Traffic Monitoring Using Low-Cost Doppler Radar

Prepared by:
Mishay Naidoo

Prepared for:
Prof. Amit Mishra
Dept. of Electrical Engineering
University of Cape Town

Submitted to the Department of Electrical Engineering at the University of Cape Town in partial fulfilment of the academic requirements for a Bachelor of Science degree in Mechatronic Engineering

October 30, 2023

Declaration

---

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
2. I have used the IEEE convention for citation and referencing. Each contribution to, and quotation in, this report from the work(s) of other people has been attributed, and has been cited and referenced.
3. This report is my own work.
4. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as their own work or part thereof.

Signature: 
Mishay Naidoo

Date: October 30, 2023

This report is **11793** words

Foreword and Acknowledgments

I would like to express my gratitude to the following people who assisted me in completing this project:

Firstly, to my supervisor Professor Amit Mishra. Thank you for taking time out of your week to assist me with any questions I have had along the way whilst, seemingly, travelling the world at the same time. Also, thank you for proposing such a fun and interesting project topic, I have learned so much from it.

To my co-supervisor Dr Stephen Paine, thank you for putting up with my badgering and for being there in Radar Lab to help me debug my system when, at times, it felt like it would never work.

To Dr Yunus Abdul Gaffar, thank you for taking such an interest in my project and for always checking in on my progress. The resources you gave me were incredibly helpful.

To Brendon Daniels, thank you for helping me with my hardware design in this project. Without your guidance, many a microcontroller and battery circuit would have died at my hand.

To Justin Pead, thank you for answering my endless questions on amplifiers and for teaching me how to read datasheets properly.

To Natasha Soldin, Daniel Jones, David Newton, Hamish Mckenzie and Cameron Wilson, thank you for taking the time out of your day to drive, sometimes quite quickly, past my radar setup. Your driving skills were admirable. To Rory Schram and Tristyn Ferreiro, thank you for ensuring that no accidents happened during some of the faster tests.

Abstract

Traffic monitoring involves counting the number of vehicles on a road at a given point in time whilst recording their velocities. This project aimed to create a traffic-monitoring device using low-cost Doppler radar.

The design process investigated a series of low-cost Doppler radars and ultimately settled on using the CDM324. This radar was integrated with an amplifier module and sampled using a soundcard. The components were all selected with the goal of minimising the cost of the system. Algorithms for collecting and processing the data were subsequently created.

Two systems were developed, one that was designed to test the radar system and another to run this system independently using a Raspberry Pi.

The first system was able to count vehicles and determine their speeds accurately however it had some minor struggles. The system struggled in scenarios where the vehicles were very slow-moving and sometimes overestimated the velocities of the cars by up to 10%. The second system was able to run off of the microcontroller, however housing the Raspberry Pi in the same box as the radar led to interference in the data.

A series of further improvements were then explored with the intention of laying the groundwork for furthering this project.

Contents

List of Figures

List of Listings

List of Tables

List of Acronyms and Abbreviations

ADC Analogue to Digital Converter

CNN Convolutional Neural Network

CW Continuous Wave

DFT Discrete Fourier Transform

EM Electromagnetic

LiDAR Light Detection and Ranging

PCB Printed Circuit Board

RCS Radar Cross Section

RF Radio Frequency

SNR Signal to Noise Ratio

STFT Short Time Fourier Transform

UART Universal Asynchronous Receiver/Transmitter

USB Universal Serial Bus

Chapter 1

Introduction

This report documents an investigation into designing and implementing a traffic monitoring system using Doppler radar. This process involved investigating existing literature on traffic monitoring before selecting components and designing a prototype. This prototype was then subjected to various tests to evaluate its performance. This chapter aims to give context to the problem being explored and the key objectives.

1.1 Background

Traffic monitoring involves counting the number of vehicles driving on a road at a given point in time and recording their velocities. A traffic monitoring device is one that is able to perform this task automatically allowing the user to obtain traffic information easily. Such a product is useful in assisting infrastructure departments by providing them with a better understanding of the amount of vehicles travelling on their roads. This information can help identify areas in need of road upgrades, plan road maintenance around traffic and understand pollution emissions due to traffic.

1.2 Problem Statement

There is a need for devices that can monitor traffic conditions on public roads. Unfortunately, existing traffic monitoring devices are very costly (more than $240 [11]) making it difficult for buyers to purchase multiple devices and monitor multiple roads.

1.3 Objectives

The objective of this project was to design a low-cost traffic monitoring system using a Doppler radar. This included finding a suitable radar as well as a means of recording and processing the data. The system was required to count the number of vehicles on a road and record their speeds. A report documenting this investigation was also required, with the intention of highlighting the achievements of the project as well as means of improving the result.

1.4 Scope and Limitations

There was a strict time limitation on this project (14 weeks). The scope of this project was focused on developing the sensing setup as well as the processing algorithm to successfully monitor traffic on a road. This involved designing the system by testing various components. Algorithms to capture and process the data were then developed before the system was tested in a realistic scenario and its performance evaluated.

The main limitation encountered in this project was the goal of keeping the system low-cost. The Doppler radars used in this project are not high quality and have various drawbacks due to their price. Significant post-processing work was required to improve the low-quality data retrieved from the sensor.

Chapter 2

Theory

This chapter details the theoretical background of all the tools and concepts explored in this report. It aims to provide a foundation for the concepts with the goal of improving understandability.

2.1 Radar and the Doppler Effect

This report explores radars extensively as they were the sensing tool used to obtain traffic information. This section details the basic radar theory concepts used.

”A radar is an electrical system that transmits Radio Frequency (RF) Electromagnetic (EM) waves toward a region of interest” [12], these waves are then reflected by targets in the region of interest. The radar detects these reflections and determines the target’s characteristics such as its velocity and position.

The EM waves transmitted by the radar have a preset frequency band. The radar sensors used in this investigation reside in the X band (8-12GHz) and the K band (18-27GHz), both of which have transmit wavelengths in the microwave range. Furthermore, there are two general radar waveform classes: Continuous Wave (CW) and Pulsed waveform. CW radars are always transmitting a signal and simultaneously receiving the return signal at a receiver. Pulsed radars emit a ”sequence of finite duration pulses” [12] during which the receiver is off. In between each pulse, the transmitter switches off and the receiver switches on. All of the radars explored in this investigation were CW radars. The EM waveforms in both continuous wave and pulsed configuration are plotted in Figure. 2.1 and Figure. 2.2.

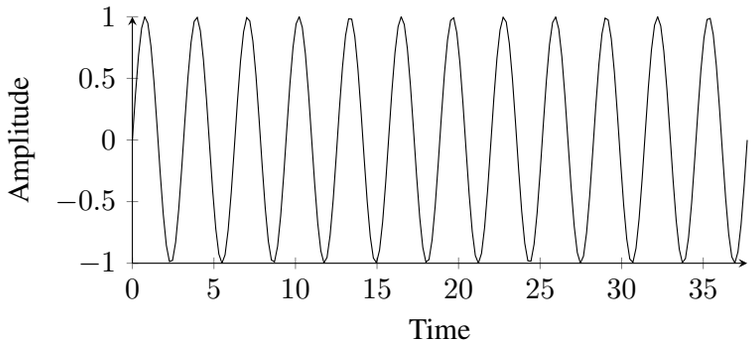


Figure 2.1: Continuous Wave Illustration

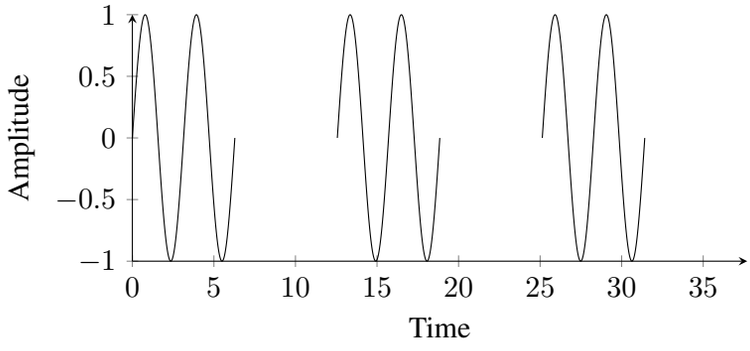


Figure 2.2: Pulsed Wave Illustration

The continuous nature of CW radar is illustrated in Fig. 2.1 which shows the transmit signal of such a radar. There are no breaks in the sinusoid whilst there are clear breaks in the Pulsed waveform seen in Fig. 2.2.

This investigation used radar to determine the velocity of the targets in a target range. Target velocity can be obtained from received EM waves using the Doppler Effect. The received signal from a moving target will be frequency-shifted based on the relative velocity of the target to the radar. The radars explored in this investigation were designed to detect motion and output only the Doppler shifted frequency. The Doppler shift caused by a target moving towards the radar will be positive (increase in frequency) whilst targets moving away from the radar will cause a negative Doppler shift. The speed of the target is obtained by observing the frequency difference between the transmitted and received signal [1]. These signals are sent and received by a microwave transceiver (a device capable of receiving and transmitting signals).

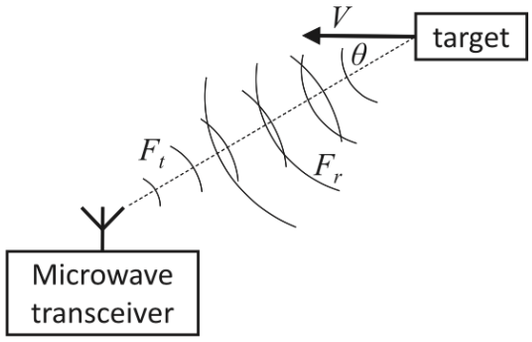


Figure 2.3: Visualisation of how microwave Doppler radars measure speed [1].

The frequency shift caused by the Doppler effect is proportional to the radial velocity (velocity of the target directly towards the radar) of the target, meaning Doppler radars are blind to stationary targets and targets moving perpendicular to the radar. Fig. 2.3 illustrates that the radial velocity of the target will increase/decrease as a factor of $cos\theta$.

Stationary targets being invisible to the radar reduces clutter from the received signal. Clutter is the received signals from other objects surrounding the target e.g. trees and buildings. Most clutter is generally stationary or slow-moving and can be removed by high-pass filtering.

Equation 2.1 describes the Doppler frequency calculation:

$$F_d = |F_t - F_r| = 2 \times V \times \frac{F_t}{c} \times cos(\theta) \tag{2.1}$$

Where F_d is the Doppler frequency, F_t is the transmit frequency, and F_r is the received frequency. V is the velocity of the target and θ is the angle between the direction of motion of the target and the radar beam. c is the speed of light ($3 \times 10^8 m/s$) [1]. The radars used in this investigation had transmit frequencies of 10.525GHz and 24.125GHz. The exact relationship between velocity and frequency for each of the above transmit frequencies is shown in Equation 2.2:

$$\begin{aligned} F_d &= 2 \times \frac{10.525 \times 10^9}{3 \times 10^8} \times V \\ F_d &= 70.16 \times V(m/s) \\ F_d &= \frac{70.16}{3.6} \times V(km/h) \\ F_d &= 19.49 \times V(km/h) \end{aligned} \tag{2.2}$$

The same process with a transmit frequency of 24.125GHz resulted in:

$$F_d = 44.68 \times V(km/h) \tag{2.3}$$

Radars have a limited range which is defined by the radar range equation:

$$SNR = \frac{P_t G_t G_r \lambda^2 \sigma}{(4\pi)^3 R^4 k T_0 F B} \tag{2.4}$$

Where:

- k is Boltzmann's constant ($1.38 \times 10^{-23} watt - sec/K$)
- T_0 is the standard temperature (290K)
- R is the distance between the target and radar
- G_t and G_r is the antenna gain for the transmitter and receiver respectively
- σ is the mean Radar Cross Section (RCS), a measure of the target's reflectivity
- P_t is the transmitted power
- λ is the carrier wavelength in meters
- F is the noise figure of the receiver subsystem

Equation. 2.4 shows the Signal to Noise Ratio (SNR) which is the ratio of signals received from the target over signals received from noise. The most important information to be drawn from Equation 2.4 is that the SNR decreases by a factor of R^4, meaning that a short increase in distance between target and radar decreases the SNR significantly.

Whilst distance is one restriction on what radar is capable of detecting, the target must also be within the radar's beam. There are two standard plots used to visualise a radar's beam, the elevation plot which shows the vertical spread of a radar beam, and the azimuth which shows the horizontal spread.

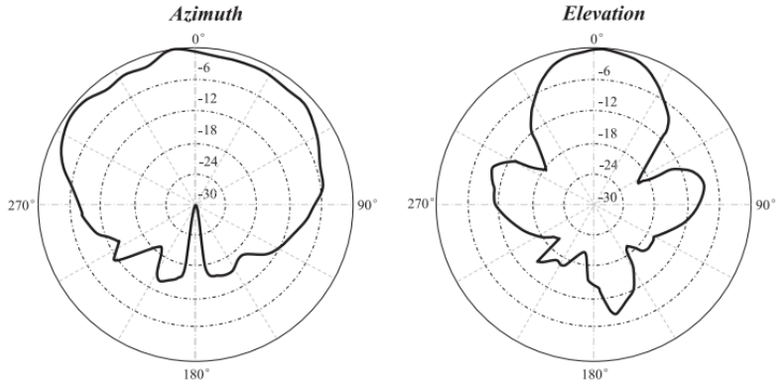


Figure 2.4: Example of Radar Elevation and Azimuth Plots [1].

2.2 Sampling

This section details the theory behind sampling analogue signals, focusing primarily on the sampling rate and Analogue to Digital Converters (ADC).

Sampling is the process of converting a continuous signal (the radar output) to a discrete one that can be processed digitally. The sampling rate dictates how often the continuous signal's value is converted to a discrete one. The higher the sampling rate, the better the discrete representation of the continuous signal. If a signal is sampled too slowly, then information from the continuous signal is lost. The minimum sampling rate to capture all of the information in a signal is known as the Nyquist Sampling rate. This states that the sampling rate must be higher than 2 times the highest frequency component in the continuous signal. For example, a 2kHz sinusoid must be sampled at a rate of at least 4ksps (kilo-samples-per-second) to accurately represent the signal discretely [13].

Whilst the sampling rate dictates how fast the continuous signal is sampled, the ADC resolution determines how well the analogue voltage is converted to a digital value. ADCs come with various different resolutions ranging from 8 - 24 bits and more. An 8-bit ADC is less precise than a 24-bit ADC. The number of values that an ADC can represent is given by:

$$Number\ of\ Values = 2^N - 1 \tag{2.5}$$

Where N is the number of ADC bits. In the scenario where 5V is the maximum continuous output of a device, the smallest voltage increment that can be represented by an 8-bit and a 16-bit ADC are shown in Equations 2.6 and 2.7:

$$8\ bit\ ADC = \frac{5}{2^8 - 1} = 0.0196V \tag{2.6}$$

$$16\ bit\ ADC = \frac{5}{2^{16} - 1} = 7.63 \times 10^{-5}V \tag{2.7}$$

It can be seen from Equations 2.6 and 2.7 that the 16-bit ADC is able to convert much smaller voltages [14]. The risk of using a low-resolution ADC is that small voltage details in the signal will be lost in the sampling process.

2.3 Spectrogram

A spectrogram is an analysis tool that was used throughout this investigation to help visualise the velocity of targets visible to the radar. A spectrogram is a frequency vs time heatmap that showcases the received frequency, the time they were received and the power of that frequency at that time.

