OPTICAL WARNING OF BICYCLE RIDERS CONCERNING MOTORIZED VEHICLES CLOSING FROM BEHIND USING THE SMARTPHONE MICROPHONE

A Degree Thesis

Submitted to the Faculty of the

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

And

Technischen Universität Wien

by

Alba Talaya Vidal

In partial fulfilment

of the requirements for the degree in

TELECOMMUNICATIONS ENGINEERING

Advisors:

Gerhard Navratil

Anna M. Calveras Auge

Vienna, January 2022

Abstract

Deaf cyclists do not receive acoustic warnings of motor vehicles approaching from behind. This can be very dangerous because they can get startled and wobble, making them loose stability and more exposed to possible accidents.

The purpose of this thesis is a proof of concept, not a product. The aim is to investigate whether it is possible to detect approaching vehicles with a smartphone microphone through an app and to announce them visually to the deaf cyclist through AR-glasses.

The result of the work is a smartphone app, developed in Ionic/Angular, that detects approaching vehicles through a local server coded in Python/Flask, and displays a visual warning on AR-glasses via wireless connection to the smartphone. The final accuracy of the detection is of 87%.

Resum

Les/els ciclistes sords no reben avisos acústics de vehicles motoritzats que s'apropen per darrera. Això pot ser molt perillós, ja que pot causar sobresalts que les/els pot fer trontollar, causant així una pèrdua d'estabilitat i exposant-les/los més a possibles accidents.

La intenció d'aquest treball de fi de grau és obtenir una proba de concepte, no un producte. L'objectiu és investigar si és possible detectar vehicles que s'apropen, a través del micròfon d'un mòbil gràcies a una app, i anunciar-ho visualment a la/el ciclista sord mitjançant ulleres de realitat augmentada.

El resultat del treball és una aplicació mòbil desenvolupada amb Ionic/Angular, que detecta vehicles que s'acosten mitjançant un servidor local programat en Python/Flask, i mostra una alerta visual en unes ulleres de realitat augmentada via connexió Bluetooth amb el telèfon mòbil. La precisió final de la detecció és d'un 87%.

Resumen

Las/los ciclistas sordos no reciben avisos acústicos de vehículos motorizados que se acercan por detrás. Esto puede ser muy peligroso, ya que puede causar sobresaltos que las/los puede hacer tambalearse, causando así una pérdida de estabilidad y exponiéndolas/los más a posibles accidentes.

La intención de este trabajo de fin de grado es obtener una prueba de concepto, no un producto. El objetivo es investigar si es posible detectar vehículos que se acercan, a través del micrófono de un móvil gracias a una app, i anunciar-lo visualmente al ciclista sordo mediante gafas de realidad aumentada.

El resultado del trabajo es una aplicación móvil desarrollada con Ionic / Angular, que detecta vehículos que se acercan mediante un servidor local programado en Python/Flask, y muestra una alerta visual en unas gafas de realidad aumentada vía conexión Bluetooth con el teléfono móvil. La precisión final de la detección es de un 87%.

Acknowledgements

First of all, I would like to thank the thesis supervisor Gerhard Navratil for giving me the opportunity to work on this project, and guiding me along the way of this long process of learning, making mistakes and figuring them out.

I would also like to thank the second supervisor Anna M. Calveras Auge, for helping me solve all the doubts I had and giving me valuable tips.

Additionally, I am grateful to Bartosz Mazurkiewicz for helping me understand the technicalities of the AR-glasses.

Lastly, I would not have been able to make this project without the endless support of my family and friends, who were always there when I hesitated the most.

Revision history and approval record

Revision	Date	Purpose
0	21/12/2021	Document creation
1	11//1/2022	Document revision
2	12/01/2022	Document revision
3	18/01/2022	Document revision

DOCUMENT DISTRIBUTION LIST

Name	e-mail
Alba Talaya Vidal	talayaalba@gmail.com
Gerhard Navratil	Gerhard.Navratil@geo.tuwien.ac.at
Anna M. Calveras Auge	anna.calveras@upc.edu

Written by: Alba Talaya Vidal		**Reviewed and approved by:** Gerhard Navratil, Anna M. Calveras Auge	
Date	20/12/2021	**Date**	18/01/2022
Name	Alba Talaya Vidal	**Name**	Gerhard Navratil Anna M. Calveras Auge
Position	Project Author	**Position**	Project Supervisors

Table of contents

List of Figures

List of Tables:

1. Introduction

1.1. Statement of purpose

The purpose of this thesis is a proof of concept, not a product. The aim is to investigate whether it is possible to detect approaching vehicles with a smartphone microphone through an app and to announce them visually to the deaf cyclist through AR-glasses. The result of the work is a smartphone app that detects approaching vehicles and displays a warning on AR-glasses via wireless connection.

1.2. Requirements and specifications

Project requirements:

- Attempt to detect approaching motorized vehicles through a smartphone's microphone.
- If the previous requirement was successfully done, an android app will be implemented to use the smartphone's microphone to analyse the surrounding sounds and detect the automobiles.
- The app will be wirelessly connected to AR-glasses that will display a visual warning whenever a vehicle is detected.

Project specifications:

- Detect automobiles from a microphone:
  - If the attempt was unsuccessful, the reasons should be properly explained.
  - If the attempt was successful, you should be able to detect whenever a vehicle is approaching from a sound recording.
    - The app should be able to use the smartphone's microphone and identify when a car is coming.
    - The smartphone should be able to connect to the AR-glasses and display a visual warning when a car is detected.

1.3. Methods and procedures

This thesis has been developed from scratch. The department has previously done some work with AR-glasses*, but nothing else required for this project.

*The group performed research on interaction with 3D data and visualization of underground features.

1.4. Work plan with tasks, milestones and a Gantt diagram.

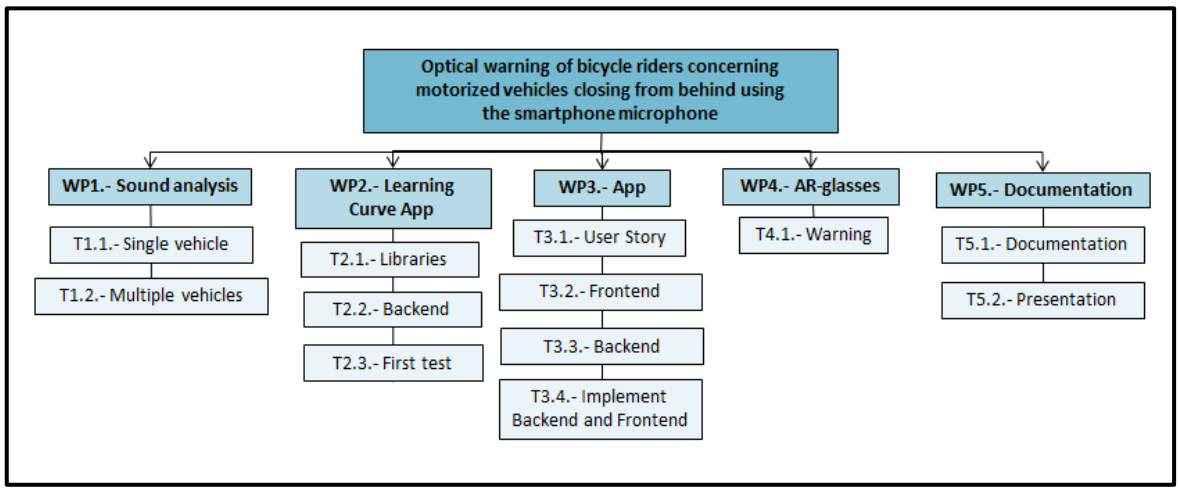

Figure 1 - Work Plan: Work Plan diagram with all the corresponding work packages and tasks

Milestones:

WP#	Task#	Short title	Milestone / deliverable	Date
1	1	Single vehicle	Result	19/11/2021
	2	Multiple vehicles	Result	10/12/2021
2	3	First test	First test	14/12/2021
3	1	User Story	User Story	15/12/2021
	4	Implement Backend and Frontend	Code	17/1/2022
6	1	Documentation	Documentation	19/1/2022
	2	Presentation	Presentation and demo	20/1/2022

Table 1 – Milestones: Milestones achieved during the developing of the project

Gantt Diagram:

