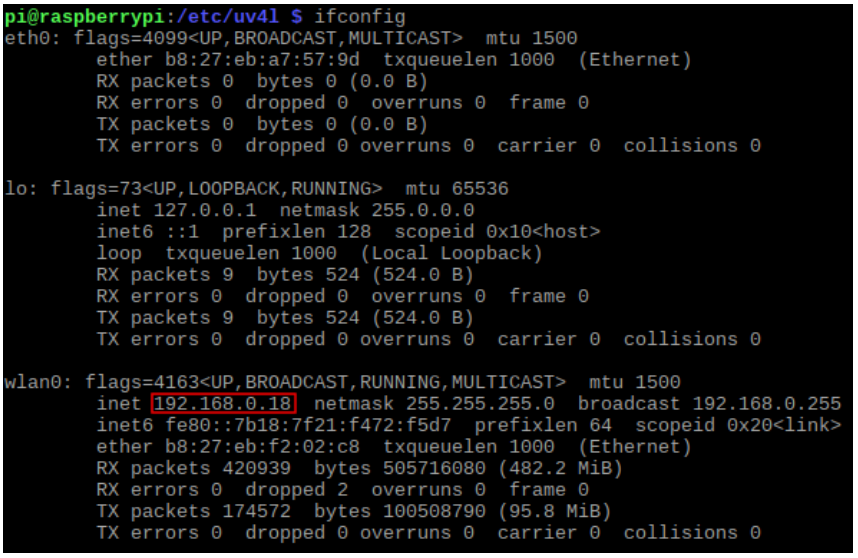

Fig. I.2 Raspberry Pi's IP

After these steps, the monitoring system would be installed and ready to be used.

1.2 Mobile application installation

The app developed in this project has not been published yet in any app store, therefore, in order to install it the user must download it and install it from a different source. The two target smartphone operating systems for this application are Android and iOS, but unfortunately this last one does not allow installing apps from a different source than their own store, therefore, the only way of using the app on a device for the moment is by means of an Android phone. The process for installing apps from outside of the "Google Play Store" is as follows:

1. Download the file *'BarkTracker.apk'* from the *'user-installation'* folder in the GitHub repository previously mentioned to the Android device desired for the installation.

2. Go to the device file explorer and click on the file, the device's default package installer will open the file.

3. Normally, a security warning will arise asking to 'Allow installs from unknown sources', in this case as the app is safe, this option can be enabled. After the installation this option should be disabled again.

4. After enabling the option and clicking again on the file, the app will start installing. Once installed, the app will appear in the phone's app collection in order to be used.

Fig. I.3 App icon

Once the app is installed, the user needs to introduce the settings in order to connect to the monitoring system. These settings have been defined along the monitoring system installation process and are necessary for the system to work properly.

Once the app is configured all the available functionalities explained throughout the document can be used.

Appendix II. System installation manual

The following appendix is aimed at the developers that may want to continue the development of this project, either adding new functionalities or improving already existing ones.

The system developed has two different working environments that have to be set in order to work either on the mobile application or on the AI model.

1. Mobile application environment

As stated before, the mobile application has been developed using the Ionic Framework along with Angular. Therefore, in order to keep developing it the following steps should be followed:

1. Download the *'app'* folder from the GitHub repository in the following link https://github.com/AlejandroMarchan/BarkTracker.

2. Download node.js from the official page https://nodejs.org/es/download/

3. Open a terminal and run the command *'npm install -g @ionic/cli'*.

4. Open a terminal and run the command *'npm install -g @angular/cli'*.

5. Open a terminal and navigate to the downloaded folder, then run the following command *'ionic serve -o'*, which will open a new browser window with the app.

6. Now, use the desired code editor in order to make changes that will update automatically in the browser's opened window.

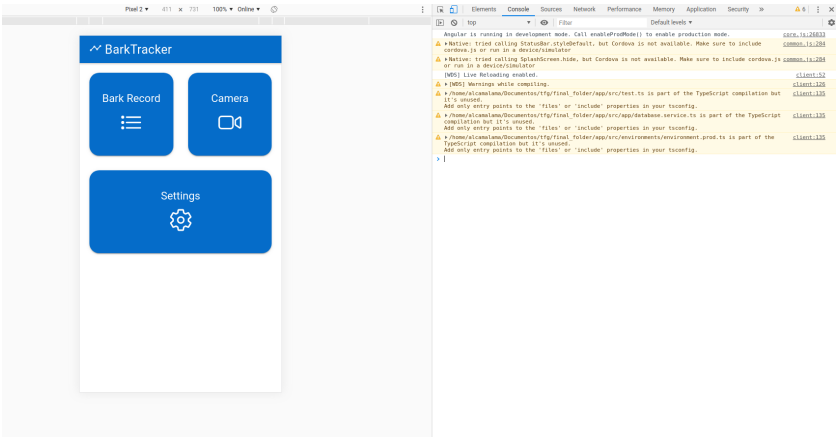

Fig. II.1 App in Google Chrome browser

2. AI model environment

The AI model has been developed using Python with the Tensorflow library along with Keras. In order to set the environment to work on this model, the following steps must be followed:

1. Download the *'model-dev'* folder from the GitHub repository in the following link https://github.com/AlejandroMarchan/BarkTracker.

2. As the data used to train the model was too large for storing it in GitHub, it has to be downloaded from the following link https://drive.google.com/drive/folders/1wdWJPCiM22IYZsOLJefhM0qBKgnEhE5s?usp=sharing. Once downloaded, it has to be moved to the *'model-dev'* folder.

3. Install Python3 in the machine. The process of installing Python3 varies from one OS to another. The following page https://realpython.com/installing-python/ covers in detail the installation processes for Windows, Linux and MacOS.

4. Install the TensorFlow library. This library comes with Keras included. In order to install it, open a terminal and execute the command *'pip3 install Tensorflow'*.

5. Lastly, there is a file named *'train.py'* and another one named *'test.py'*, in which the model architecture is defined as well as the hyperparameters of the model. These files can be modified in order to generate and test new models by executing them.