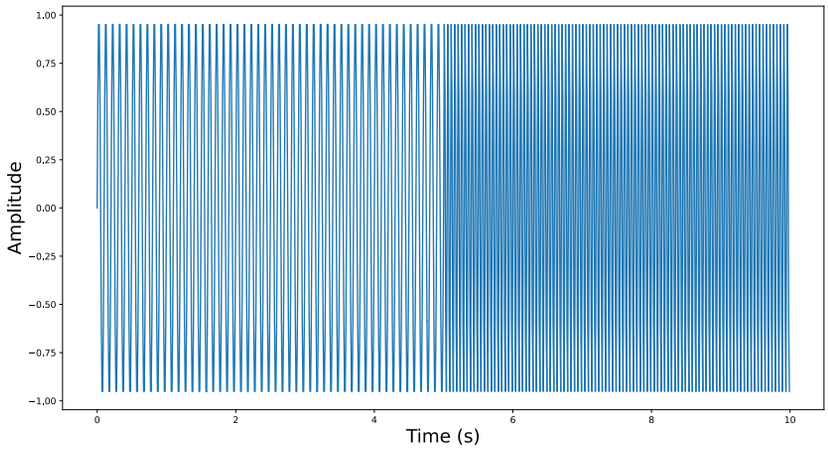


Figure 2.5: Time Domain Plot Used to Showcase Spectrogram [2]

Fig. 2.5 shows the time domain plot of a sinusoid oscillating at 10Hz from 0-5 seconds and 20Hz from 5-10 seconds. This signal is used to visualise the spectrogram plot.

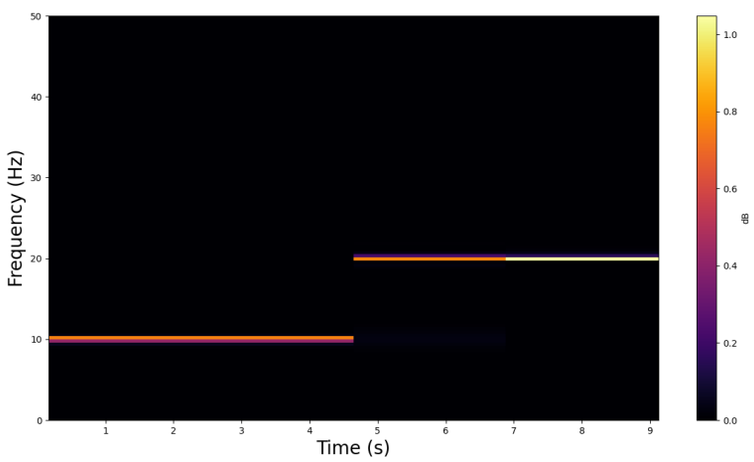


Figure 2.6: Spectrogram of Time Domain Plot Seen in Figure. 2.5 [2]

Fig. 2.6 clearly shows the frequency elements of the sinusoid in Fig. 2.5 and their respective times. This plot has high power elements at 10Hz for the first 5 seconds and then at 20Hz for the last 5 seconds as expected.

The spectrogram works by performing a Discrete Fourier Transform (DFT) on a short portion of the signal at a time. This is known as the Short Time Fourier Transform (STFT) and is done by windowing the signal and performing a DFT on each windowed signal.

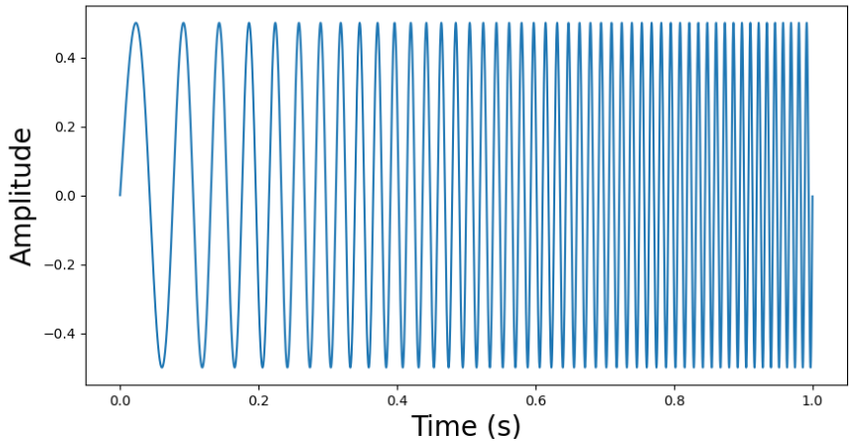


Figure 2.7: Time Domain Signal Before Windowing

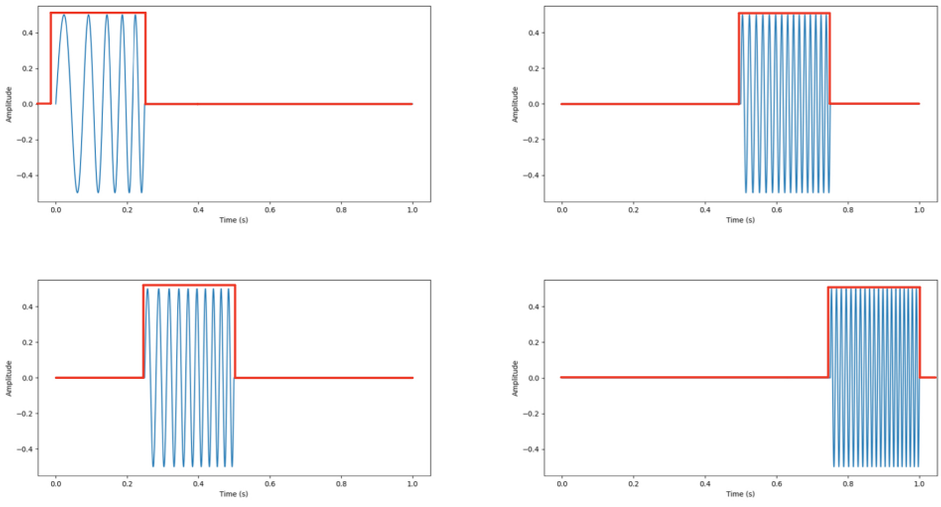


Figure 2.8: Time Domain Signal with Basic Rectangular Window Applied

Fig. 2.8 shows a time domain signal with a rectangular window applied. Each windowed block observed in the figure is referred to as a frame. The spectrogram is plotted using the frequency spectrum of each frame and the time of each frame. This is different to a standard DFT which is technically just one large frame and therefore has no time information. The spectrogram typically uses a Hann window seen in Fig. 2.9 as opposed to the rectangular window shown in Fig. 2.8 as it allows for a smoother frequency spectrum [15].

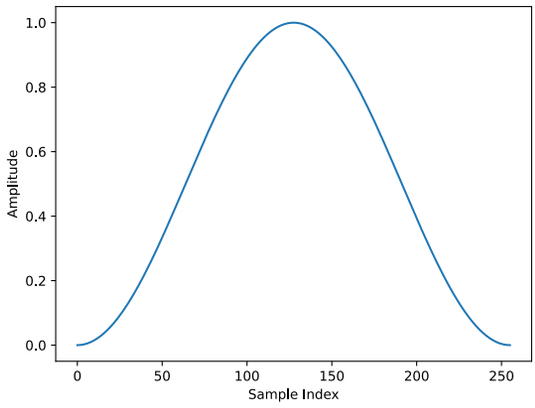


Figure 2.9: Hann Window

Furthermore, the frame size can be selected based on the user's requirements with a larger frame favouring frequency resolution and a smaller frame favouring time resolution [16]. The windows can also be overlapped to create a 'zoom' effect on the time axis of the spectrogram [17].

2.4 Gaussian Smoothing

This investigation also required the removal of noise from spectrograms. One such technique for reducing spectrogram noise is Gaussian Smoothing.

Gaussian Smoothing is an image-processing technique used to remove noise from images by 'blurring' the image. This is done by convolving the image with a kernel that has a Gaussian distribution of weightings.

0.003	0.013	0.022	0.013	0.003
0.013	0.060	0.098	0.060	0.013
0.022	0.098	0.162	0.098	0.022
0.013	0.060	0.098	0.060	0.013
0.003	0.013	0.022	0.013	0.003

Figure 2.10: 5x5 Gaussian Kernel

Convolution works by moving a kernel, like the one seen in Fig. 2.10, across all the pixels in an image and calculating the weighted sum of each square in the kernel multiplied by the pixel value. The pixel at the centre of the kernel is then set to that summed value. This is repeated for all pixels in the image. The result is a removal of high-frequency noise present in the image but also blurring of the image [18].

The smoothing can also be applied to spectrograms to filter out high-frequency noise, as well as increase the signal power of the target. SciPy (a Python library) has a Gaussian Filter function that computes the convolution operation with the spectrogram. The kernel size can be adjusted by varying a $\sigma = (A, B)$ variable in the function. Changing A varies the height of the kernel (and thus varies the smoothing of the frequency axis of the plot) and changing B varies the width of the kernel (adjusting the time axis smoothing). The effect of this smoothing is visualised by adding noise to the plot in Fig. 2.5:

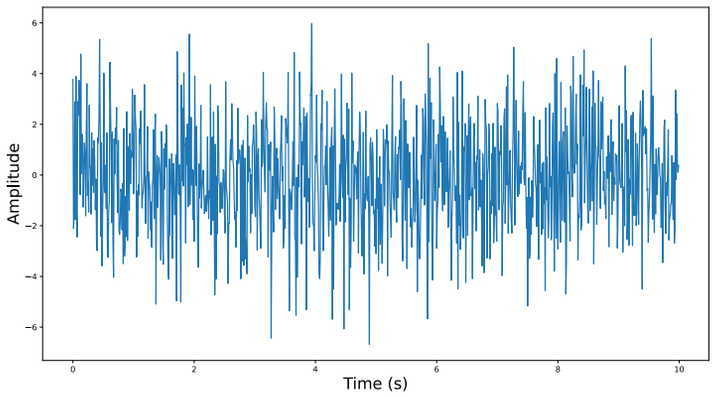


Figure 2.11: Fig. 2.5 with Added Noise

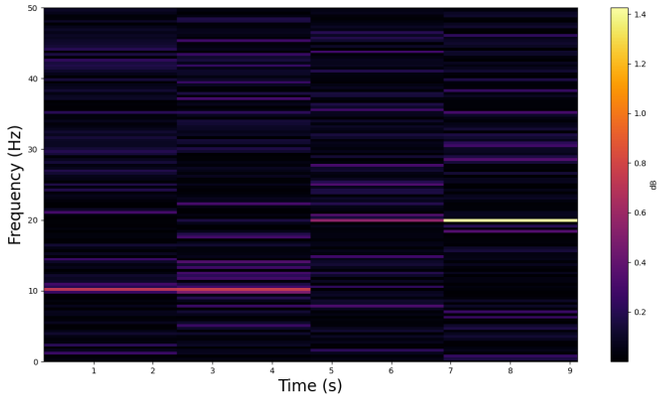


Figure 2.12: Spectrogram of Fig. 2.11

The effect of Gaussian smoothing on the time and frequency is shown in Fig. 2.13 and 2.14:

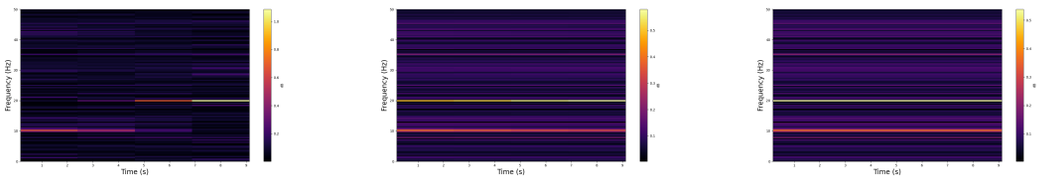


Figure 2.13: Gaussian Smoothing Applied to Time Axis with Progressively Increased Kernel Size

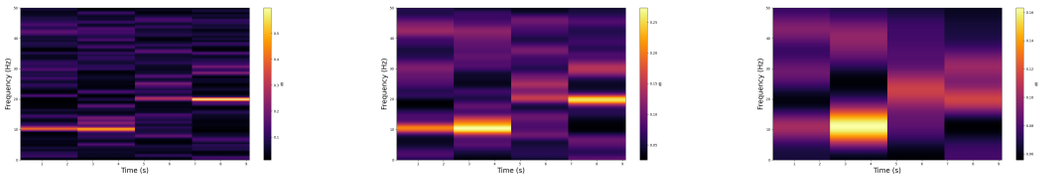


Figure 2.14: Gaussian Smoothing Applied to Frequency Axis with Progressively Increased Kernel Size

Fig. 2.13 clearly shows horizontal blurring with both the 10Hz and 20Hz frequencies running along the entire time axis. Fig. 2.14 shows a vertical blurring with the 10Hz and 20Hz frequency lines becoming much wider.

A selective combination of smoothing along the time and frequency axis of the spectrogram can be used to improve the SNR of a target amongst noise. However, over-smoothing can result in a loss of frequency and time accuracy.

2.5 Moving Average Filter

The Moving Average Filter applies a similar concept to Gaussian Smoothing and is used to remove high-frequency noise from signals (essentially smoothing the signal). However, unlike the Gaussian Kernel, the moving average filter is merely an average of a number of data points in a plot. For example, a 3-point moving average filter averages the centre point and a point on either side of the centre point to obtain a new filtered value for that point.

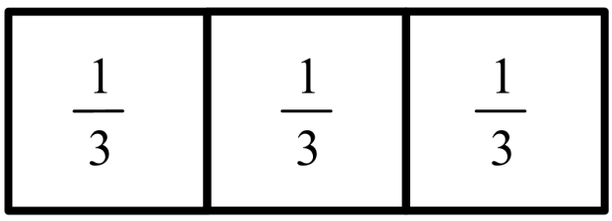


Figure 2.15: Illustration of 3-Point Moving Average Filter Kernel

Fig. 2.15 illustrates the kernel of a 3-point moving average filter. The number of points in the filter can be increased to increase the smoothing effect.

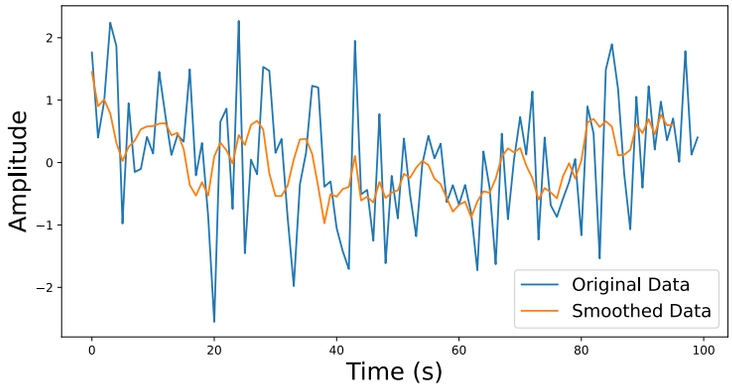


Figure 2.16: 5-Point Moving Average Filter Applied to Signal

Fig. 2.16 showcases that the high-frequency elements of the time domain signal have been removed and it appears to have been smoothed with much gentler gradients.

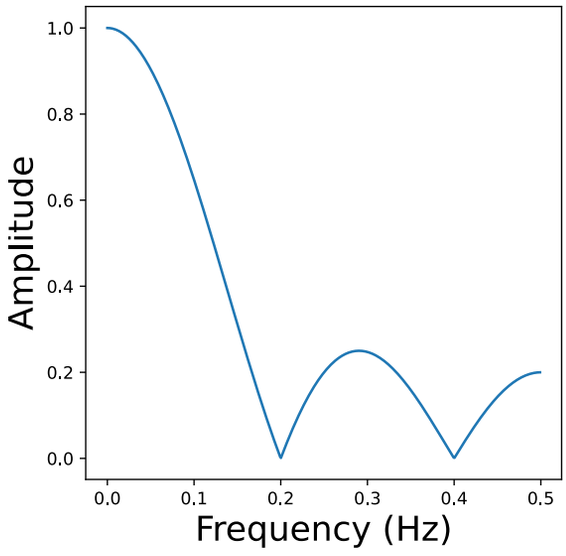


Figure 2.17: Frequency Spectrum of 5-Point Moving Average Filter

The frequency spectrum, seen in Fig. 2.17 of the filter clearly shows that only very low frequencies are not attenuated by the filter. Thus, these filters cannot be used if the frequency content in the signal is important [19].