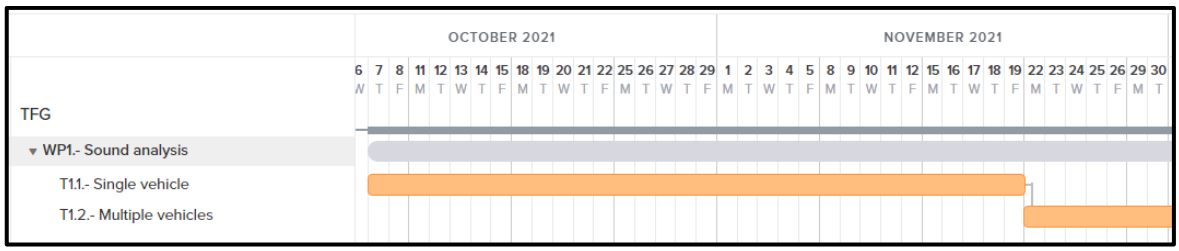

Figure 2 - Gantt Diagram 1

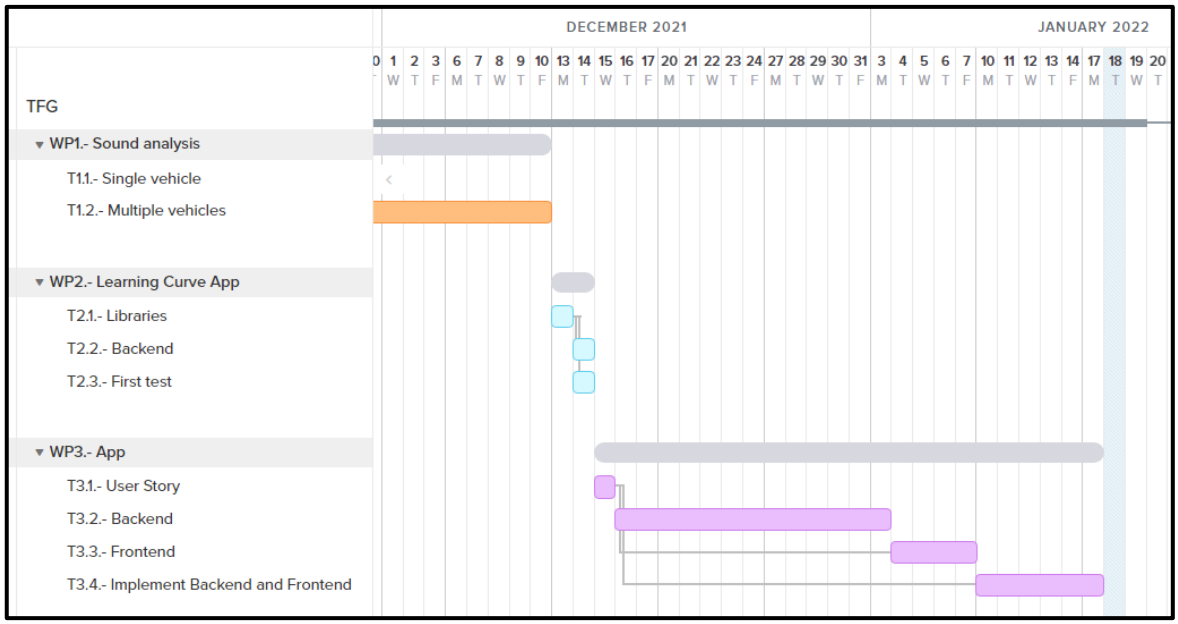

Figure 3 - Gantt Diagram 2

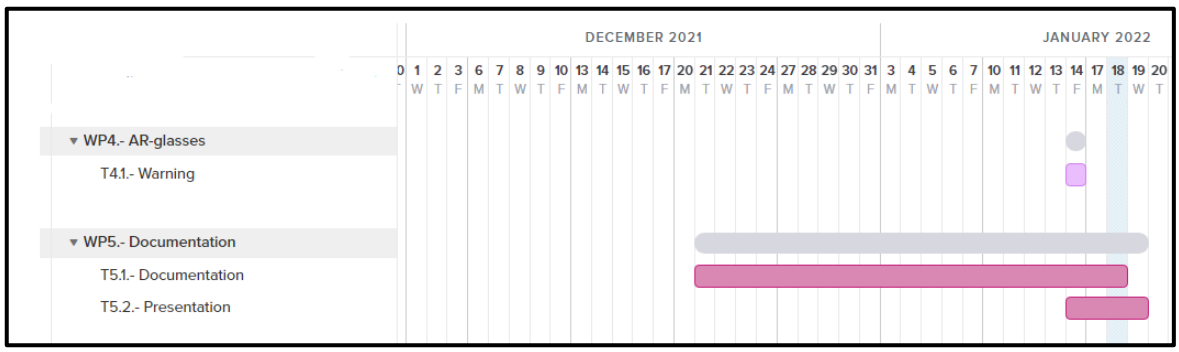

Figure 4 - Gantt Diagram 3

1.5. Deviations and incidences

While working on the project, I have come with multiple incidences that have delayed or forced me to deviate from the initial plan.

- Sound analysis → the sound analysis has taken much more time than initially planned, this is due to the fact that all the conclusions I was getting were not good enough to prove it was possible to detect approaching vehicles with a smartphone's microphone. And as I previously said, if this could not be proved, I could not move to the next work packages.
- PWA → once I started to code the app, I had planned to do it as a native app, but after researching for the best way to access the microphone and stream the audio to a server, I came to the conclusion that I should use MediaStream Recording's API which cannot run as a native app. In the end, instead of just creating an APK, I turned the app to a Progressive Web App and hosted it in Firebase.
  This could not work as a permanent solution, because Firebase doesn't allow HTTP requests. As the server is running as a localhost, I tried to turn the requests to HTTPS but it wasn't enough, because no self-signed certificates are allowed.

- AR-glasses → due to time and COVID-19 restraints, I could not perform the last part of the project, which is displaying a visual warning through the AR-glasses. Instead, what I have done is explain how I would have worked. To prove the app's performance then, you can read the state of the warning (True or False) through a console.log.

2. State of the art of the technology used or applied in this thesis:

2.1. Research done

Previous to this thesis, there have been a few other tries to research and implement a similar project by other people around the world. Most of them had the aim to detect approaching vehicles for common cyclists in order to avoid accidents by having to constantly turn around to check their surroundings. None of them were made for deaf people in mind.

Of these few projects, the most of them were not accurate enough, but there was one in particular very similar to this one that had a great accuracy. [10]

The difference between that project and mine, is that they used machine learning for the sound analysis, and only used recordings of one same and quiet environment.

In my project, in contrast, I have used different environments with loud and constant background noise, which lowers my accuracy of detection but makes the result closer to the one you would have in a real use.

2.2. Technology used

In order to achieve the result of this thesis, I have used different technologies and mathematics:

FFT:

The Fast Fourier Transformation is one of the most important tools used to analyse audio and other media. It converts the data into spectral components, which will help us see the predominant frequencies found in the studied data.

This will be used for the sound analysis so we can try to detect the vehicles focusing on the frequencies they tend to be present in.

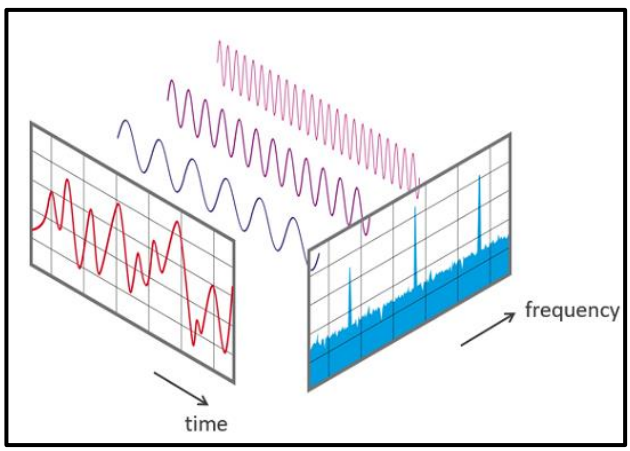

Figure 5 - Fast Fourier Transform: *View of a signal in the time and frequency domain*
[https://www.nti-audio.com/en/support/know-how/fast-fourier-transform-fft]

Percentile:

The percentile is a mathematics tool used to calculate the value at which point a certain percentage of data falls below.

This will be used to compute the threshold that will determine whether there is a vehicle approaching or not.

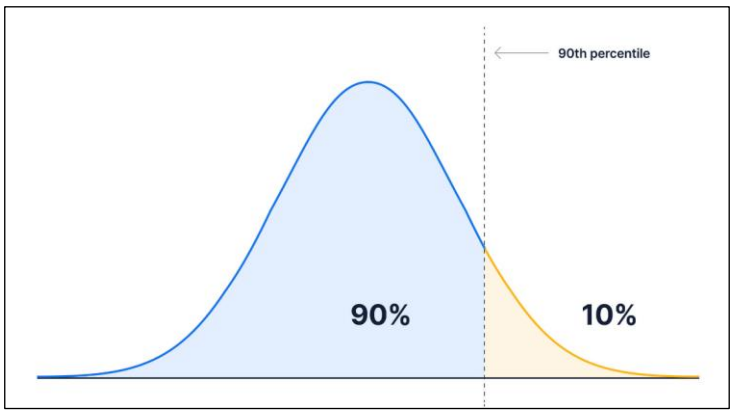

Figure 6 – Percentile: A normal distribution with the 90th percentile depicted [https://blog.timescale.com/blog/how-percentile-approximation-works-and-why-its-more-useful-than-averages/]

Ionic:

Ionic is a framework for app developing that allows you to build cross-platform web mobile and desktop applications. It offers a great selection of components, gestures and tools that will make their implementation fast and easy.

This is the framework that has been used to build the frontend of the app.

Flask:

Flask is also a framework for app developing but written in Python. This has been used to create the server that analyses the audio that the frontend streams.

MediaStream Recording API:

The MediaStream API is related to WebRTC. This has been used for the stream of audio. It was implemented with the frontend, so it gets access to the device's microphone and continuously streams the audio that will be sent to the backend for analysing.

3. Methodology / project development:

This project can be divided in four parts:

3.1. Sound analysis

The first thing I did was obtain several recordings on the street of cars approaching so I could extract the audio and analyse it. This was done in different ways each time:

I started by recording single cars and trying to see if it was possible to detect it or not. Then I moved to recording with more than one car, around 1 minute each recording. This brought to my attention a few problems I was facing:

- Setting apart sounds that come from the front and back → The recordings were made while standing still on the side of the road perpendicularly, so I could not see any difference between the cars approaching from the front or the back.
- Slow vehicles → Another incidence were slow vehicles (especially if they were slowing down), because even though they are technically approaching, the sound analysis does not detect an increase on the studied frequencies, meaning the warning does not show up.
- Wind, pedestrians and other noises → The current sound analysis has a lack in precision, mainly due to unwanted noises picked in the recording, being it wind, pedestrians talking near the smartphone microphone or other loud noises you could find on any street. The aim is to reduce these noises as much as possible to make the warning more accurate.

The final batch of videos were recorded taking in account the past mistakes and trying to find a solution. This time they were longer (more than 5 minutes each). In order to work around the past problems I did the following:

- Setting apart sounds that come from the front and back → This time the recordings were made while standing parallel to the road, facing what is supposed to be the back of the cyclist. This was supposed to create a difference between the sound coming from the back vs from the front, because now the body would be acting as a “barrier” on one side. Unfortunately this did not occur. There was no noticeable difference despite the effort. After some thought, we came to the conclusion that those cases would not cause too much trouble in the end result, because what is more important is that the vehicles from the back are picked up, which it is the case.