2.6 The Hough Transform

The Hough transform is briefly discussed later and applied to spectrograms. This section details the theoretical process of performing a Hough transform.

The Hough transform is an image-processing technique used to help identify shapes in an image. The technique works by plotting shapes as a function of their parameters as opposed to their x and y values.

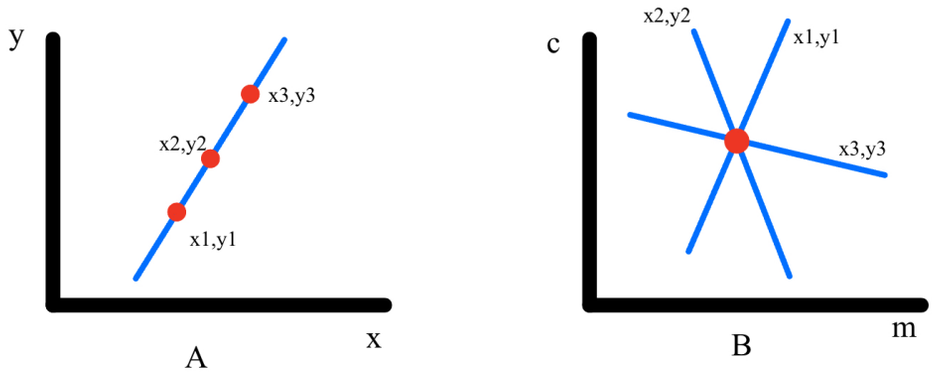


Figure 2.18: Hough Transform Illustration

Plot A in Fig. 2.18 is a straight line in image space of the form seen in Equation 2.8.

$$y_i = mx_i + c \tag{2.8}$$

$$c = y_i - mx_i \tag{2.9}$$

The Hough transform can be used to determine the m and c values of the line in A using the coordinates of the points on the line. Plot B shows the parameter space of the line which is of the form seen in Equation 2.9. Substituting the coordinates for the red points seen in Plot A obtains three straight-line equations in parameter space. The intersection of these three lines gives the m and c values of the blue line in Plot A [20].

Chapter 3

Literature Review

This chapter investigates some of the existing research into traffic monitoring using radar. Firstly, an overview of some of the common sensing tools used in traffic monitoring is given. This overview does not deal exclusively with radars but all sensors.

The chapter then delves into the different radars that others have used, as well as how these radars were set up in relation to the road and the software analysis techniques applied to the data. The same few radar modules were commonly used in the literature and this has been used to order this chapter. Each section deals with a single radar module and all of the literature in that section deals with that specific module. Information on these radars is also given at the start of each section to provide an understanding of the sensors used in each piece of literature.

3.1 Traffic Monitoring Techniques

Pneumatic tubes are a common method for observing road traffic. A pair of tubes can be laid out across a road, separated by some distance, and can measure the speed of a vehicle by recording the time the vehicle drove over each tube. These tubes are limited, however, to work only on two-lane roads and at slow vehicle speeds. Due to these limitations, these tubes are typically used to verify the data obtained from some of the other sensing methods mentioned in this section [21].

Video analysis is another tool often used for traffic monitoring and uses cameras as the primary sensor. The vehicles in the video are often identified using a Convolutional Neural Network (CNN). The advantage of video analysis is that multiple lanes can be monitored simultaneously and it can be used to monitor high traffic scenarios. Its major drawback is light levels. The cameras used for data capture perform significantly worse at night when visibility is poor [21].

Light Detection and Ranging (LiDAR) can also be used for traffic monitoring. LiDAR uses a narrow infrared beam to identify a target. These beams are narrower than radar beams (like the one in Fig. 2.4) and are less prone to measurement errors. However, a single LiDAR is generally only able to identify

and classify vehicles but cannot determine velocity [21].

Lastly, Doppler radars are the most common tool used in traffic scenarios. The theory behind these devices has been detailed in Chapter 2. Doppler radars are not limited by light level and are able to detect a wide range of speeds. These devices struggle in multi-lane scenarios where one car can be occluded by another. Furthermore, Doppler radars struggle in slow-moving traffic scenarios where vehicles stop and start in front of the radar [21].

3.2 Doppler Radars

With a basic overview of the popular traffic monitoring tools established, this section deals specifically with different low-cost Doppler radars and their implementation.

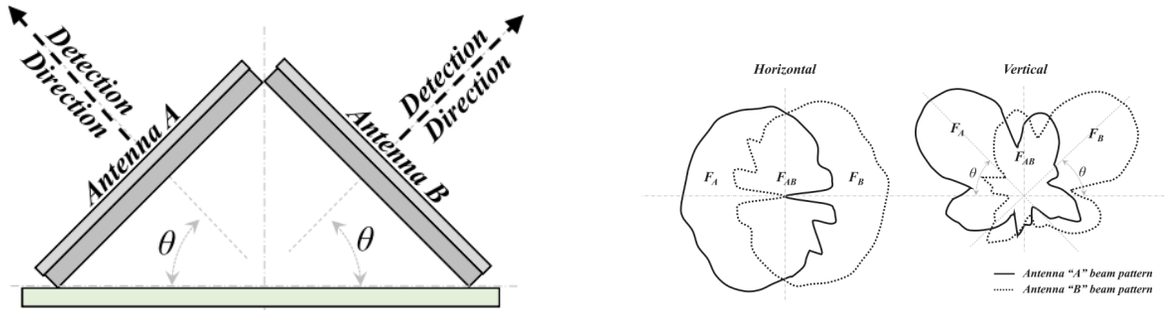
3.2.1 HB100

HB100 is a low-cost microwave Doppler radar transceiver module that operates at a fixed frequency of 10.525GHz in the X-band. It is a bi-static radar, meaning its transmitter and receiver are at separate locations on the radar. It is capable of a max radiated power output of 20dBm and can detect objects up to 15m away.

Figure 3.1: HB100 Radar Module [3]

This radar transmits a continuous wave signal and outputs the Doppler signals received by the radar from moving targets. This section details some of the traffic-related designs implemented using the HB100.

Traffic in parking bays has been monitored using a combination of two HB100 radars. These two radars were used to identify when a car entered a parking bay, was parked in the bay, and was leaving. The layout of the two radars is seen in Fig. 3.2



(a) Layout of the two radars

(b) Antenna beams of both radars combined

Figure 3.2: Parking bay monitoring setup [1]

The angle θ of each radar was selected to be $45°$ because a small θ (around $0°$) resulted in too much interference from the ground, and a large θ (around $90°$) resulted in the radar's beam being perpendicular to the vehicle's motion and thus no Doppler shift would be detected [1].

The output voltage of the radar is directly proportional to the energy reflected off of the target (in the millivolt range) meaning this output had to be amplified before being sampled. Agilsense (the manufacturer of the radar) recommended a two-stage high-gain low-frequency amplifier. This system implemented the recommended amplifier which is shown in Fig. 3.3.

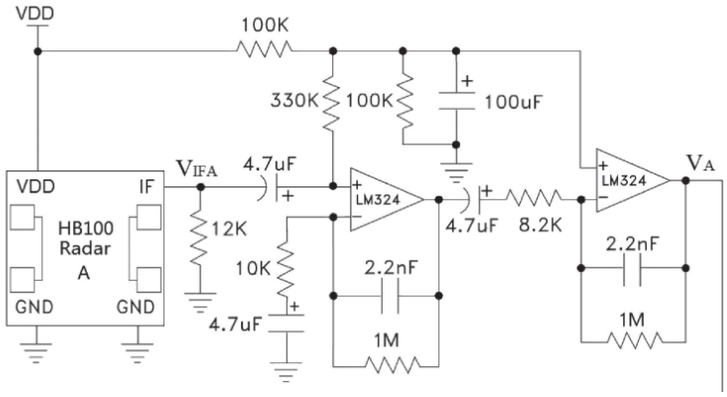


Figure 3.3: The amplifier used by Bao et al. to amplify the HB100 output voltage [1].

The system was designed to be placed in the centre of the parking bay such that Antenna A (as labelled in Fig. 3.2) was facing the entrance to the bay, and monitored for cars as follows:

1. If radar A detected an object slightly before radar B did, then a car was driving into the bay.
2. Subsequently, if both A and B stopped detecting any objects then the car was parked in the bay and had stopped moving.
3. If radar B detected an object slightly before radar A did, and then lost sight of the car before A did, then a car was driving out of the bay.

Once the amplified output of the radar was sampled, the data processing was done on a microcontroller.

A detection time window (W seconds) was selected and the area under the voltage-time graph was computed for this duration. If the area under this plot exceeded a chosen threshold (in this case a value of 0.05) then the radar had detected an object [1].

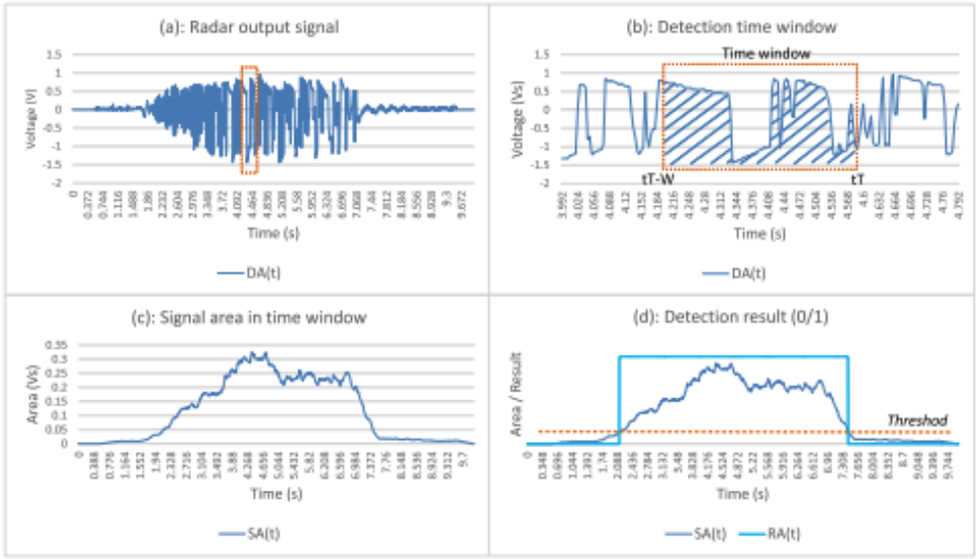


Figure 3.4: Example plots of the data analysis process [1].

In Fig. 3.4 the blue rectangle labelled RA[t] represents a binary value that is set to 1 whenever a vehicle is detected and 0 otherwise. This system was able to successfully identify targets but velocity information was not important.

In addition to parking lot traffic, a single HB100 radar, placed in the centre of the road above where the cars would drive, has been used to identify the speed and size of vehicles driving on the road. The placement of the radar is shown in Fig. 3.5. The output of the radar was amplified and converted to a square wave. This square wave was then fed into a PIC 16F886 microcontroller where the length and speed of the vehicle were determined [4].

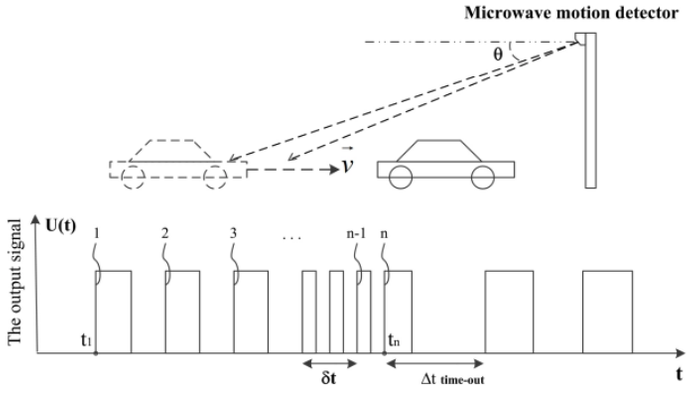


Figure 3.5: Radar setup and voltage output [4].

The length (S) of the vehicle was calculated using Equation 3.1:

$$S = \int_{t_1}^{t_n} \frac{n}{k} \tag{3.1}$$

Where n was the gross number of times the output signal went from low to high, and k was a scale factor given by:

$$\frac{2F_t}{c} cos\theta \tag{3.2}$$

Equation. 3.1 was determined by integrating the relationship between Doppler frequency and velocity (seen in Equation. 2.1) to obtain a distance relationship. In order for this to work, θ was assumed to be constant for the detection interval. This is shown in Equation 3.3.

$$S = \int_{t_1}^{t_n} \frac{xc}{2F_t cos\theta} = \frac{nc}{2F_t cos\theta} \tag{3.3}$$

Where x is the number of times the signal goes from high to low in an interval dt. $t_n - t_1$ was the duration the vehicle was visible to the radar.

If the time Δt that the signal was low exceeded a chosen $\Delta t_{time-out}$ then the car was said to be moving out of the detection zone of the radar. This occurred because as the vehicle approached the radar, its radial velocity towards the radar would decrease resulting in lower frequency Doppler shifts.

The $\Delta t_{time-out}$ was selected using the minimum desired detectable speed, the transmit frequency of the radar and the angle the radar made to the road:

$$\Delta t_{time-out} = \frac{c}{2V_{min}F_t cos\theta} \tag{3.4}$$

The average speed of the vehicle was computed as follows:

$$V = \frac{S}{t_n - t_1} \tag{3.5}$$

This method saw an accuracy rate of greater than 80% when detecting trucks, cars and buses moving in a speed range of 5-40km/h and at a distance of up to 8m away from the radar.

3.2.2 CDM324/IPM-165

CDM324/IPM-165 (this radar will be referred to as CDM324 in this report but it goes by both names) is another low-cost CW radar. The radar operates in the K-band at a frequency of 24.125GHz [22]. It is equipped with 4 pairs of receiving and transmitting antennas.

Figure 3.6: CDM324 Radar Module [5]

This radar has been used in a smart traffic monitoring system, to identify vehicles moving past the radar and their speeds. In this system, the radar was placed on the side of the road and pointed across the road. This resulted in smaller Doppler shifts than some of the previous scenarios because the vehicle was not moving directly towards the radar [6].

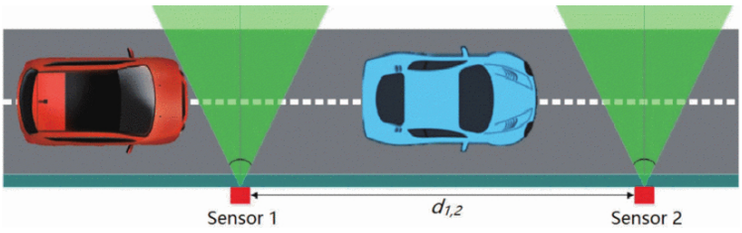


Figure 3.7: Radar setup [6].

As seen in Fig. 3.7, this setup used two radars that were separated by a distance greater than the length of the vehicles being observed. The difference in time between sensor 1 detecting the vehicle and sensor 2 detecting the vehicle was used (along with the distance between the two sensors) to determine the speed of the vehicle.

The two radar outputs were processed by low-pass filtering and thresholding, after which the time difference between each detection could be observed:

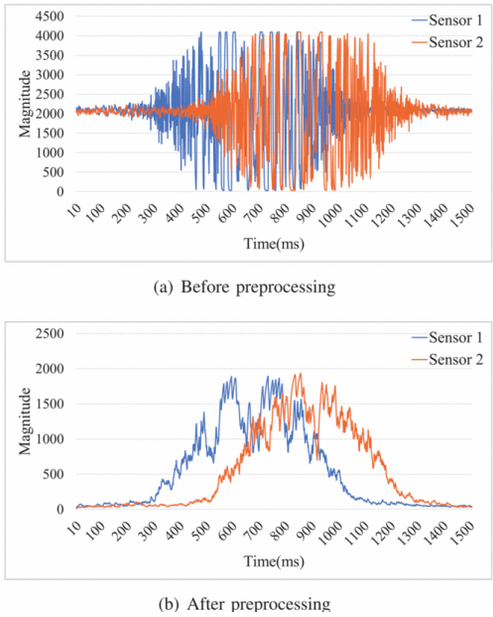


Figure 3.8: Sensor Data [6].

This system was only tested on a single lane of traffic but had a vehicle detection accuracy of 98.3% and speed measurement accuracy of 95.8%.

3.2.3 RSM-2650

RSM-2650 is another microwave Doppler radar. It operates in the K-band at 24.250GHz.

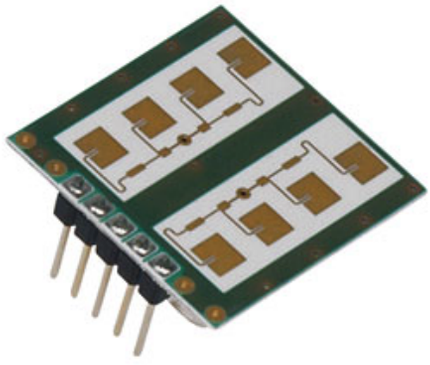


Figure 3.9: RSM-2650 Radar Module [7]

This radar module has been combined with a soundcard and a Raspberry Pi Module B in a traffic logger. The radar module has two outputs, one outputs the real part of the received signal and the other the imaginary. The real output can be used to determine the velocity of the target, and the imaginary part can be used to determine the direction of motion, a positive phase is caused by motion toward the radar and a negative phase by motion away from the radar [8].

The data was amplified using an LM386 Amplifier Module.

The radar output was sampled using a soundcard (the details of this process are explored later in Chapter 5) and processed in Python (running on the Raspberry Pi). The system was placed on the side of the road facing towards traffic (the exact positioning was not specified) [8].

Interestingly, it was found that placing the whole system in a single box caused a lot of interference in the radar data. This interference was avoided by keeping only the radar and amplifier in a box and housing the Raspberry Pi, Soundcard and power supply elsewhere nearby [8].

The accuracy of this system was not specified but plots of the vehicles detected by the system and their velocities were given:

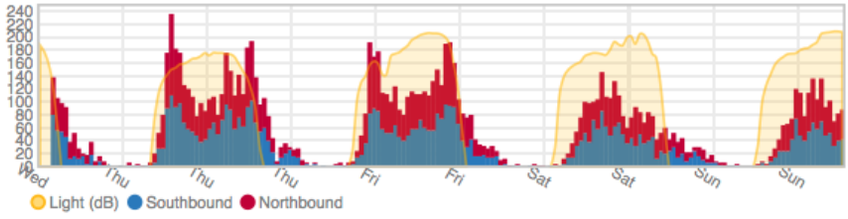


Figure 3.10: Number of Cars Counted Over 4 Days [8]

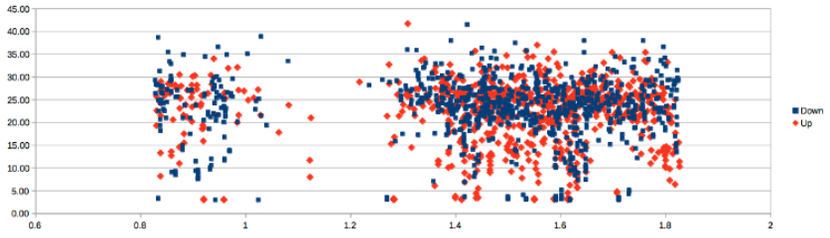


Figure 3.11: Scatter Plot of Car Velocities Over 24 Hours [8]

From Fig. 3.10 it can be seen that the system was able to count large numbers of cars at a given time (up to 230 cars per hour) and was able to run continuously for multiple days. Fig. 3.11 shows that most of the vehicles recorded in the 24-hour interval were travelling at a similar speed (as expected due to road speed limits). The system seems to be able to function at low speeds (up to 40 mph). However, because the accuracy of the system was not specified, its performance cannot be easily verified.

3.2.4 Other Radars

This section explores a traffic monitoring solution that implemented a radar that has not been mass-produced. This radar was a continuous wave K-band radar (24 GHz) made with the objective of traffic monitoring. This radar was capable of detecting a vehicle, measuring its speed, and classifying the type of vehicle simultaneously. The radar module was made up of the following [9]:

- A horn antenna with a receiver that output Doppler signals.

- An analogue signal amplifier.
- A Digital Signal Processor (the TMS320F2808 from Texas Instruments).

This radar was designed to be affordable and cost less than $250.

The system had a configuration much like the one seen in Fig. 3.5:

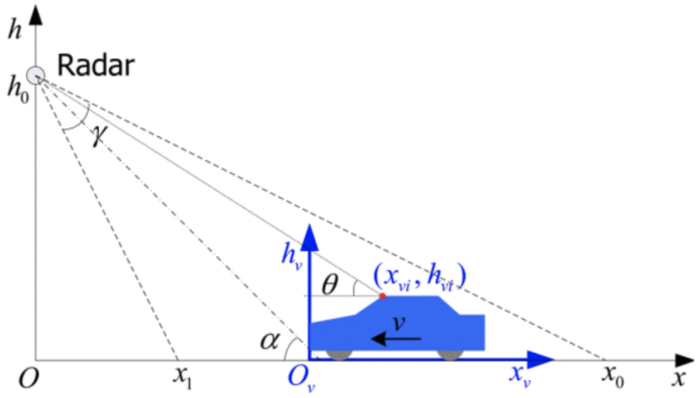


Figure 3.12: Radar Positioning on the Road [9].

The system identified the vehicle using various scatterings mapped across the car, which each reflected different Doppler frequencies. These scatterings are illustrated in Fig. 3.13. The Doppler frequency reflected by each scattering was given by:

$$f_{di}(t) = \frac{2vf}{c} cos\theta_i(t) \tag{3.6}$$

Where $\theta_i(t)$ is the angle each scattering makes with the radar. These reflected frequencies decreased with time due to the increasing θ_i as the car approached the radar.

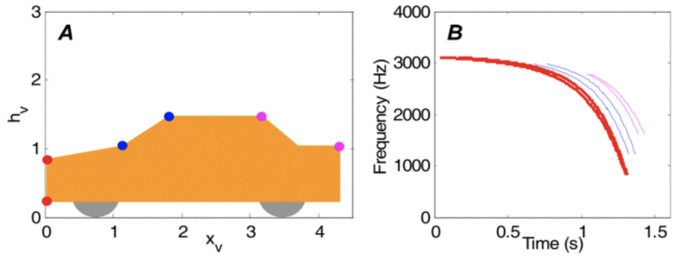


Figure 3.13: Scatterings Shown on a Spectrogram [9].

The important information in the spectrogram was separated from noise using thresholding. The thresholding process worked by finding the start point of the target using a low power threshold TH_1 and finding the time and frequency (t_{start}, f_{dstart}) when the signal power was greater than this threshold.

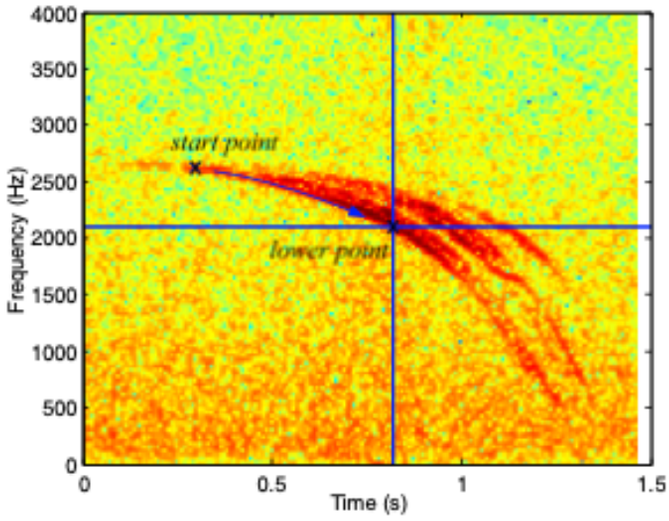


Figure 3.14: Implementation of Thresholding Algorithm on Spectrogram [9].

A lower point was then obtained using a frequency threshold, TH_2 which was set to be ρf_{dstart}. ρ was typically between 0.7 and 0.9, Fig. 3.14 shows the results with a $\rho = 0.9$. Once this lower frequency and time t_l had been found, then the entire spectrogram power from the start point to the lower point was calculated and if that value was above a large power threshold TH_3 then the target was confirmed.

This detection algorithm was able to detect vehicles in high-traffic scenarios successfully. The algorithm was able to eliminate interference from vehicles in other lanes and had a detection accuracy greater than 95%. The speed detection was also very accurate, with an average accuracy of 97%.

The classification of the vehicles used the different Doppler signatures caused by the scattering points on the car to identify the car type. This system made use of the Hough Transform explained earlier in Chapter 2.6 [23].

The Hough transform was used to transform from a frequency-time plot to an x coordinate vs height plot for each scattering. This was done by applying the transform to a formula that related the car position to the Doppler frequency received from the spectrogram.

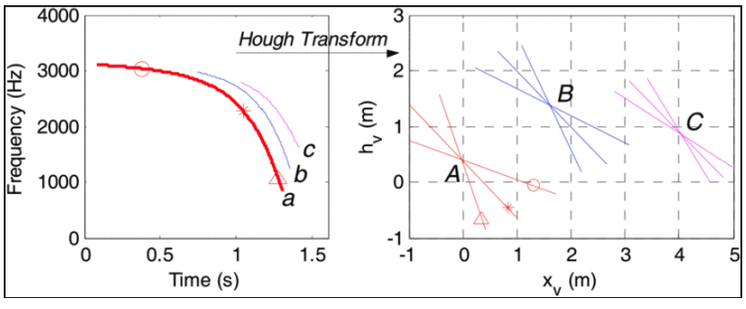


Figure 3.15: Hough Transform Applied to Radar Readings [9].

The intersections of the lines in Fig. 3.15 represent the coordinates of the scattering points on the car. The transform applied to an actual spectrogram obtained the plot shown in Fig. 3.16.

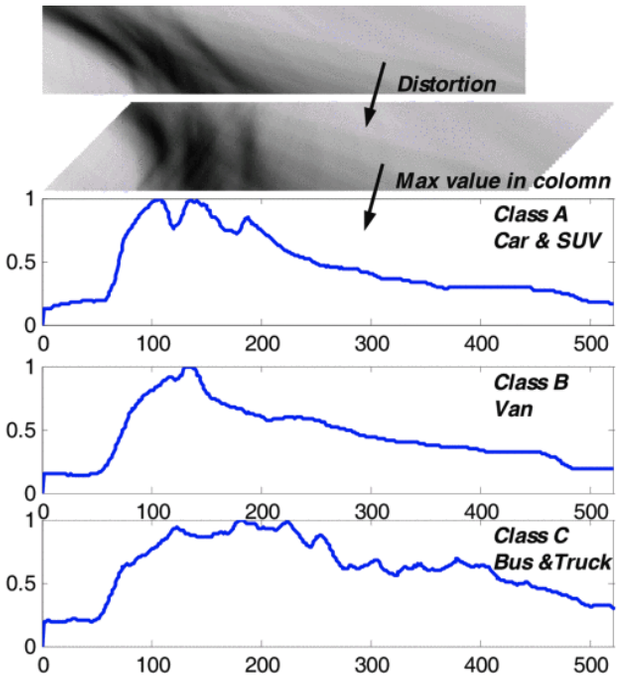


Figure 3.16: Actual Plots of Transformed Data [9].

This plot was then rotated by $45°$ to orient the black stripes vertically. Lastly, the intersection point of each stripe was found by finding the maximum pixel value in each column of the plot. This obtained the third plot shown in Fig. 3.16. The peaks of the blue lines indicate the scatterer's location.

These final plots were then classified using a machine-learning algorithm. This algorithm was able to identify three categories of vehicles shown in Fig. 3.16 using the different scattering locations [9]. This classification method achieved an accuracy of 94.8%.

Chapter 4

Requirements Analysis

This section details the requirements of this project. These requirements were derived from the project description. Technical specifications required to meet the listed requirements are also detailed. Lastly, mechanisms for testing that the system meets the requirements are listed under the Acceptance Test Procedures (ATP) section.

4.1 Requirements

The system must:

1. Identify the number of cars moving along a road
2. Identify the speeds of the cars
3. Be scalable, such that multiple lanes of traffic can be monitored
4. Be portable
5. Be low cost, less than $100

4.2 Specifications

1. The devices must be small, and the whole system must fit in a 4-litre box
2. The devices used must run on 5V (typical battery power supply or power bank)
3. The code used must be compatible with a microcontroller
4. The obtained data must be transmitted wirelessly to a central device

4.3 Acceptance Test Procedures

1. Test the system on a busy roadside and verify that it correctly detects the cars travelling past it
2. Verify that the speeds detected by the system agree with the typical speed limit of the road
3. Place the system in a small box (4L or less) to ensure that the system is compact
4. Run the system off of a portable power supply (batteries/power bank)
5. Run the data capture algorithm on a microcontroller
6. Capture data using the compact system and transmit the data to a laptop wirelessly
7. Calculate the total cost of the system

Requirements	**Specification**	**Acceptance Test Procedures (ATP's)**
R1		ATP1
Identify the number of cars moving along a road		Test the system on a busy roadside and verify that it correctly detects the cars travelling past it
R2		ATP2
Identify the speeds of the cars		Verify that the speeds detected by the system agree with the typical speed limit of the road
R3	S1	ATP3
Be scalable, such that multiple lanes of traffic can be monitored	The devices must be small, and the whole system must fit in a 4-litre box	Place the system in a box to ensure that the system is compact
R4	S2	ATP4
Be portable	The devices used must run on 5V (typical battery power supply or power bank)	Run the system off of a portable power supply (batteries/power bank)
	S3	ATP5
	The code used must be compatible with a microcontroller	Run the data capture algorithm on a microcontroller
	S4	ATP6
	The obtained data must be transmitted wirelessly to a central device	Capture data using the compact system and transmit the data to a laptop wirelessly
R5		ATP7
Be low cost, less than $100		Calculate the total cost of the system

Table 4.1: Requirements Breakdown

Chapter 5

Design

This chapter details the process of designing the traffic monitoring system. It begins with an exploration of the components required and justifies the component choices made. The process of designing the hardware setup using the components selected is then explained. Lastly, the design process for the system software is presented.

5.1 Component Selection

This section describes the process of selecting components to be used in the implementation of the traffic monitoring device. The following component categories were investigated:

- Doppler Radars
- Amplifiers
- Microcontrollers and Sampling Devices

5.1.1 Doppler Radars

The radar used in this system had to be able to identify the speed of a target using the Doppler shift caused by the target. The sensor also had to be low-cost, portable and low-power enough to run off of a battery-powered system. Three radars were investigated in this project:

- SEN0192
- CDM324
- HB100

The RSM-2650 seen in Chapter 3 was excluded due to its cost of $41.57 (R784).

SEN0192

SEN0192 is a microwave motion sensor that is powered by 5V and typically draws 37mA. The sensor has a built-in amplifier to account for the very weak outputs of the Doppler radar itself. The radar module has a built-in potentiometer to adjust the gain of the amplifier. The radar transmits a continuous wave with a frequency of 10.525GHz and outputs only the received Doppler frequency. The device cost $8.89 (R167).

Figure 5.1: SEN0192 Radar Module [10]

Unfortunately, the module was designed to be used as a motion sensor and only output a digital low whenever a Doppler frequency was detected by the device.