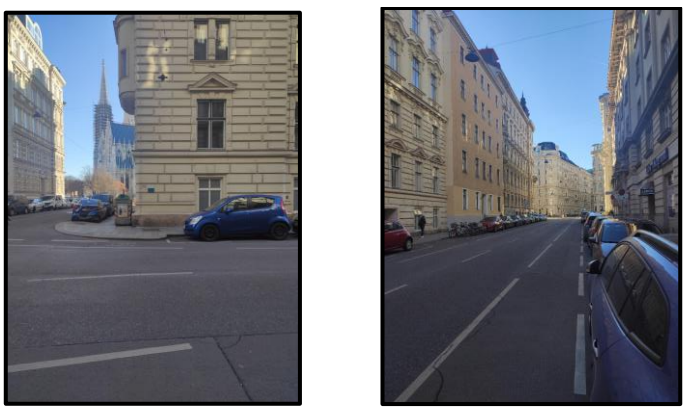

Figure 7 - Camera angles: Difference of camera angles of the first vs final batch of the recordings

- Slow vehicles, wind pedestrians and other noises → in order to solve this problem, I tried to use recordings of different environments, meaning there would be different kind and levels of background noise in each so I could study them more. After different approaches of trying to find the best threshold to use that would determine if there is any vehicle approaching or not, I ended up using a variable threshold by analysing only small fragments of the audio as a moving window. This way, the fake warnings due to external noise were substantially lowered.

Once I had the videos, I extracted the audio and used Matlab to analyse it and find a way to detect the approaching vehicles. The output of Matlab is an array that tells you for every second if the warning should be displayed or not.

In order to see if the warnings were accurate or not, I used Adobe Premiere Pro to display a warning over the video of the analysed audio.

The way the analysis works is the following:

- I get the audio of the recording, and for every second, I compute the FFT. I then focus on the values of the frequencies where we can detect vehicles (between $f_1 = 10^2 Hz$ and $f_2 = 10^3 Hz$), and copy the maximum value between those frequencies to the array called F.
- I now have to find a threshold that will be used to determine if there is a car approaching or not. The best thing to do is use a variable threshold, so if you move from a very quiet street to a busier one, you don't get affected by the previous one.
    - I use the first three seconds of the recording to get a fake threshold, which will be the max value of F from t=1s to t=3s. This fake threshold will be used until t=9s.
    - At t=10s I start to compute the fake threshold as a moving window of the last 9 seconds. The value of the threshold comes from computing the percentile 70 of that moving window. There were a lot of other approaches to finding this value, like using the mean or median, but none of those came as close to the real output as the percentile 70 did.
    - At the same time I am computing the fake threshold, I am also getting the real one. In order to avoid fake warnings or missing vehicles when we have more than 9s with either a very quiet or very busy environment, we have to filter the fake threshold in order to get the real one. This is done this way:
        - If the fake threshold is lower than the median of fake thresholds from the last minute (or until that point if it still hasn't come to a minute), I use that median for the real one.
        - If the fake threshold is higher than the double of the median of fake thresholds from the last minute (or until that point if it still hasn't come to a minute), I use that value for the real one.
        - If the fake threshold is in between those two values, I use it for the real one.
- The warning will be issued whenever the F value goes over the threshold plus an extra one second of margin.

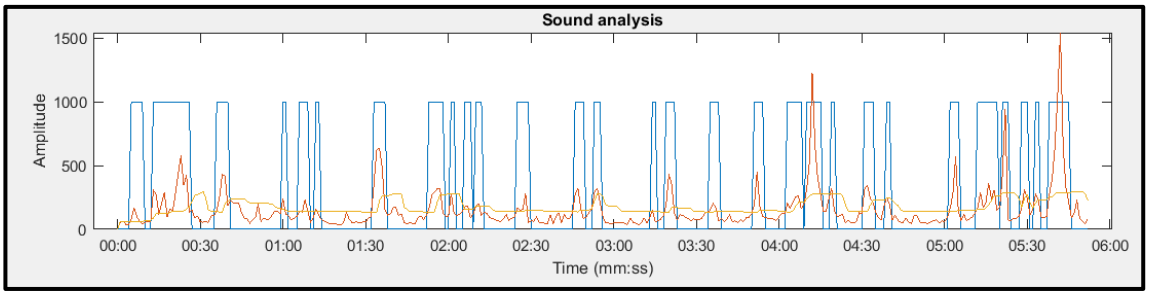

Figure 8 - Sound Analysis: Representation of the final result of the Sound analysis. Legend: Blue → Warning Red→ Maximum frequency of the FFT between f1 and f2 Yellow→ variable threshold

3.2. Frontend of the app

For the frontend of the app, I have used Adobe Photoshop CC to design the User Story and Ionic Angular to code it.

The app consists of two main pages: Home Page and Start Page.

Home page:

The home page consists of a background, a button and a bottom drawer.

The bottom drawer was created as a component. It contains the Instructions to use the app. You can either slide it up or click the toggle and it will slide along until you can read everything. This was done with the GestureController. The background will then darken and fade a little bit.

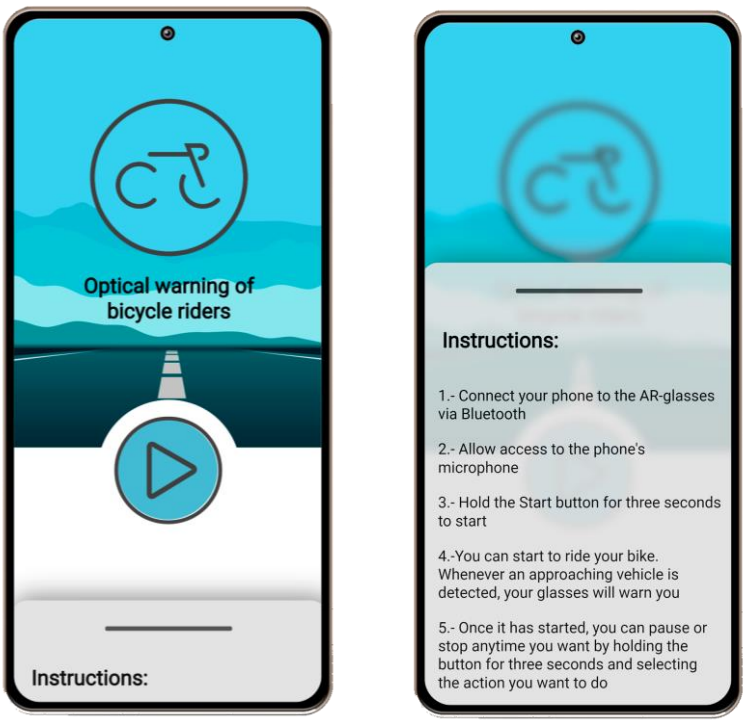

Figure 9 - Home Page and Instruction Page [5][14]

The button was also created as a component so it can be reused for the Start Page. The name of the page we are in, is passed as a parameter to the component so it knows which icon to show and which actions to perform.

The GestureController was also used for the button to create the long press. The user is supposed to hold the button during 3s in order to start, so with the gesture controller we can add actions (like a small vibration) to the start and the end of the event. When a user starts to press the button, the Boolean pressing turns true and we call the functions starting() and audioRecording().

Starting() gets the time of when it gets called and once three seconds have passed it redirects you to Start Page.

AudioRecording() is the one in charge to start Recording the audio from the microphone. It uses the MediaStream Recording API to call getUserMedia(). This returns a stream of audio that stops either when the user stops pressing the button or when the three seconds have passed. Once the data is available, we convert it to a blob and send it as a FormData through a POST request to the server, where it will be analysed and used to start a threshold.

While the button is pressed, a blue ring appears around it which will keep growing clockwise as the time passes until the desired three seconds. This was done with HTML and CSS. I created a blue circle with four white squares on top, stacked them as 2x2 and rounded the external corners, so the white squares represent the quadrants of the blue circle, which is hidden behind. Once the button is pressed, in the correct order and time, each white square is skewed slowly revealing the circle underneath as a progress mark. If the user stops pressing the button, the progress circle will disappear and we will be back to the normal Home Page.

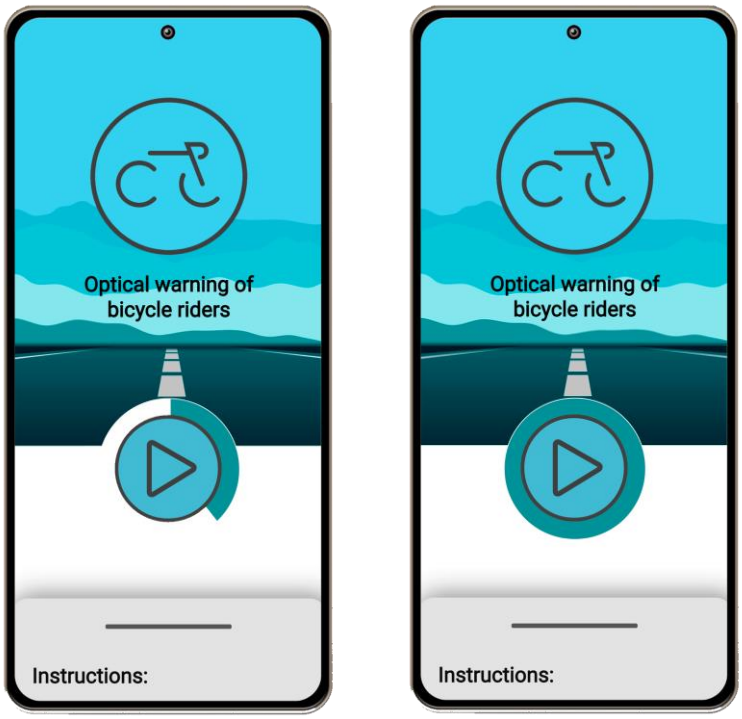

Figure 10 - Starting Page

Start page:

The start page visually is the same as the Home page but with a different icon on the button and without the bottom box.

As soon as the page appears, the getUserMedia() gets called every second, so there is a constant stream sending POST requests to the server the same way it was done while the button was being pressed before. This time though, we get a response from the server with a Boolean that will indicate if the warning should be on (True) or not (False).

There is a timer on the bottom of the page that indicates for how long have you had the app analysing the audio (format hh:mm:ss).

Once the user wants to stop, they can press the pause button for 3 seconds which will automatically stop the audio stream. The long press in this page is useful to prevent stopping the app by error.