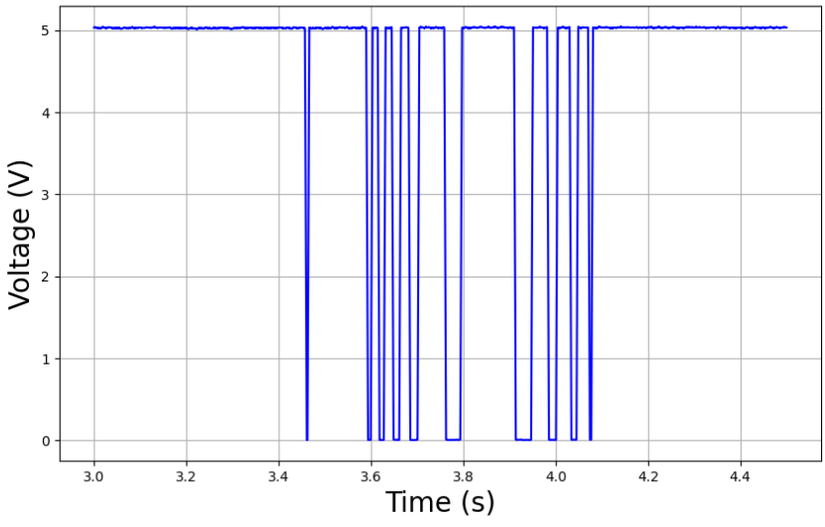


Figure 5.2: SEN0192 Time Domain Plot

Fig. 5.2 was obtained by moving a hand towards and away from the device. Whilst the motion was clearly detected (due to the multiple oscillations seen in the plot) and a target could be identified, the frequency spectrum of this output could not be used to determine the speed of the target. This was due to the nature of square waves and the fact that they are made up of multiple sinusoidal harmonics. This

made it impossible to distinguish the Doppler frequency from the harmonics. Furthermore, because the device did not output the Doppler frequency but instead output 0V whenever a Doppler shift was detected, the square wave seen above was not oscillating at the received Doppler frequency.

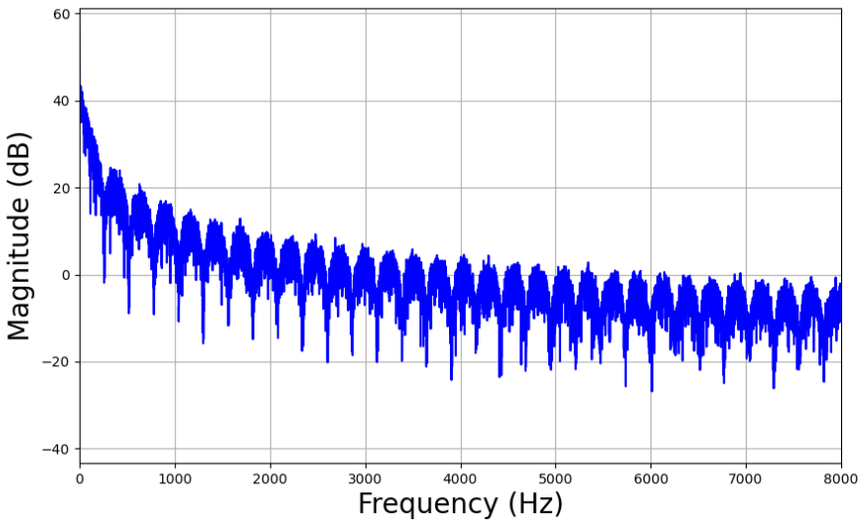


Figure 5.3: SEN0192 Data Frequency Spectrum

This made the SEN0192 unsuitable for this application as obtaining velocity information on the target was an important requirement of the sensor.

CDM324

This device was mentioned earlier in Section 3.2.2. The device runs off of a 5V power supply, requires 30mA of current and was priced at $6.47 (R121.52). Unlike the SEN0192, this device does not have a built-in amplifier and outputs a sinusoid oscillating at the Doppler frequency created by the target.

The following plot shows the raw output of the device when a hand was moved towards and away from the sensor:

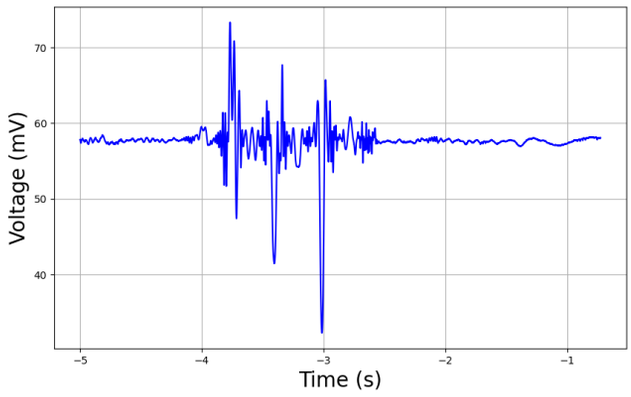


Figure 5.4: CDM324 Raw Output

As can be seen from the above figure, the output voltage of the sensor was very small, around 40mVpp when the target was very close to the device, and had a DC shift of just less than 60mV. This small output was found to be problematic as when targets were farther away (like when the sensor was placed around public roads) the output of the device would be too small to be adequately sampled by a microcontroller. Furthermore, noise might have drowned out the target signal in practical applications. To account for this, the radar output had to be amplified. The amplifier design process is detailed later in Section 5.1.2.

HB100

This device was mentioned earlier in Section 3.2.1. It is powered using 5V and was priced at $4.18 (R80). The raw output from the radar, created when a hand was moved towards and away from the device, is shown in Fig. 5.5:

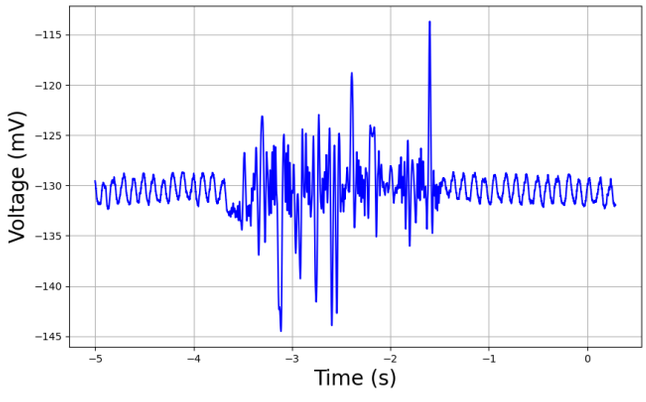


Figure 5.5: HB100 Raw Output

The HB100 applies a negative DC shift to the signal (around -130mV) and there is also a clear low-frequency signal visible when there are no targets moving. This low-frequency signal was absent in the CDM324 plot. A drawback of this negative DC shift was that the signal could not easily be amplified.

This was because the OpAmps used were powered using a 5V rail and ground. Thus, negative signals could not be amplified using this configuration. This could have been circumvented by applying a DC shift to the signal however, the circuit used to perform this DC shift had potential drawbacks.

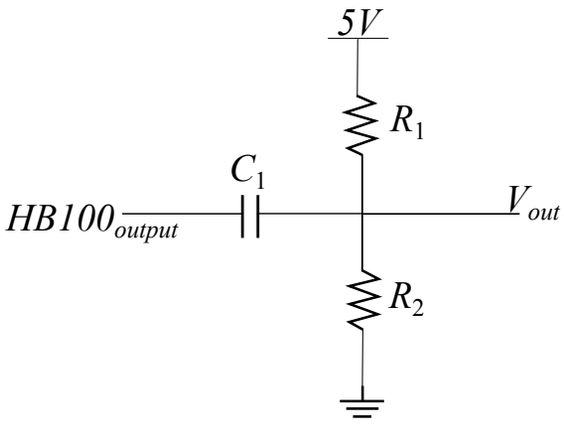


Figure 5.6: DC Shift Circuit Diagram

Firstly, the extra circuitry increased the likelihood of additional noise. Furthermore, the configuration seen in Fig. 5.6 acts as a low pass filter potentially filtering out important information.

Due to the DC shift requirement as well as the low-frequency oscillation always present in the HB100 output, the CDM324 was selected to be integrated into the final system.

5.1.2 Amplifiers

Fig. 5.5 and 5.4 both showcase that the CDM324 has a very weak output signal. In order to be sampled effectively, this signal required amplification. The first amplifier tested was the one recommended on the datasheet of the HB100 seen in Fig. 3.3.

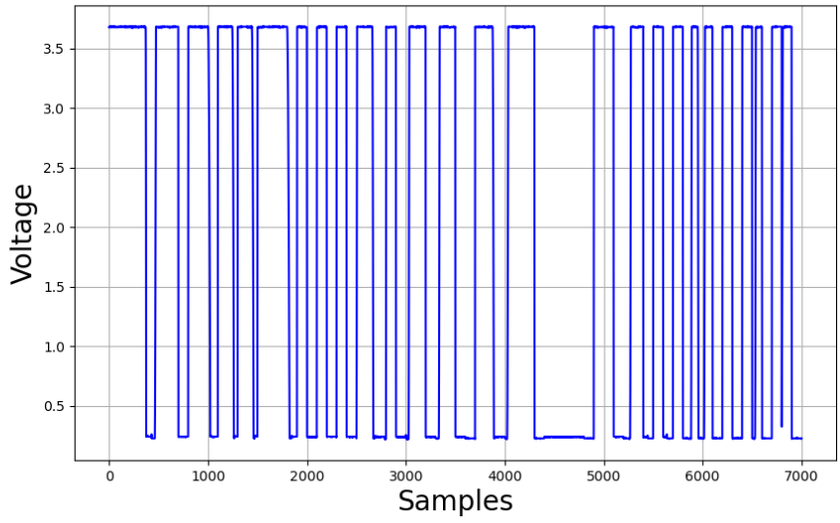
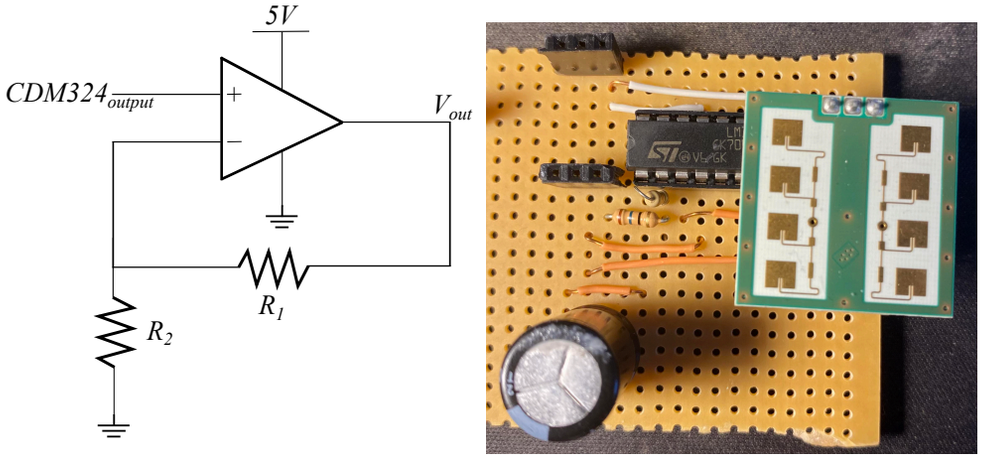


Figure 5.7: Sensor Output After Amplification

The Doppler shift observed in Fig. 5.7 was obtained by moving a hand towards and away from the sensor. Unfortunately, the output of this amplifier was also a square wave and thus the frequency content of the signal could not be observed by performing a DFT or using a spectrogram.

To account for this, a new amplifier circuit was designed. The radar signal was amplified using an LM324 non-inverting amplifier. This amplifier was able to increase the output of the radar from peak-to-peak voltages in the range of 10mV to values greater than 100mV and up to 1V when targets were close to the sensor.



(a) Non-Inverting Amplifier Circuit (b) Amplifier on Veroboard

Figure 5.8: Amplifier Design

This setup was tested on the side of a low-traffic road by pointing the CDM324 across the road and having a car drive past the sensor at a known speed of 20km/h. This setup was much like the one seen in the Literature Review in Fig. 3.7 except with only one radar. A spectrogram was used to analyze this data to observe if the vehicle's speed could be extracted from the recorded data. The Fig. 5.9 illustrates

the velocity of the car perceived by the sensor:

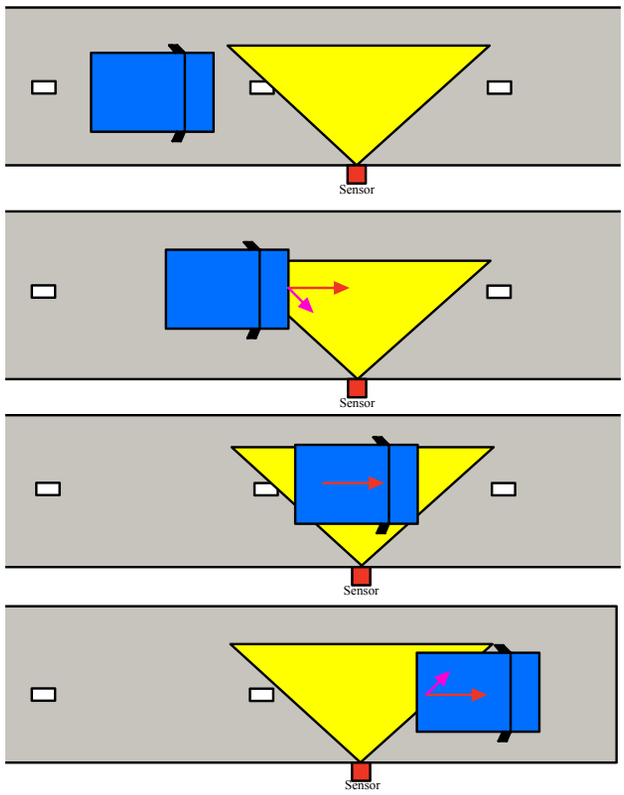


Figure 5.9: Visualisation of the Sensor Setup

In Fig. 5.9, the red arrow represents the vehicle's velocity vector whilst the pink arrow is the vehicle's radial velocity towards/away from the radar. The pink vector was expected to be at its largest when the vehicle first entered the azimuth of the sensor's beam and rapidly decreased to appear stationary when its velocity was perpendicular to the sensor's beam. The velocity would once again increase but the phase of the Doppler shift would be inverted as the vehicle would be moving away from the sensor.

The maximum velocity recorded by the sensor would be much smaller than the vehicle's actual velocity considering only a portion of the velocity vector was visible to the sensor.

The expected spectrogram obtained from this sensor setup looked as follows:

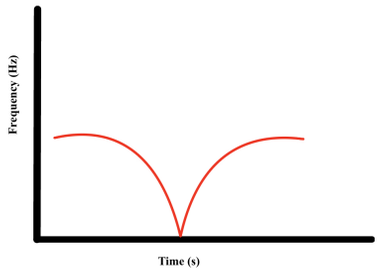
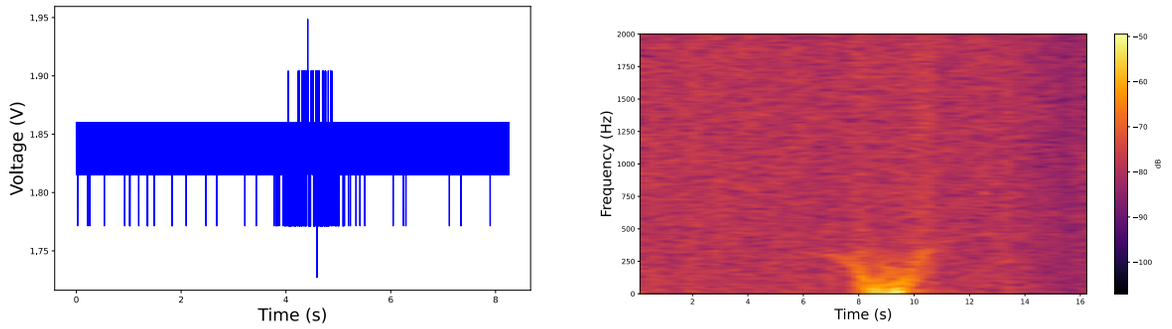


Figure 5.10: Illustration of Expected Spectrogram

In the experiment, the sensor data was sampled using a PicoScope 2000 (more details on this later in Section 5.1.3) at a sampling rate of 20ksps. This sampling rate was more than fast enough to obtain all frequency information in the signal because the Doppler shift caused by a vehicle travelling at 20km/h straight towards the CDM324 is $44.68 \times 20 = 893.6Hz$. In this case, the vehicle was not travelling directly towards the sensor so the maximum Doppler shift caused by its motion would be much less than $893.6Hz$. According to the Nyquist Sampling Criteria, a sampling rate of 20ksps can successfully sample signals up to $10kHz$.