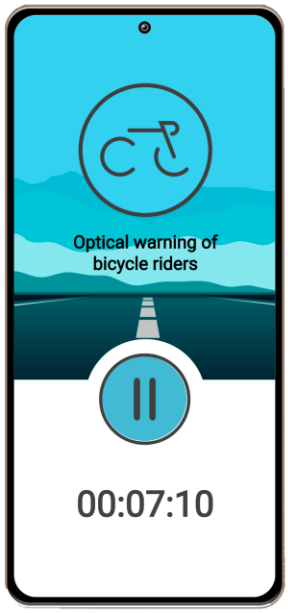

Figure 11 - Start Page

In this page the button acts similar as before, but it does not record anything. If the three seconds pass and the progress gets completed, a warning will pop asking you if you either want to pause or stop.

If you press pause, you will be kept in the start page but with a couple differences: the button looks and works the same as the start page, and the timer and streaming are not running. If you press the button to start again, those first 3 seconds will be sent to the server for a new threshold and the streaming will resume as normal.

If you press stop in the alert, you will be redirected to the home page so you can start the whole process again.

Both the actions you can do from the alert, will warn the server of your choice with a POST request.

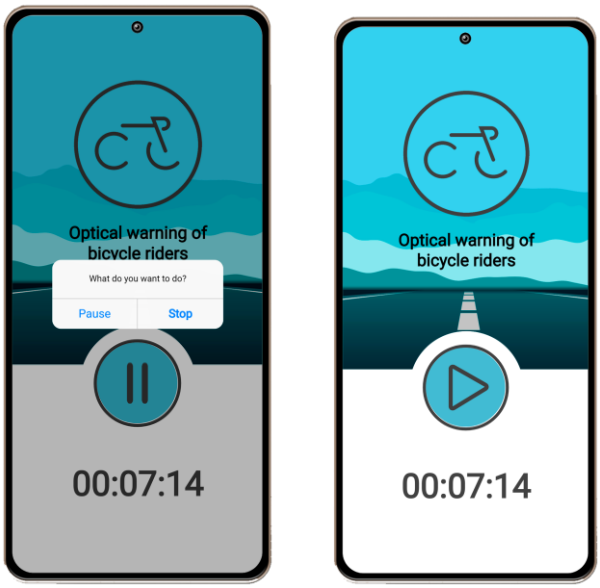

Figure 12 - Warning and Resume Page

3.3. Backend of the app

For the backend of the app, I have used Flask (Python) to write the server that will run as localhost on a computer.

The server only takes the method POST and has a parameter on its route. That parameter is the status, used for the frontend to warn the backend if the streaming is playing, paused or stopped.

If the streaming is playing, the server receives the blob of the audio as a FormData, so it saves it as a temp.wav file. As it is not a pure wav file, it cannot be directly read, so I use AudioSegment to transform that file to a real wav that will be read as audio. Once I have the audio, as it is only of one second, I can analyse it the same way I did in Matlab. Finally it returns a json with the warning as a Boolean.

In case the user stops or pauses the stream, the server will get it from the status and will reset the local variables that is using to calculate the threshold.

3.4. AR-glasses

Like I said before, It was not possible to actually implement this part, but I will explain how it would have worked.

The glasses (Microsoft Hololens 1) are supposed to be connected via Bluetooth to the smartphone. This way, whenever a warning is issued, the glasses can display it, so the cyclist knows there is a vehicle approaching from behind.

The warning would be enough to alert the user but not so big and bright that it would block their view of the road or suppose a distraction, especially if we keep in mind that the field of view of the glasses is very narrow. This could be done by displaying an intermittent colour around the side borders of the view like shown on the image.

Figure 13 - Warning example: Example of what the user would see through the AR-glasses whenever a warning is issued

Given the fact that it is AR technology, a normal object would stay static so when the user moves their head, they could lose visibility of said object. In order to avoid that issue, it is possible to attach a script associated to that object so that it follows your position every time your head moves. That script would be written in C#.

Another issue we can encounter is the contrast of the warning. If it does not have the correct contrast, it could not be visible enough for the user depending on the environment of use (it would not be the same to use the glasses on a very sunny place vs a dark one). One solution to this is the use of a protective shield over the glasses that would enhance the contrast of the display.

In order to send the warning data to the AR-glasses, the Bluetooth LE advertisements will be used. The glasses will be constantly scanning, and the smartphone will send a BLE advertisement whenever a warning must be displayed. This will allow the Hololens to know when should the red frame be visible and when not.

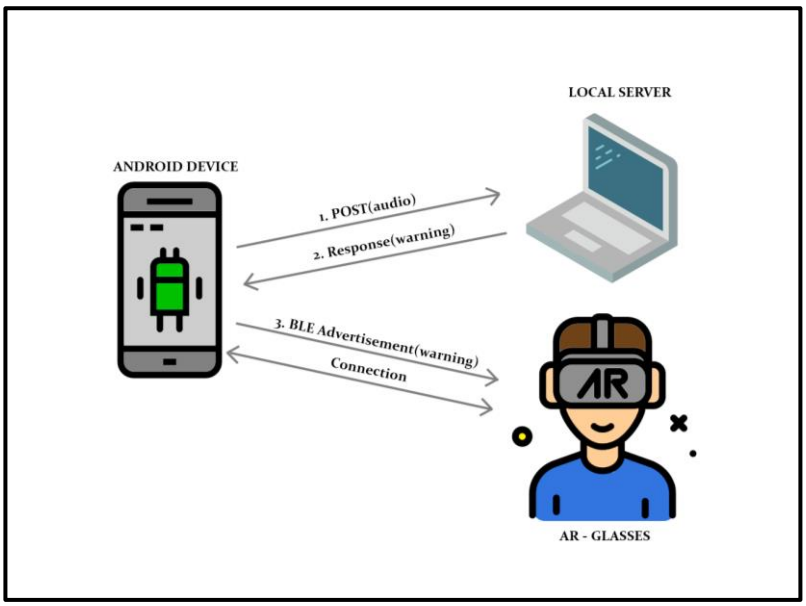

Figure 14 – Diagram: Diagram of the connections between the devices [3] [11] [12]

4. Results

From the recording of multiple single vehicles, once I computed the FFT, I could clearly see that the frequencies where you can find a lot of presence is between $f_1 = 10^2 Hz$ and $f_2 = 10^3 Hz$, so this is where I focus the rest of the analysis in.

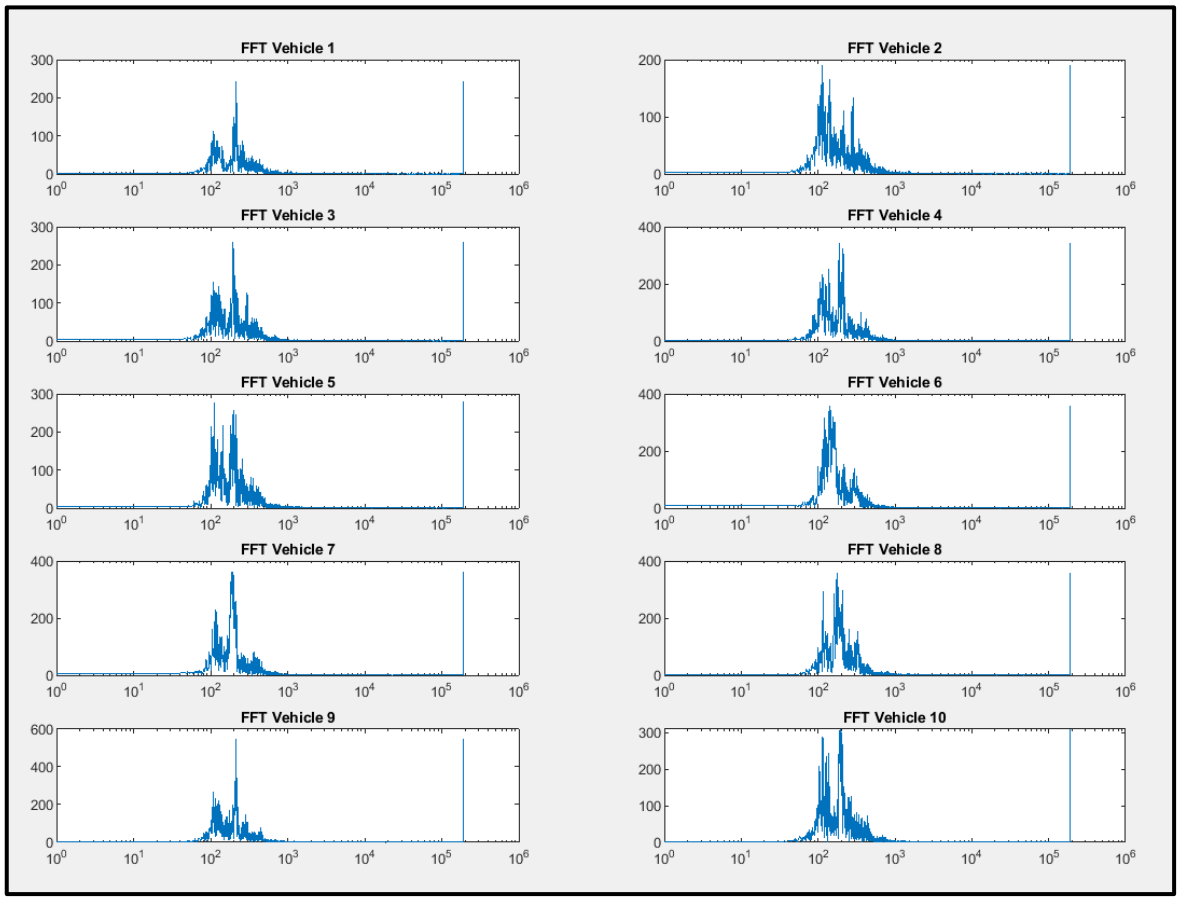

Figure 15 - FFT Vehicle: FFT analysis of 10 different vehicle recordings

On the final version of the Matlab code, I analysed several recordings of various streets with different background noises. Some were quieter, some were louder and some had construction work going on around.

After running each recording through Matlab and displaying the suggested warning over the video, I could then see if the warning was accurate and properly detected the vehicles approaching.

For each recording I counted:

- Number of vehicles that were detected
- Number of vehicles that were not detected
- Number of fake detections

With these values, I was able to calculate the probabilities of a fake positive (warning displayed but there were no vehicles) and fake negatives (vehicle not detected).

These are the results of some of the recordings:

	Warning	No warning
Vehicle	27	4
No vehicle	3	

Table 2- Recording with quite a bit of wind noise: Number of vehicles detected, undetected and fake warnings

	Warning	No warning
Vehicle	30	2
No vehicle	8	

Table 3 - Recording in a quiet street with slow vehicles Number of vehicles detected, undetected and fake warnings

	Warning	No warning
Vehicle	37	12
No vehicle	7	

Table 4 – Recording in a busy street with loud construction work on the background: Number of vehicles detected, undetected and fake warnings

	Warning	No warning
Vehicle	36	3
No vehicle	2	

Table 5 - Recording on a normal street: Number of vehicles detected, undetected and fake warnings

Overall, the probability of a fake positive and a fake negative is both 13%. It is not a perfect detection, but we can confirm that it is possible to detect approaching vehicles from a smartphone's microphone.

The first 10 seconds of the recording are always the least trustworthy, but with the app implementation it does not suppose a problem, because the first three seconds are recorded while the user is pressing the start button, and during the next ones they are supposed to put their phone on their pocket/backpack before they start cycling.

In the results of these recordings, we noticed that for the use of this app they were not that bad. The most important thing is to warn the cyclists of upcoming vehicles, so we need to have the least amount of fake negatives as possible. Overall those fake negatives were mostly produced by very slow cars (which would most likely not startle the cyclist) or by loud construction work on the background (which is not the case for most

streets, and whenever it is, it would not last long, because the cyclist would pass those by and continue with normal streets).

The fake positives are not ideal, but would also not suppose a danger to the cyclist, it would only mean that they would expect a vehicle when there was not.

5. Budget

Taking into account this project is just a proof of concept, I will budget it as so.

Design and developing:

There was a total of 479 hours that were put into work for this project, so it will be evaluated at cost of junior engineer.

ROLE	Hours worked	Cost x hour	Gross Salary	Social Security charges (30%)	TOTAL SALARY
Junior Engineer	479 h	15 €/hour	7185 €	2155,5 €	9340,5 €

Table 6 - Role Costs: Salary of a Junior Engineer for the amount of work done

Now I will break it down by the previously defined work packages and tasks:

WP #	Task #	Title of the task	Hours x cost
1	1	Single vehicle	146,5 hours x 15 €/hour = 2197,5 €
	2	Multiple vehicles	71 hours x 15 €/hour = 1065 €
2	1	Libraries	7 hours x 15 €/hour = 105 €
	2	Backend	3 hours x 15 €/hour = 45 €
	3	First test	3 hours x 15 €/hour = 45 €
3	1	User Story	10,5 hours x 15 €/hour = 157,5 €
	2	Frontend	96,5 hours x 15 €/hour = 1447,5 €
	3	Backend	23 hours x 15 €/hour = 345 €
	4	Implement Backend and Frontend	33 hours x 15 €/hour = 495 €
4	1	Warning	10 hours x 15 €/hour = 150 €
5	1	Documentation	54,5 hours x 15 €/hour = 817,5 €
	2	Presentation	21 hours x 15 €/hour = 315 €

Table 7 - Tasks Costs: Costs broken down into tasks

Equipment

In order to do this project, I have needed a computer and a smartphone. Since in the end I did not actually do the implementation of the AR-glasses, I will not count them.

If I take into account the amortization of this equipment, I can calculate it by assuming 15% of its useful life (5 years for the computer and 3 for the smartphone) and see how much it would be for the duration of this project.

$$Residual\ value = Coefficient * Price$$

$$Annual\ deprecation = \frac{Price - Residual\ value}{Useful\ life}$$

$$Amortization\ for\ x\ hours = \frac{Annual\ deprecation}{8760\ hours/year} * \#\ hours$$

Equipment	Price	Useful life	Coefficient	Residual value	Annual Depreciation	Amortization for 479 hours
Laptop	1000 €	5 years	15%	150 €	170 €	9,30 €
Smartphone	300 €	3 years	15%	45 €	85 €	4,65 €
					Total amortization for 479 hours	**13,95 €**

Table 8 - Equipment Costs: Amortization of the equipment used

Licenses:

Some of the programs I have used to work can require a paid subscription for a license, but in all the cases I managed to get by with the free version, or in Matlab with the free license that the university provides for its students. Seeing this, the cost of licenses for this project is of **0 €**.

TOTAL:

If I add the total amount of costs that we have considered before, we can get the total cost of this project:

Salary	9340,5 €
Amortization	13,95 €
Licenses	0 €
TOTAL	**9354 €**

Table 9 - Total Costs

6. Environment Impact

For the environmental impact study, I will measure the overall consumption during the developing of the whole project.

Since I have only used a laptop and a smartphone, these are the only aspects apart from the electricity usage that have left an impact. There is no transport of commuting to keep in mind, because everything has been developed from my room.

Laptop usage:

In this category I will include the coding, designing, writing documentation and communication with the supervisors.

If the computer is 35 W, I include 10W for network and data centers:

$$45W * \frac{479h}{1000} = 21{,}55\ kWh$$

Smartphone usage:

In this category I include the testing of the app and recording of the videos for the sound analysis.

If the phone is 3W and I include 10W for network and data centers:

$$13W * \frac{20h}{1000} = 0{,}26kWh$$

Electricity usage:

Since the internet use has already been considered with the previous categories, this will only take in mind the power usage.

If the room has only 2 lightbulbs of 60W each:

$$2\ lightbulbs * \frac{60W}{lightbulb} * \frac{479h}{1000} = 57{,}48kWh$$

TOTAL:

Laptop usage	21,55 kWh
Smartphone usage	0,26 kWh
Electricity usage	57,48 kWh
TOTAL	79,29 kWh

Table 10 - Environmental Impact

7. Conclusions and future development:

As previously said on the Results section, this project has 87% accuracy when detecting approaching vehicles by analysing the audio through a smartphone's microphone.

It is not a perfect result but it is quite good and could make a good extra help to those deaf cyclist who constantly get startled by the sudden presence of vehicles around them.

In order to be true to the aim of this project, in a possible future development, it would be very important to warn its users that this app is not 100% accurate and they should still be very careful when riding a bike. This app is meant to be a helpful tool to try to help the cyclist put more focus on the road in front of them, it should not be blindly trusted because there can always be some cases where there is a vehicle approaching and it won't be detected.

Given the previous warning, I think this app could suppose a big help especially for people who are hard of hearing rather than completely deaf. This is due to the fact that they still have some residue of hearing that can help them be more aware of that 13% of fake warnings that can occur while still having an assistance that they can rely on for extra awareness of their surroundings.

In a future this could be developed as an actual product, with the implementation of the AR-glasses being executed and bringing to life this project.

If the glasses suppose a setback, considering they can be inaccessible to most people because of their price, a workaround could be made by using a smart band with haptics, so that whenever a vehicle is detected, the user can feel a small vibration on their wrists instead of seeing an optical warning.

Bibliography:

[1] BluetoothLeAdvertiser. Android for Developers. https://developer.android.com/reference/android/bluetooth/le/BluetoothLeAdvertiser

[2] C.Chen."Rear Approaching Vehicle Detection with Microphone". B. thesis, Electrical Engineering with Emphasis on Wireless System Design, Halmstad University, Halmstad, Sweeden, 2013.

[3] D. Dan, AR-glasses icon https://www.flaticon.com/free-icon/ar-glasses_4636048?term=ar%20glasses&page=1&position=19&page=1&position=19&related_id=4636048&origin=search

[4] F. Meucci, L. Pierucci, E. Del Re, L. Lastrucci and P. Desii, "A real-time siren detector to improve safety of guide in traffic environment," *2008 16th European Signal Processing Conference*, 2008, pp. 1-5.

[5] Freepik, Background Vector, https://www.freepik.com/free-vector/travel-background-realistic-style_1919861.htm#query=road&position=45&from_view=search

[6] Kohn, D. How percentile approximation works (and why it's more useful than averages). Timescale. 2021 ,https://blog.timescale.com/blog/how-percentile-approximation-works-and-why-its-more-useful-than-averages/

[7] Media Capture and Streams API (Media Stream). 2021. Developer Mozilla. https://developer.mozilla.org/en-US/docs/Web/API/Media_Streams_API

[8] NTi "Fast Fourier Transformation FFT – Basics". NTi Audio. https://www.nti-audio.com/en/support/know-how/fast-fourier-transform-fft

[9] S. Glen. "Percentiles, Percentile Rank & Percentile Range: Definition & Examples" From StatisticsHowTo.com: Elementary Statistics for the rest of us! https://www.statisticshowto.com/probability-and-statistics/percentiles-rank-range/

[10] S. Kawanaka, Y. Kashimoto, A. Firouzian, Y. Arakawa, P. Pulli and K. Yasumoto, "Approaching vehicle detection method with acoustic analysis using smartphone for elderly bicycle driver," 2017 Tenth International Conference on Mobile Computing and Ubiquitous Network (ICMU), 2017, pp. 1-6, doi: 10.23919/ICMU.2017.8330069.

[11] Smalllikeart, Android device icon, https://www.flaticon.com/free-icon/android_886580?term=android%20phone&page=1&position=6&page=1&position=6&related_id=886580&origin=search

[12] Smashicons, Laptop icon, https://www.flaticon.com/free-icon/laptop_742201?term=laptop%20server&page=1&position=1&page=1&position=1&related_id=742201&origin=search

[13] UI Components. Ionic Framework. https://ionicframework.com/docs/components

[14] Vintiset, 36 logotipos de bicicletas, 2016, https://vinti7.com/36-logotipos-de-bicicletas/

Appendices :

FFT Analysis:

These are various FFT Graphics of different sounds made to compare the predominant frequencies with the ones of motorized vehicles.