(a) Time Domain Plot

(b) Spectrogram Obtained from Sensor

Figure 5.11: Plots from LM324 Amplifier Test

From Fig. 5.11, it can be seen that the spectrogram obtained from the sensor setup agrees with the expected plot shown in Fig. 5.10. The maximum frequency visible from the target was clearly below $893.6Hz$ which agreed with the prediction. The primary issue found with this test was that the SNR was far too weak making it very difficult to isolate the target from the surrounding noise. This was especially apparent when the target was farther away from the sensor which was the data point that was closest to the actual velocity of the vehicle. Observing the time domain plot in Fig. 5.11, the target can be distinguished from the surrounding noise. However, there is a lot of noise in the data and there is a solid band from roughly 1.82V - 1.86V which was very unexpected.

Initially, the cause of this noise was attributed to the power supply. To reduce the noise caused by the power supply, a capacitor was placed between the power line and ground. However, this had no effect on the obtained plots.

Next, an investigation into the amplifier was conducted and a low-cost LM358 Weak Signal Amplifier was purchased. This amplifier has a gain of up to $G = 100$ which is easily adjustable using a built-in potentiometer. This device was priced at $1.86 (R34.92) and was purchased with the hope that the module would have reduced noise.

Figure 5.12: JYVA2 Weak Signal Amplifier

This amplifier was interfaced with the CDM324 setup and a test was done to observe if the noise element was removed from the time domain plot. The data was generated by moving a hand back and forth in front of the sensor.

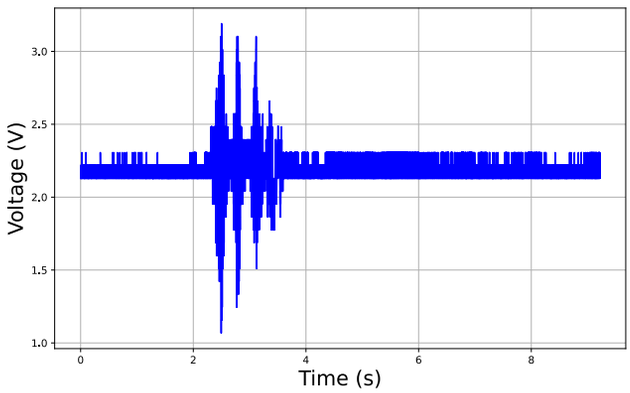


Figure 5.13: Amplifier Module Interfaced with CDM324 Plot

Once again, it can be seen that the noise elements seen in Fig. 5.11 are still present in Fig. 5.13. This prompted an investigation into the sampling tools used to obtain the data.

5.1.3 Sampling the Sensor Data

This section details the experimentation done on various sampling devices as well as processing techniques to improve the data sampled with a low-resolution ADC.

The first device tested was an ESP32 Discovery Board [24] as it has a 12-bit ADC. This was tested by sampling using the ESP32 and transmitting the samples in real time over Universal Asynchronous Receiver/Transmitter (UART). However, the sampling capability of the ESP32 was significantly hampered by the UART transmission baud rate. The maximum baud rate that could be communicated to the serial monitor was limited to 250000. This resulted in some samples being missed during communication causing sampling inaccuracies. It was found that this baud rate achieved a sampling rate of approximately

4kHz. Furthermore, the onboard memory of the board was far too small to store all of the samples and then transmit them in one package. This was an issue because firstly, the slow sampling rate of 4kHz was undesirable because radar was going to be monitoring fast-moving vehicles which could cause Doppler shifts larger than 2kHz which could not be sampled by the ESP32. Secondly, the sampling rate was inconsistent as it was dependent on the baud rate rather than the microcontroller itself.

A PicoScope 2000 [25] was also tested. These devices are portable oscilloscopes that are connected to a laptop or PC. These devices have a maximum sampling rate of 1GHz and an ADC resolution of 8 bits. The PicoScope software has a 12-bit mode which artificially increases the sampling rate by applying a moving average filter to the data. This was not used as the moving average filter removes high-frequency content from the data as explained in Chapter 2.

The sampled data could be saved as a text file directly from the PicoScope software, transmitting the data over USB, avoiding the transmission issue encountered with the ESP32. The plots in Fig. 5.11 and Fig. 5.13 were generated from samples obtained using the PicoScope. It was found that the noise visible in both of those plots was caused by the low ADC resolution of the PicoScope. The ADC on the PicoScope was unable to sample the noise accurately, leading to the thick block of noise visible in the prior plots.

In an attempt to reduce this noise, post-processing techniques were tested on the data. Firstly, the sensor setup in these tests was altered slightly in an attempt to improve the SNR. This new setup is illustrated in Fig. 5.14:

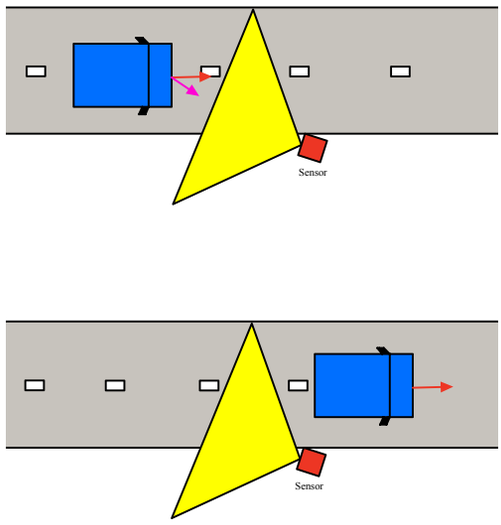


Figure 5.14: New Sensor Setup

This new configuration had the sensor placed on the side of the road, but pointing at an angle of 20° to the direction of motion of the vehicle. This was done such that the velocity of the vehicle would appear much larger to the sensor as it would be travelling almost directly toward it. Furthermore, a stand was

designed and 3D printed to hold the setup at a fixed angle such that the sensor was pointing up towards the vehicle at an angle of 20° from the ground.

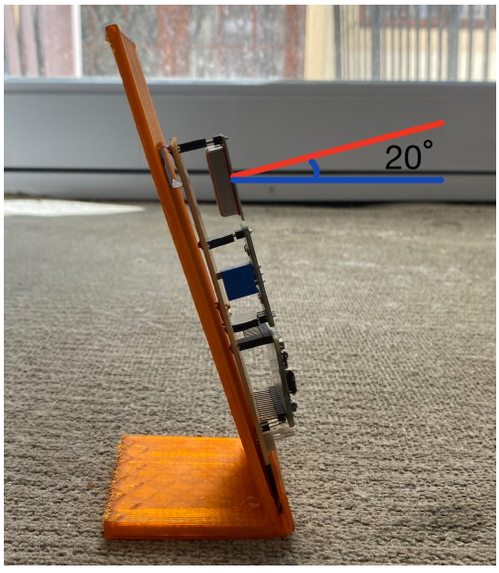


Figure 5.15: Stand Setup

This setup was once again tested in a controlled environment (a low-traffic road), where a single car was driven at 40km/h towards the sensor. The ideal and expected spectrogram from this test is shown in Fig. 5.16:

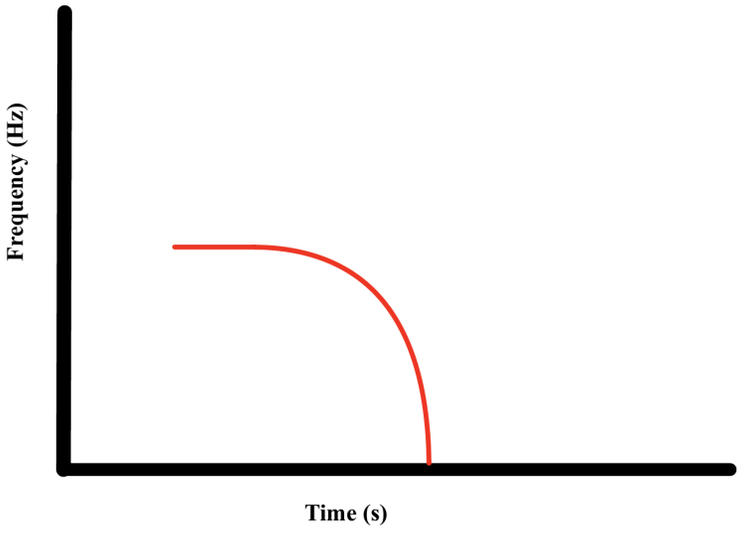
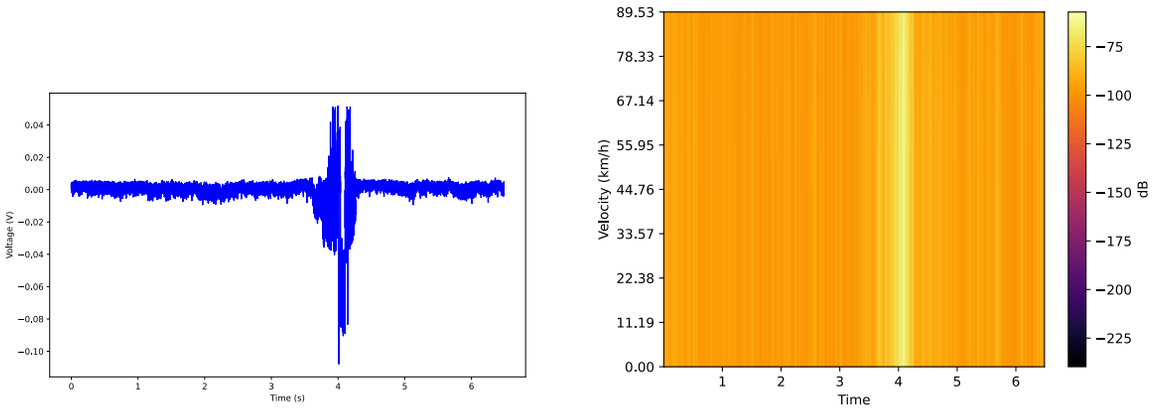


Figure 5.16: Illustration of Expected Spectrogram

This spectrogram is very similar to the one seen in Fig. 3.13 because this setup was designed to emulate the results obtained when the sensor was placed in the centre of the road. The decision to place the sensor on the side of the road was made because it seemed impractical to have the setup only function when placed in the centre of the road. This would have limited the use case to areas where there are bridges or overpasses that the sensor could be attached to. In the roadside configuration, the setup could work on any road.

The sampling procedure was also altered in an attempt to reduce the noise. Instead of sampling at 20ksps as done before, the data was sampled as 1Msps. This was done to allow for downsampling by averaging the extra samples with the intention of reducing the very high-frequency noise signals. Furthermore, the frequency axis in the spectrogram was scaled by 44.68 to obtain Velocity in km/h on the y-axis. The resulting plots looked as follows:

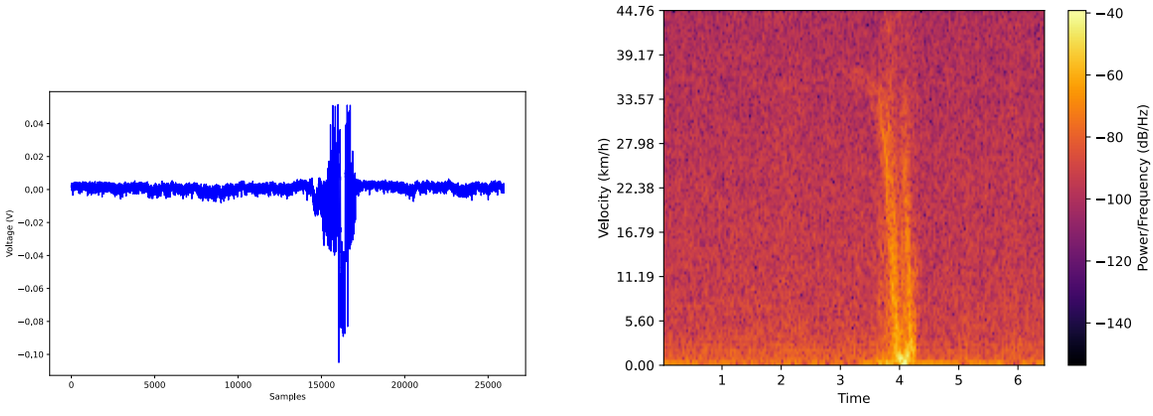


(a) Time Domain Plot

(b) Spectrogram Plot

Figure 5.17: Data Before Downsampling

From the above plots, it can be seen that the solid bar of noise that was apparent in 5.13 is no longer present and instead is more random as expected from noise. The target can also be seen in the spectrogram however the shape created by the target is not as expected. The plots obtained after downsampling are shown in Fig. 5.18:



(a) Time Domain Plot

(b) Spectrogram

Figure 5.18: Data After Downsampling

In the time domain plot, the noise was slightly reduced by the downsampling whilst the target remained the same. Furthermore, the spectrogram plot looked much more as expected and the curve shape that was seen in Fig. 5.16 could be picked out in the spectrogram. Unfortunately, the target could only be

identified when it was very close to the sensor and its radial speed was much smaller than the vehicle's actual speed. The most important region of the spectrogram, being the area at the start of the curve, cannot be distinguished from the surrounding noise easily. This was unexpected because, in this test, the sensor was placed extremely close to the vehicles driving along the road (roughly 1m away when the vehicle was at its closest). This was unusual considering the CDM324 has a range of up to 15m [26]. From this data, it was found that the PicoScope 2000 ADC resolution of 8 bits was insufficient to sample the sensor data and was leading to a very weak SNR. This prompted testing a new method of sampling using high quality ADC's found in sound cards.

The Doppler shifts caused by these sensors are in the audible range (approximately 20Hz - 20kHz) and thus could be sampled by a sound card designed for audio recordings. A sound card is a device used to record audio data from a microphone or musical instrument as well as to output audio to a headset or speakers. While not designed specifically for radar sensors, the audio input to these devices can be used to sample the data. Sound cards generally have two input types: Line In (AUX in) and Microphone In. The key difference between the two is the voltage level expected at the input. Line In requires a higher voltage input to the device (-10dB or higher) whilst Mic In is designed for lower voltages (-60dB to -40dB) [27]. Mic Inputs have a pre-amplifier built into the device in order to amplify the lower voltage signals received whilst Line Inputs have no amplifier and just sample the received data. Two sound cards were tested:

SoundBlaster G3 by Creative Labs	**Steel Sound 5HV2**
• ADC Resolution = 16 bits • Sampling Rate = 48ksps • Inputs = Mic In and Line In • Cost = $68	• ADC Resolution = 16 bits • Sampling Rate = 48ksps • Inputs = Mic In • Cost = $8.05

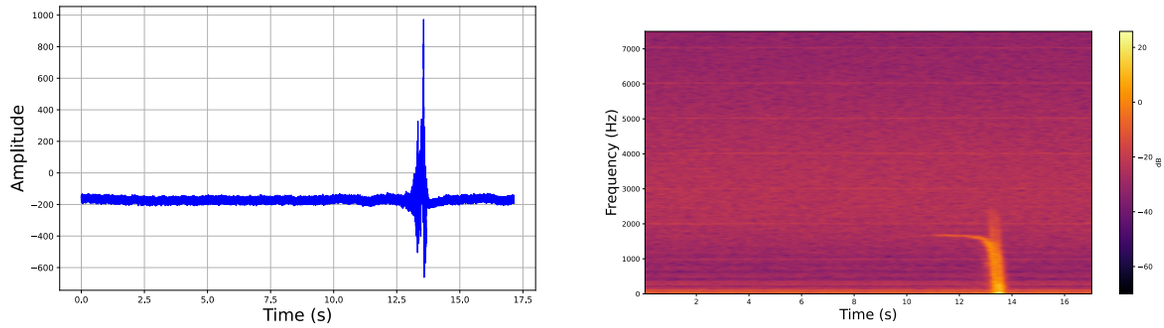
Table 5.1: Comparison of Two Sound Cards

(a) SoundBlaster G3 Image [28] (b) Steel Sound 5HV2 Image [29]

Figure 5.19: Images of Sound Cards Investigated

Whilst the Steel Sound 5HV2 is much cheaper and has similar specifications, it does not have a Line Input. Ultimately, both devices met the sampling rate requirement and have a high resolution ADC. Furthermore, the issue of transmitting the sampled data over UART was avoided as both devices are USB-powered and transmit the data over USB. This allowed for the data to be analysed in Python and had the added benefit of being compatible with audio analysis software like Audacity which was used to obtain real-time plots of the recorded data. Both devices were used to test the sensing setup.