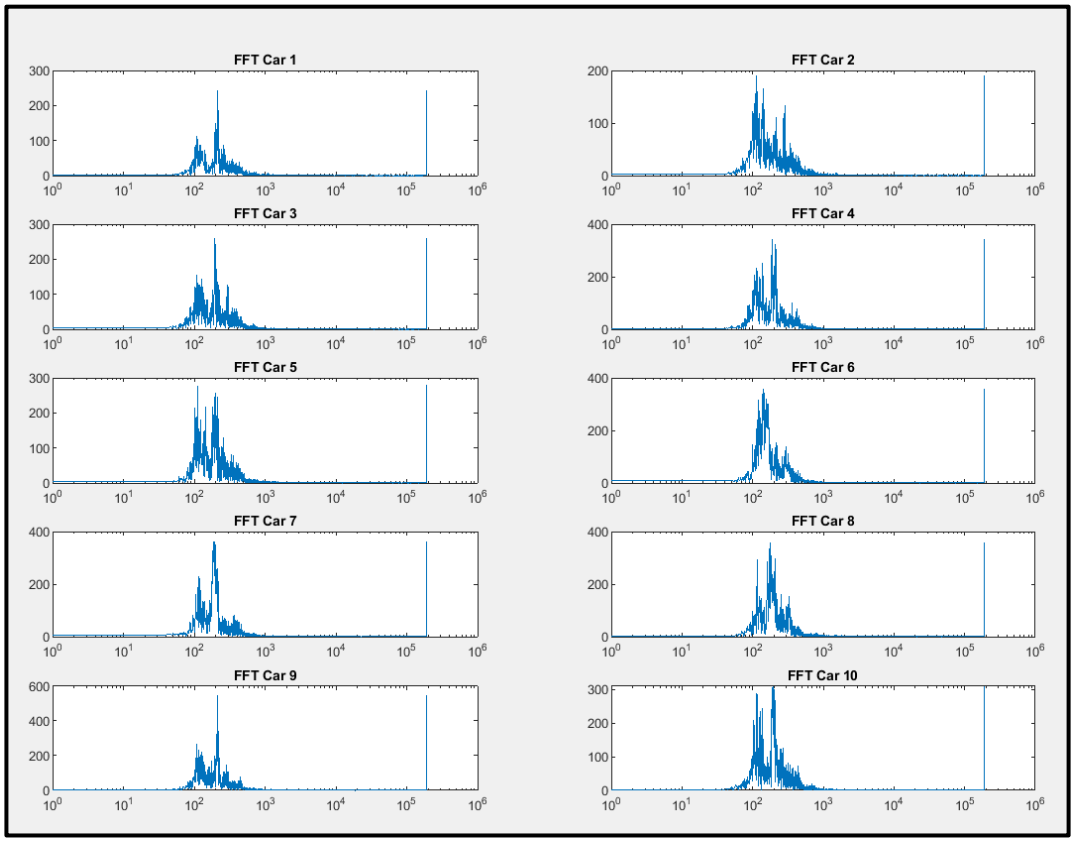

Figure 16 - Car Recordings: FFT analysis of 10 different car recordings

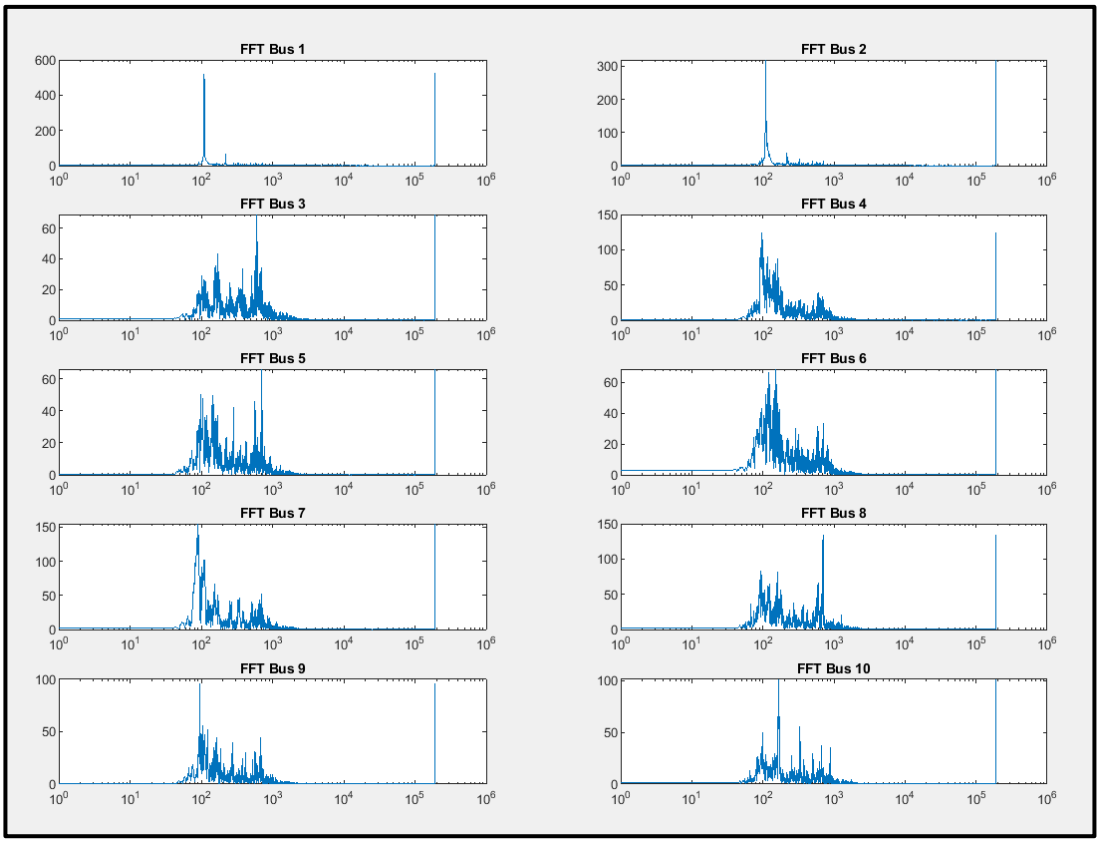

Figure 17 - Bus Recordings: FFT analysis of 10 different bus recordings

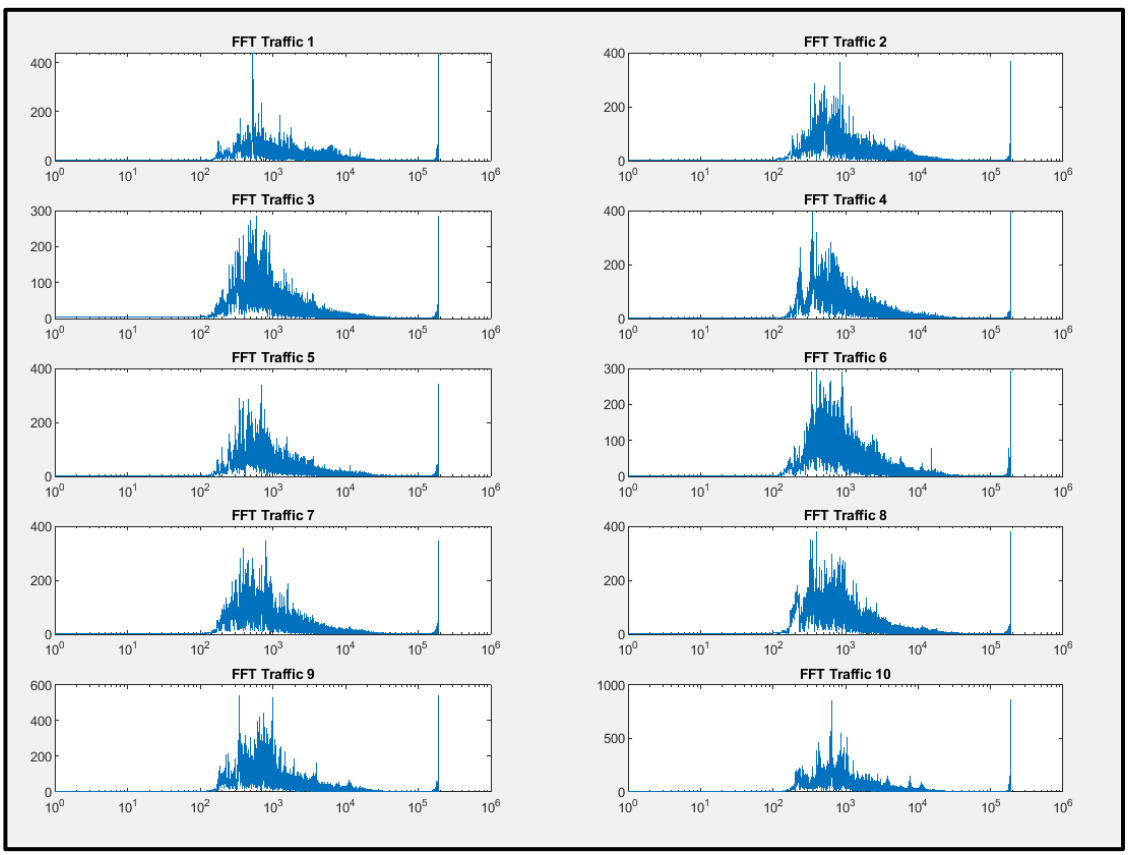

Figure 18 - Traffic Recordings: FFT analysis of 10 different traffic recordings

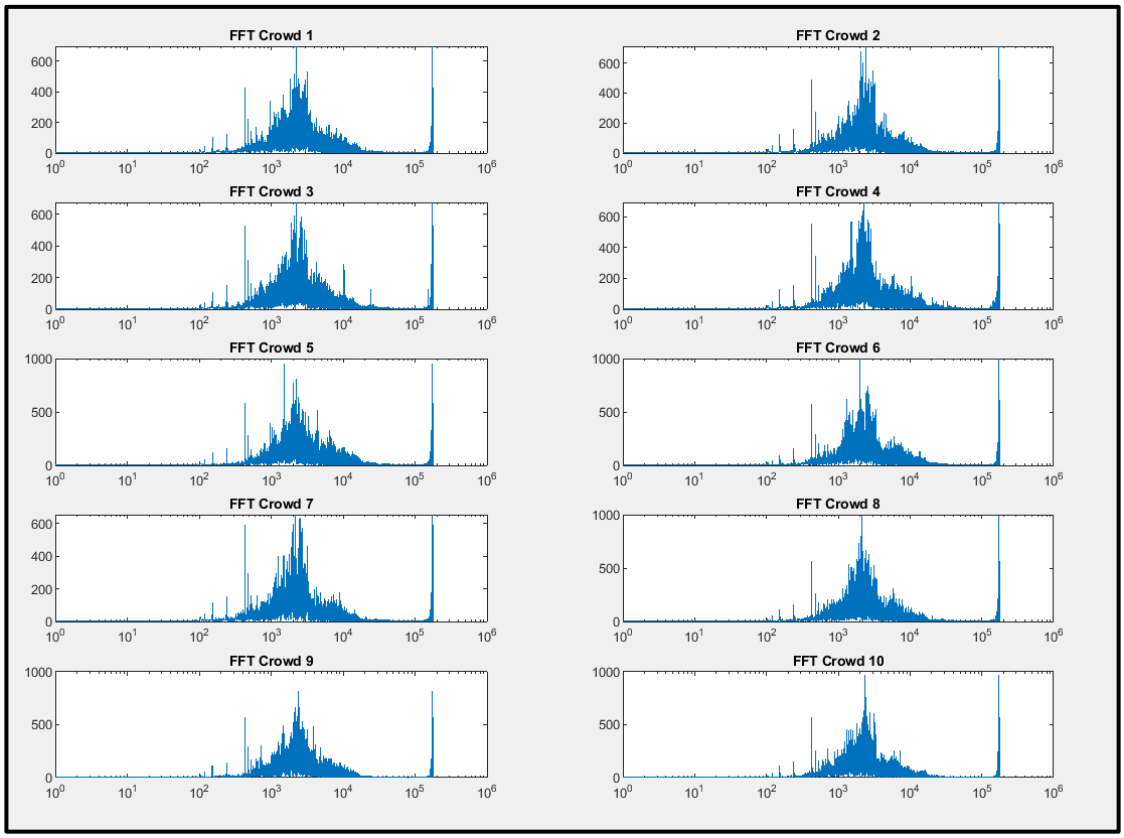

Figure 19 - Crowd Recordings: FFT analysis of 10 different crowd recordings

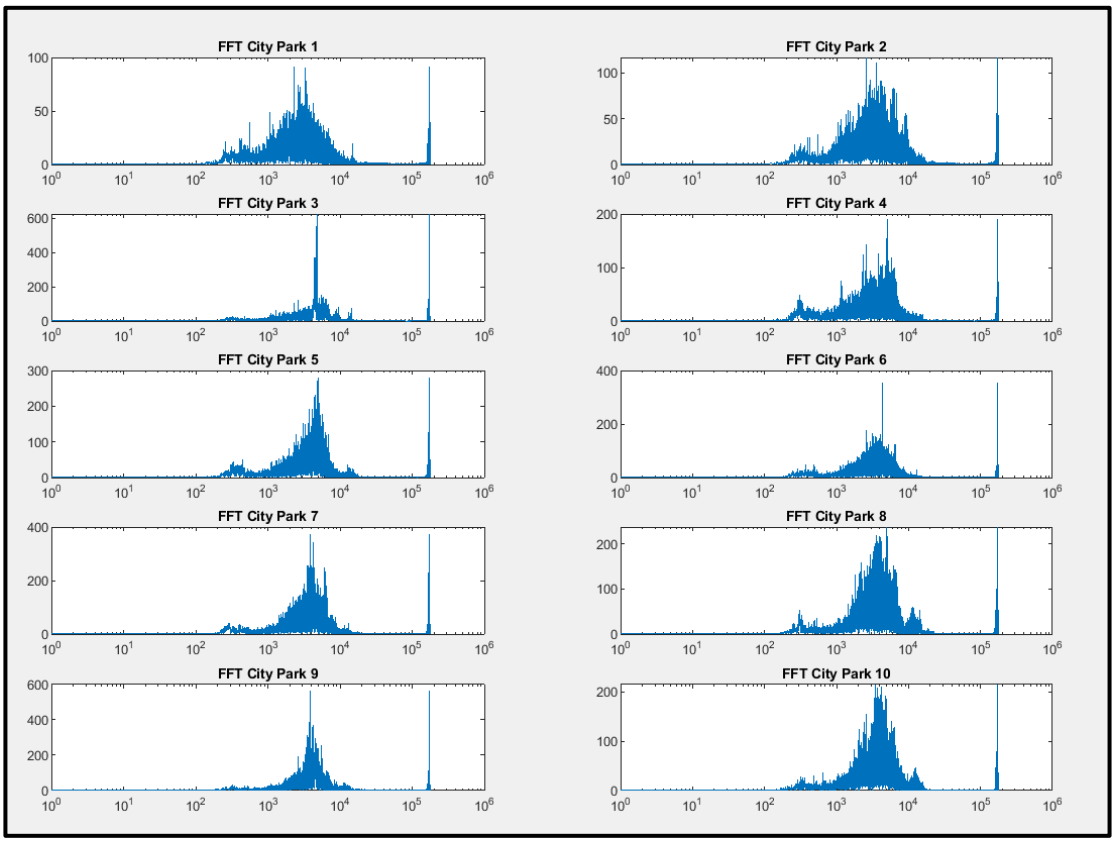

Figure 20 - City Park Recordings: FFT analysis of 10 different city park recordings

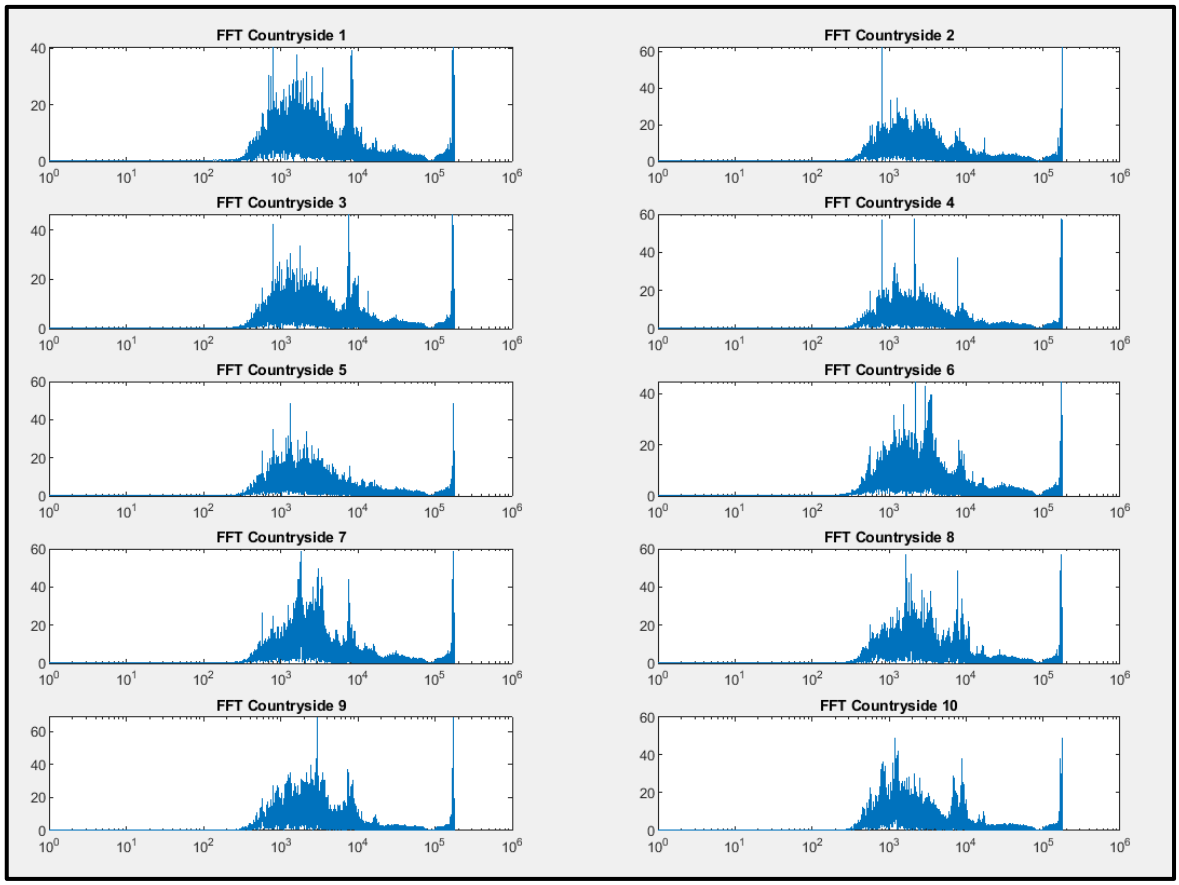

Figure 21 - Countryside Recordings: FFT analysis of 10 different countryside recordings

Sound Analysis:

These are the Matlab plots that I got as a result of the Sound Analysis in the last batch of videos. Specifically, these are the audios that I attached to demonstrate the accuracy of the code.

On these plots, we can see a representation of the audio itself, and the conclusions of the analysis, in which we have the warning (blue), the maximum frequency value between $f_1 = 10^2 Hz$ and $f_2 = 10^3 Hz$ (red) and finally the variable threshold (yellow).