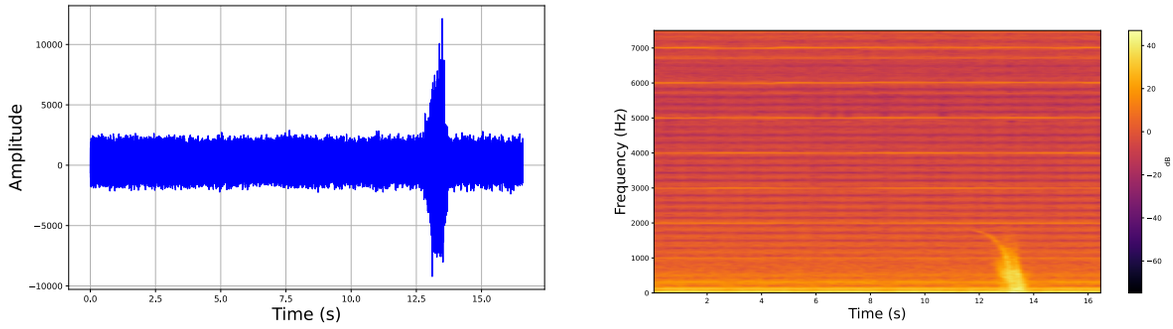
The SoundBlaster G3 was tested first, and the sensor data was sampled using its line input. The same testing setup was used as in Fig. 5.14 where the sensor was placed on the side of that road at an angle of 20° to the road. Once again, a vehicle was driven at 40km/h toward the sensor, and the following results were obtained:



(a) Time Domain Plot (b) Spectrogram Plot

Figure 5.20: Results Obtained from SoundBlaster G3 Test

As can be seen from the above plot, the target is clearly visible in the Spectrogram and the plot looks exactly like the expected plot shown in Fig. 5.16. Furthermore, the frequency curve caused by the vehicle begins at around 1600Hz which corresponds to a velocity of 40.55km/h (when accounting for the scale factor to convert from Hz to km/h and the angle of the sensor). This indicated that the sensing system was functioning correctly. The same test was conducted using the SteelSound 5HV2:



(a) Time Domain Plot (b) Spectrogram Plot

Figure 5.21: Results Obtained from Steel Sound 5HV2 Test

As can be seen from Fig. 5.20 and 5.21, the mic input on the Steel Sound amplifies the signal much

more than the line input on the SoundBlaster. This amplification also decreased the SNR significantly, lowering the quality of the obtained spectrogram. The target appears much wider in the spectrogram and its constant velocity is not clearly visible in the plot.

5.1.4 Final Component Selection

As mentioned earlier, the CDM324 was the selected Doppler radar.

In terms of amplifier choice, both the Veroboard design used in Fig. 5.8 and the JYVA2 in Fig. 5.12 achieved comparable and satisfactory results. Ultimately, the JYVA2 was selected due to its more user-friendly PCB design as opposed to the Veroboard. Furthermore, the Veroboard allows for more noise interference in the signal due to the exposed traces on the back of the board.

Additionally, the SoundBlaster G3 soundcard was selected as it had the highest quality sampling and the best SNR.

Finally, a microcontroller that was capable of interfacing with the USB soundcard and transmitting the data to a central device for analysis was required. The easiest option for this was the Raspberry Pi Zero 2 W [30]. This was because it has micro-USB ports that allow for easy interfacing with the soundcard and could run the Python scripts used in this project (detailed later in Section 5.3). The device was also low cost, priced at $21.09. Furthermore, the Wi-Fi module on the Pi allowed for wireless data transmission.

5.2 Hardware Setup

The selected components were used to develop two testing setups.

5.2.1 Setup 1

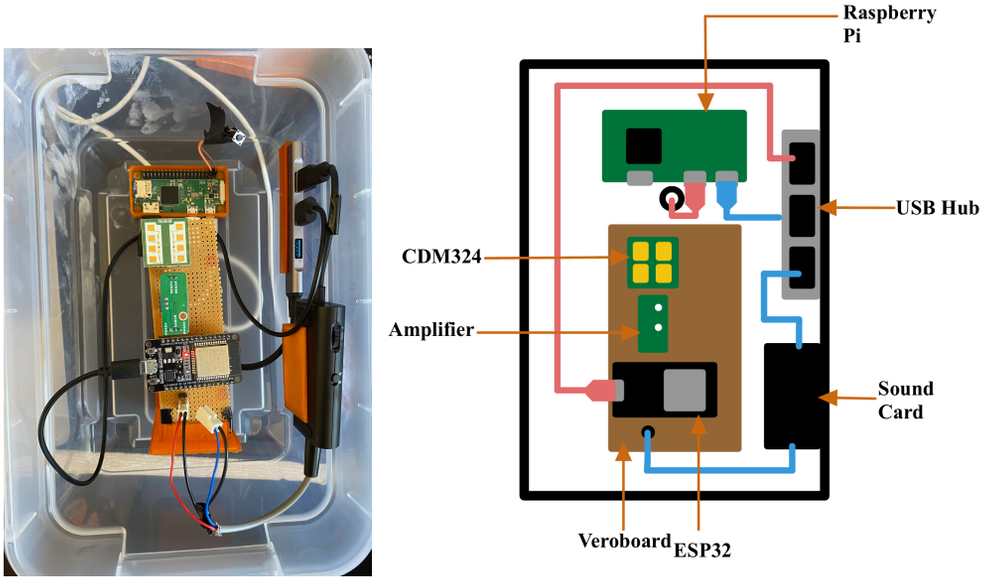
The first setup was designed purely to test the sensing setup in typical traffic scenarios (more realistic traffic situations as opposed to the controlled tests performed above) and only consisted of the following devices:

- CDM324 Sensor
- JYVA2 Amplifier
- ESP32 Development Board
- SoundBlaster G3 Soundcard
- 3D Printed Sensor Stand

The system was powered using a USB connected to a laptop and the ESP32 Development board. The ESP32 was used purely to power the Veroboard with all the components on it. This is the same system seen in Fig. 5.15.

5.2.2 Setup 2

The second system consisted of all the components selected in the previous section combined into a plastic housing. The Pi Zero was originally powered using a Lithium Ion battery however, the Pi was unstable when powered through its GPIO pins and kept restarting. Due to this, the Pi was also powered by a laptop through USB, and the ESP32 Development board was used to power the radar.



(a) Setup 2 Layout Photo (b) Setup 2 Layout Diagram

Figure 5.22: Setup 2 Layout

In Fig. 5.22, the red wires represent power lines, and the blue wires data lines. The Pi was powered externally, and through a USB Hub powered the ESP32, and interfaced with the Sound Card.

5.3 Software Design

This section details the Python scripts used to obtain and analyse the traffic data obtained using the hardware designs in Section 5.2. Setup 1 was connected directly to a laptop and the data was recorded using Audacity. Audacity is an audio recording tool that interfaced easily with the soundcard used in this design. Audacity was used to record the soundcard data and save it as a WAV file. This data was imported into Python and a data processing algorithm was applied:

5.3.1 Algorithm 1: Data Processing

The data processing algorithm was designed to identify each target in the spectrogram and obtain its velocity from the plot. The SNR of the spectrogram was initially very weak and required filtering to improve the plot's quality. Furthermore, the sensor's placement allowed for it to detect vehicles in both lanes of traffic. A pitfall was predicted regarding the scenario when two vehicles pass the sensor side by side, resulting in one occluding the other.

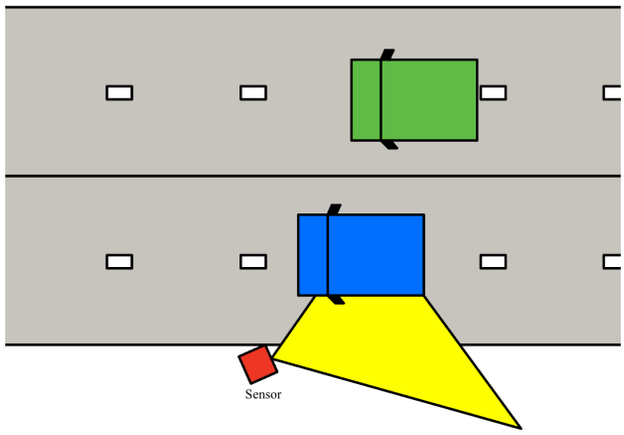


Figure 5.23: Two-Lane Pitfall Illustration

In Fig. 5.23, it can be seen that the green target in the far lane would pass the sensor completely undetected because the blue car blocks the sensor beam. To avoid this error, the sensor setup was designed to work only with a single lane at a time. To achieve this, the targets in the far lane had to be filtered out. The signals from cars in the far lane were predicted to be much weaker due to the increased distance from the radar.

The algorithm used Gaussian Smoothing with $\sigma = (0.5, 3)$, with 0.5 indicating the kernel spread in the frequency axis whilst 3, the spread in the time axis. This was done to improve the clarity of the close lane targets and to filter out the weak far lane targets. A small frequency spread was used to preserve the accuracy of the frequency axis.

After smoothing, thresholding was applied to the entire plot. This was done by removing all points with a power below a selected threshold. This threshold was selected through experimentation to be -3dB. This was done to remove any background noise from the spectrogram.

Furthermore, to remove the low-frequency noise caused by the amplifier as well as all high-frequency noise, frequencies $< 700Hz$ and $> 7000Hz$ were removed from the spectrogram. These frequencies correspond to vehicle speeds $< 17.74km/h$ and $> 177.44km/h$ which are either too slow or too fast to encounter on public roads and could thus be removed from the spectrogram.

After removing the noise, each target was isolated in the spectrogram. As illustrated in Fig. 5.14, the radial velocity of the target decreases as it approaches the sensor, however, the return power of the sensor beam increases as the distance from the target to the sensor decreases. Thus, the target's return power

was at its greatest at low frequencies. The targets were identified by finding a cluster of low-frequency, high-power points on the spectrogram and isolating the time those points appeared as well as a short time interval prior. The frequency, size of the cluster, power, as well time duration to capture the target were selected through experimentation:

- Frequency = 750Hz
- Cluster size = 10
- Power = 0dB
- Time Interval = 0.6s

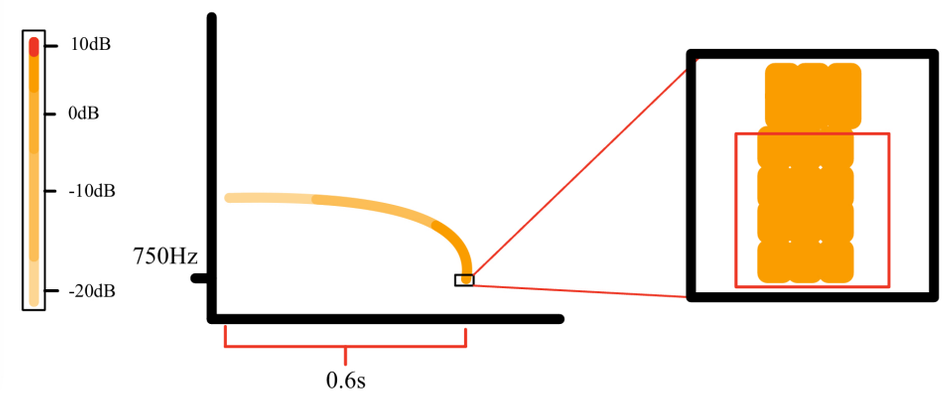


Figure 5.24: Target Isolation Process Visualisation

Once each target was isolated, the highest frequency present in each spectrogram was obtained. This frequency was converted to a velocity using the following equation:

$$Velocity(Km/h) = \frac{Frequency(Hz)}{44.68 \times cos(20) \times cos(20)} = \frac{Frequency(Hz)}{39.45} \tag{5.1}$$

Each $cos(20)$ in Equation 5.1 accounts for the angle of elevation of the sensor as well as the angle the sensor makes to the direction of motion of the vehicle.

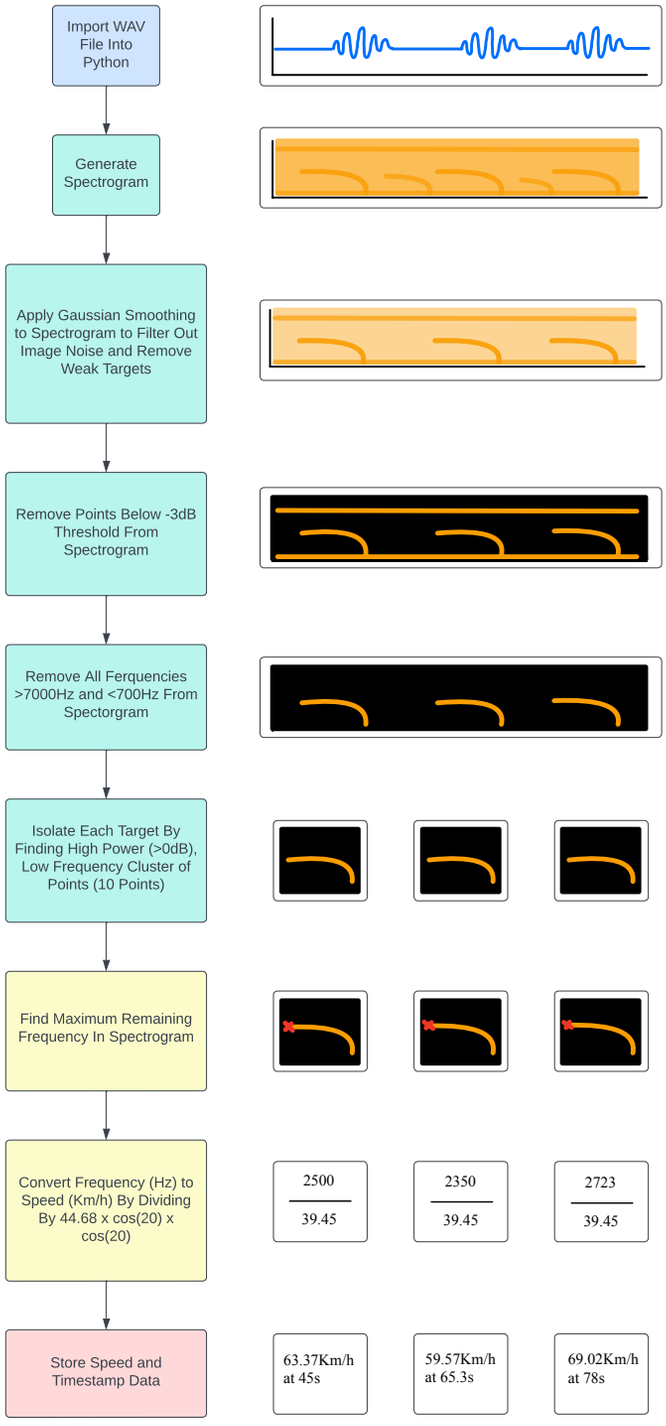


Figure 5.25: Algorithm 1 Full Visualisation

The full code described in this section can be found in Appendix 11.1.