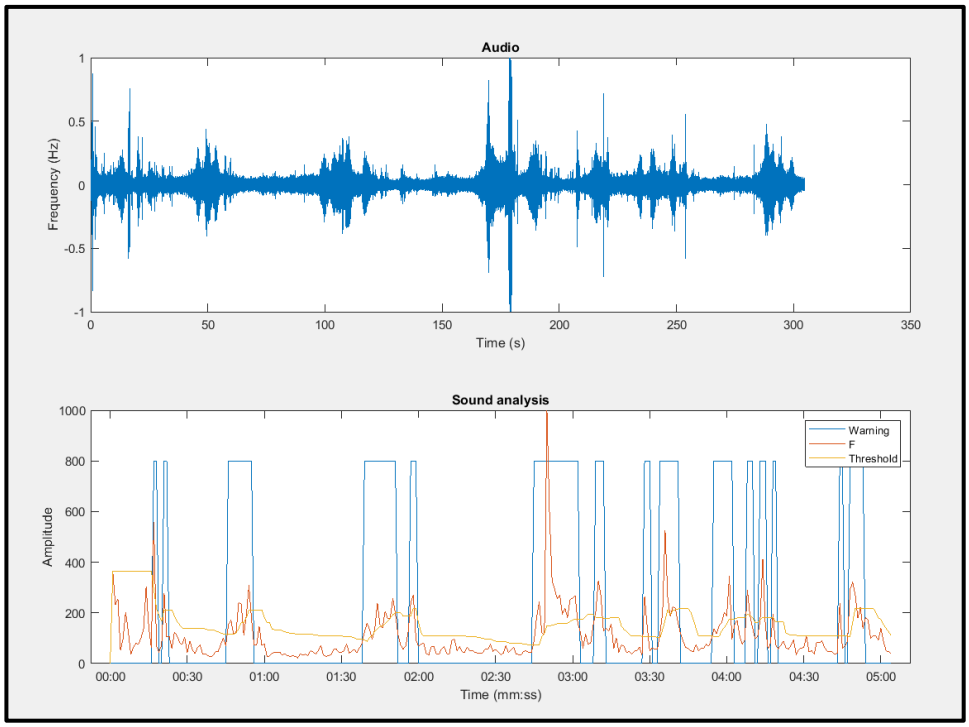

Figure 22 - Sound analysis of Table 2

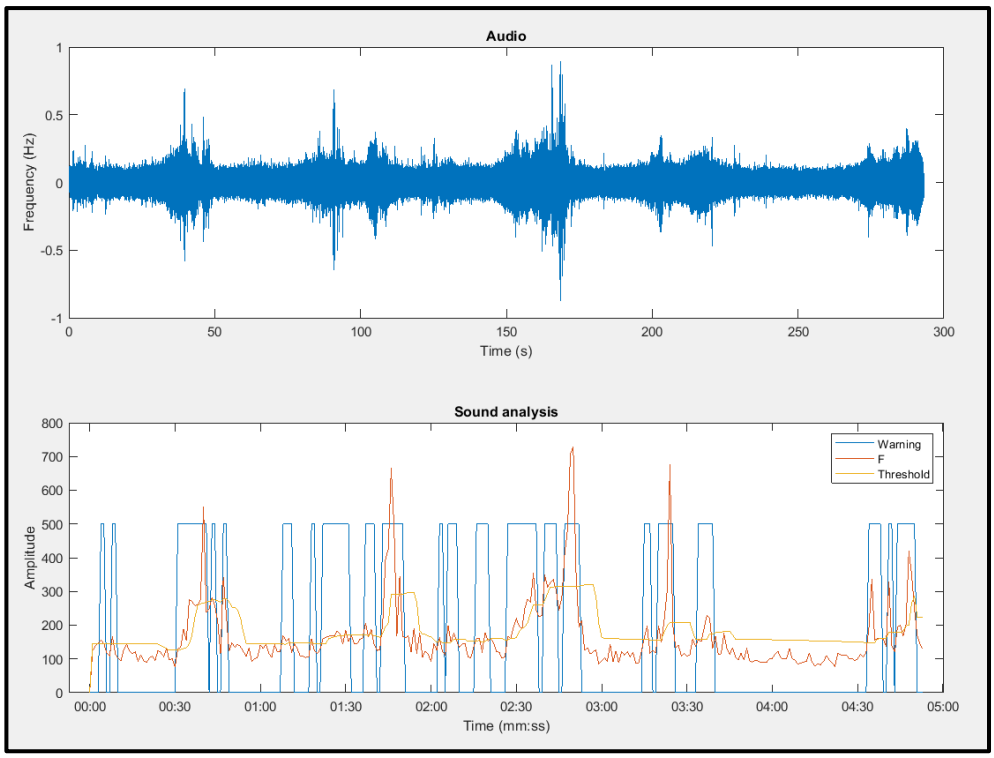

Figure 23 - Sound analysis of Table 3

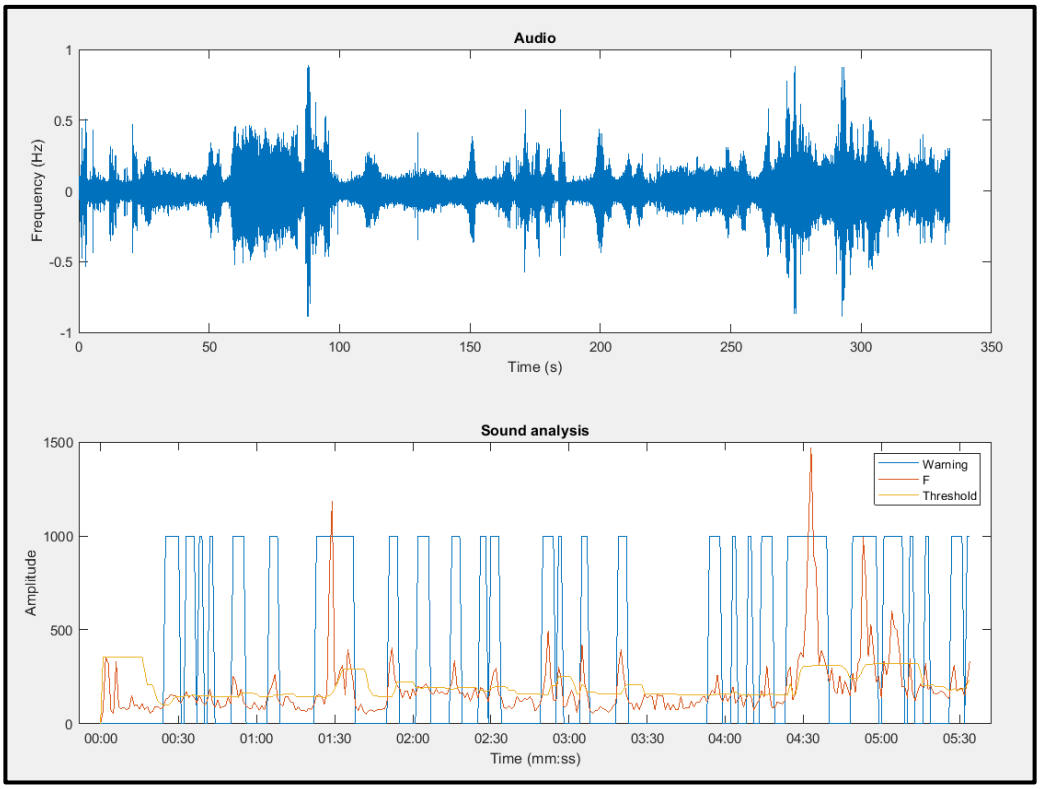

Figure 24 - Sound analysis of Table 4

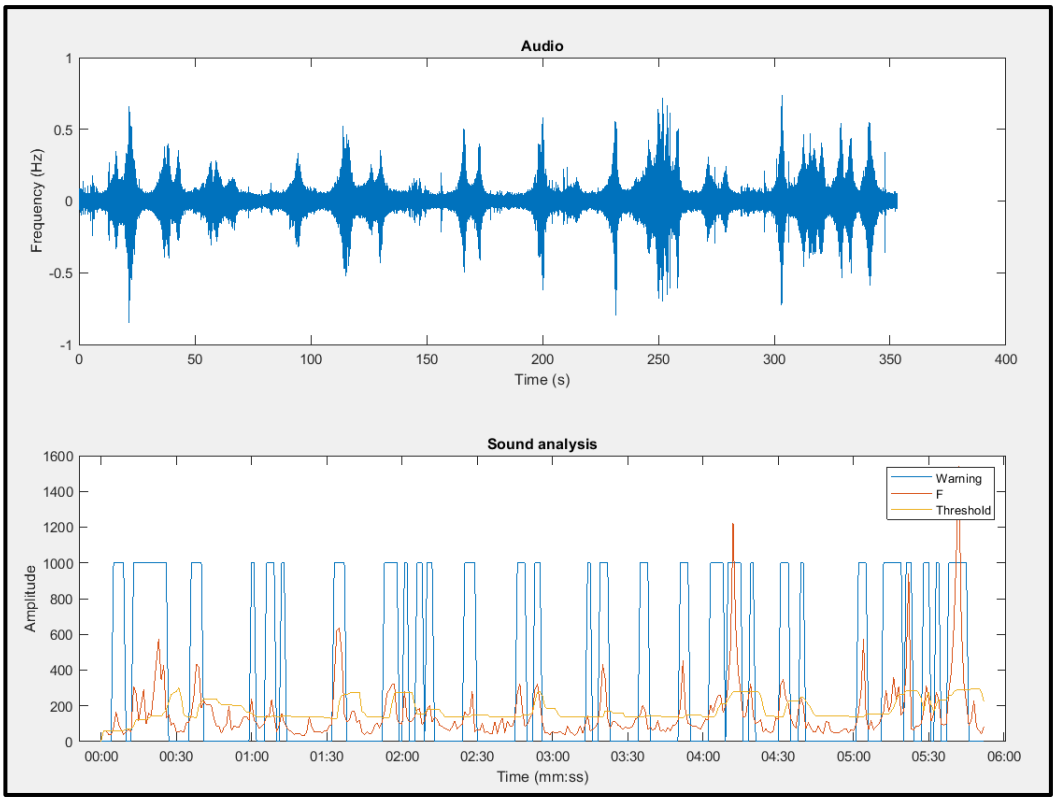

Figure 25 - Sound analysis of Table 5

GitHub Repositories:

The following links take you to my GitHub repositories with the final codes of the Matlab Sound Analysis, the Frontend App and the Backend Server:

https://github.com/albatalaya/BachelorThesis-SoundAnalysis

https://github.com/albatalaya/BachelorThesis-Frontend

https://github.com/albatalaya/BachelorThesis-Backend

PWA link:

In order to be able to test the app on a phone, I created a PWA so that I could also install it on my smartphone.

This link is a version of the original app but without the connection to the server. This was made so other people could see the app without having to keep the server running on my laptop at all times.

I the link is accessed through a computer, it should be on the proper window proportions or through device mode (Developer tools).

https://bicycle-warning.web.app/home

Video demo:

Finally, I attach a link to a video demonstration of how the final app works:

https://youtu.be/GPKONG1mJ90

Glossary

Acronym	Meaning
API	Application Programming Interface
APK	Android Application Package
APP	Application
AR-Glasses	Augmented Reality glasses
BLE	Bluetooth Low Energy
Bluetooth LE	Bluetooth Low Energy
BLOB	Binary Large Object
CSS	Cascading Style Sheets
FFT	Fast Fourier Transform
HTML	Hyper Text Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
PWA	Progressive Web Apps
WebRTC	Web Real-Time Communication

Table 11 – Glossary: A list of all acronyms and the meaning they stand for