5.3.2 Algorithm 2: Data Collection

This section details the algorithm that was run on the Raspberry Pi Zero, allowing for seamless integration with the sensor setup and data capturing. On boot, a Python script was run on the Raspberry Pi Zero which connected to the Soundcard plugged into the Pi. The script ran in an infinite while loop which recorded for a duration that could be set in the script. The recording was then saved as a WAV file with the time the recording began as the file name. This file was then transmitted to a laptop over Wifi which was running Algorithm 1. The script also had an interrupt that would trigger if a button attached to the Pi was pressed, causing the device to power off.

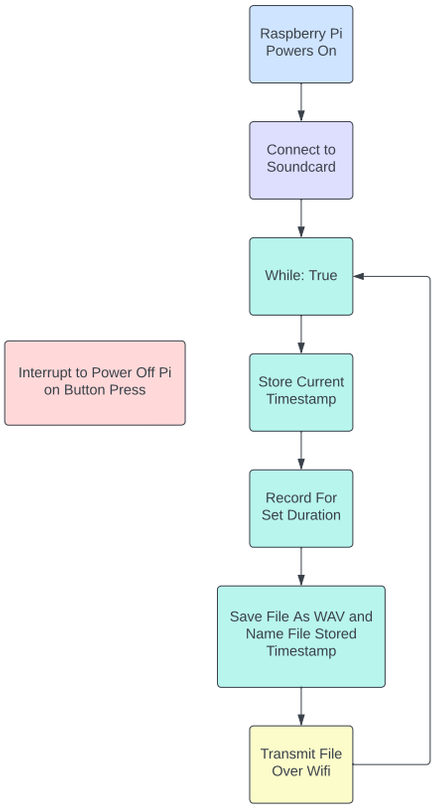


Figure 5.26: Algorithm 2 Flow Chart

The full code described in this section can be found in Appendix 11.2.

Chapter 6

Implementation

This chapter details the process of data collection for both setups seen in Section 5.2.

Both setups were tested on the side of a two-lane road, angled at 20° to the road. The sensor was placed approximately 1m away from the centre of the closest lane. Setup 1 was tested by placing the stand on the roadside.

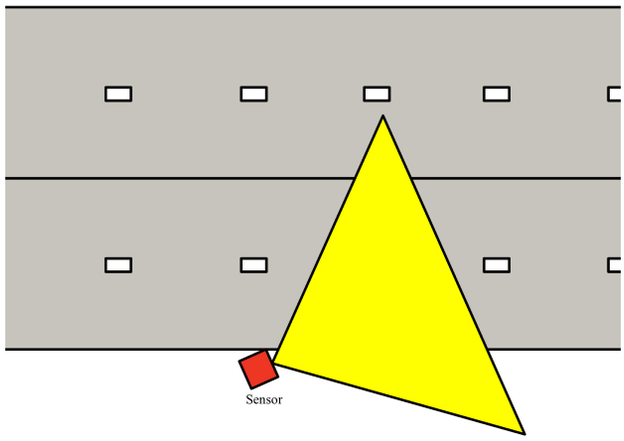


Figure 6.1: Sensor Placement

Setup 1 was tested 3 times and the first two tests were conducted on the roadside in a setup like the one seen in Fig. 6.1. The third test was conducted in front of a turn onto the road to observe the effects of this alternate sensor location.

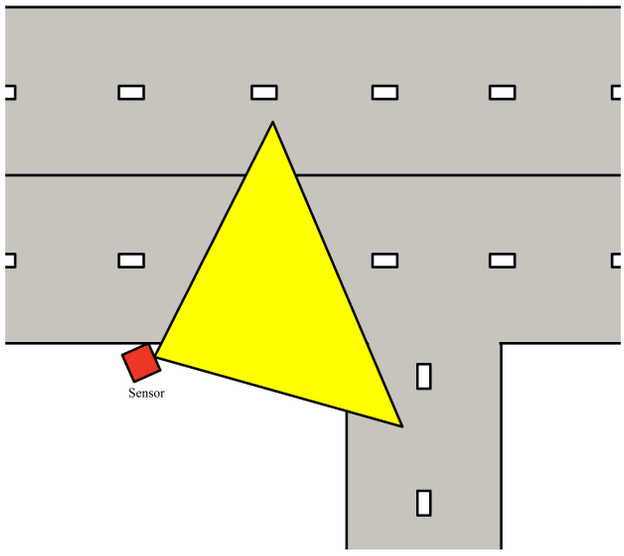


Figure 6.2: Setup 1 Test 3 Placement

Table 6.1 details the number of cars obtained in each test conducted and their respective lanes.

Test Number	Number of Cars in Close Lane	Number of Cars in Far Lane	Recording Duration (minutes)
1	20	21	2:40
2	8	3	0:53
3	7	4	1:03

Table 6.1: Setup 1 Test Details

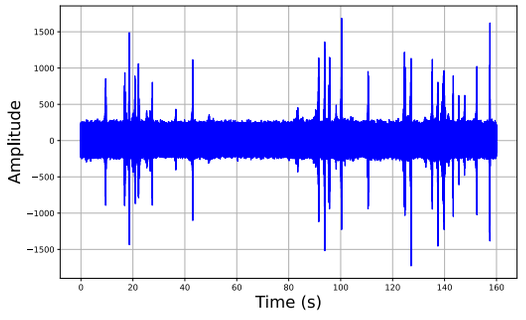
Setup 2 was tested twice, once with the lid on the box and once with the lid off of the box. These tests were done purely to observe Algorithm 2 detailed in Section 5.3 function correctly and verify that the setup worked independently. The lid was removed in the second test to observe the effect of having a lid in front of the sensor on the data. The recording duration for these tests was set to 1 minute. The road where the tests were conducted had a speed limit of 60km/h.

Chapter 7

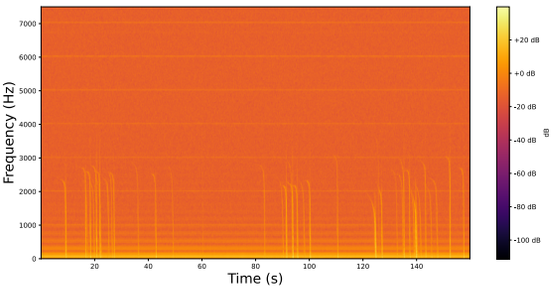
Results

This section showcases the results obtained from experiments detailed in Chapter 6.

7.1 Setup 1 Test 1 Results

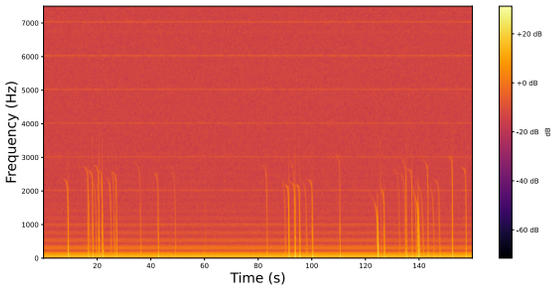


(a) Test 1 Time Domain Plot

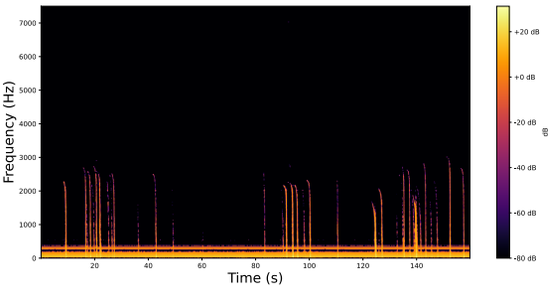


(b) Test 1 Spectrogram

Figure 7.1: Test 1 Time Domain and Spectrogram



(a) Test 1 Spectrogram After Gaussian Smoothing



(b) Test 1 Spectrogram After Thresholding

Figure 7.2: Test 1 Spectrogram After Processing

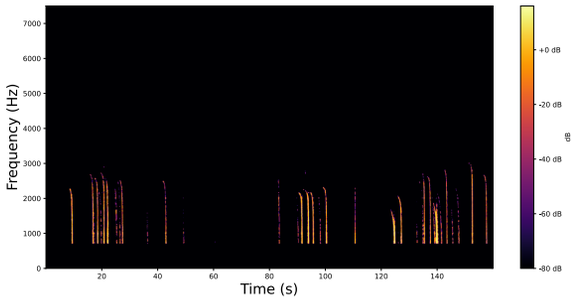


Figure 7.3: Test 1 Spectrogram Plot After Removing Frequencies Below 700Hz and Above 7000Hz

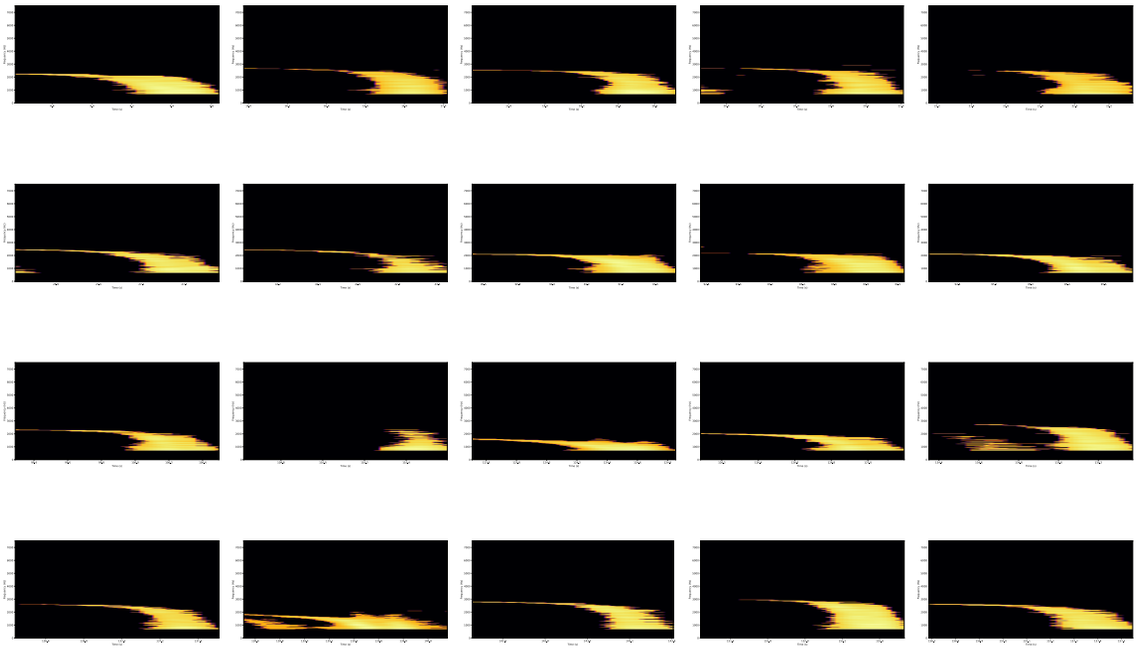


Figure 7.4: All Vehicles Isolated From the Spectrogram

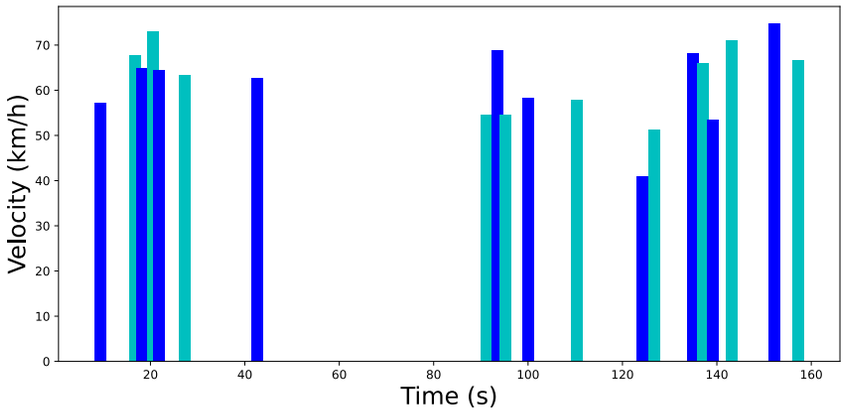
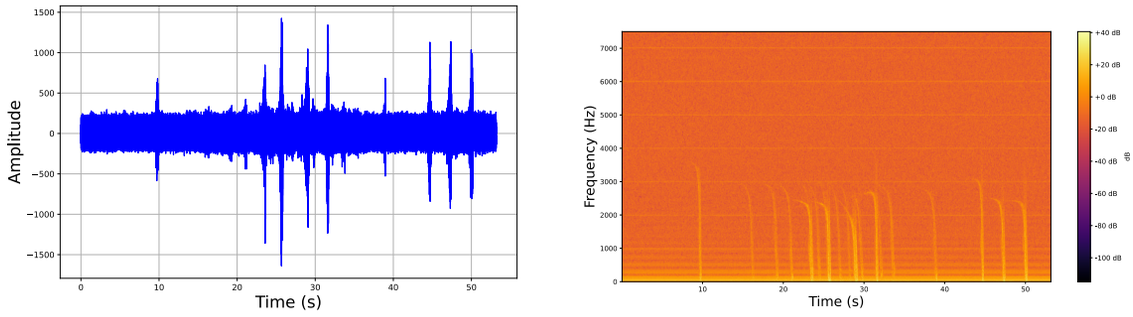


Figure 7.5: Bar Graph Showing Detected Targets and Their Speeds

A table of the data used to plot Fig. 7.5 can be found in Appendix C 11.3.

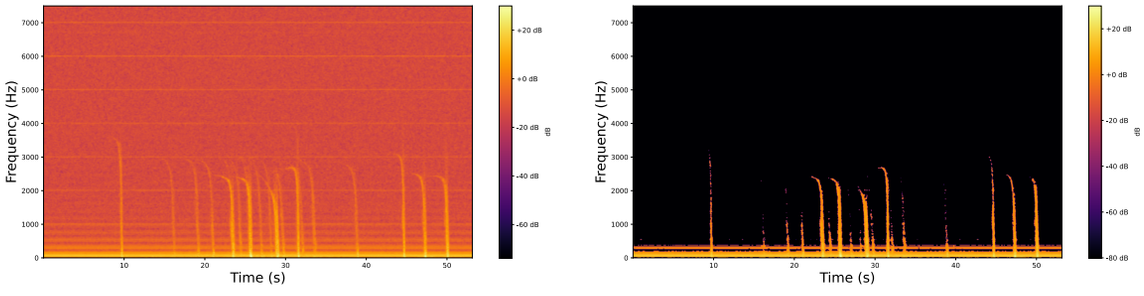
7.2 Setup 1 Test 2 Results



(a) Test 2 Time Domain Plot

(b) Test 2 Spectrogram

Figure 7.6: Test 2 Time Domain and Spectrogram



(a) Test 2 Spectrogram After Gaussian Smoothing

(b) Test 2 Spectrogram After Thresholding

Figure 7.7: Test 2 Spectrogram After Processing

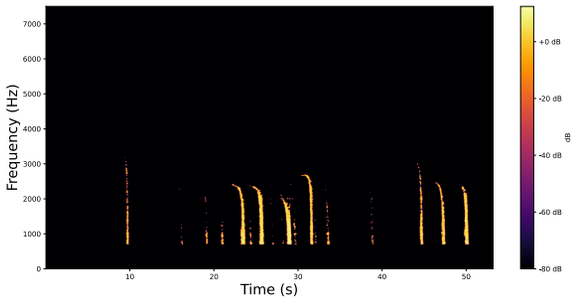


Figure 7.8: Test 2 Spectrogram Plot After Removing Frequencies Below 700Hz and Above 7000Hz

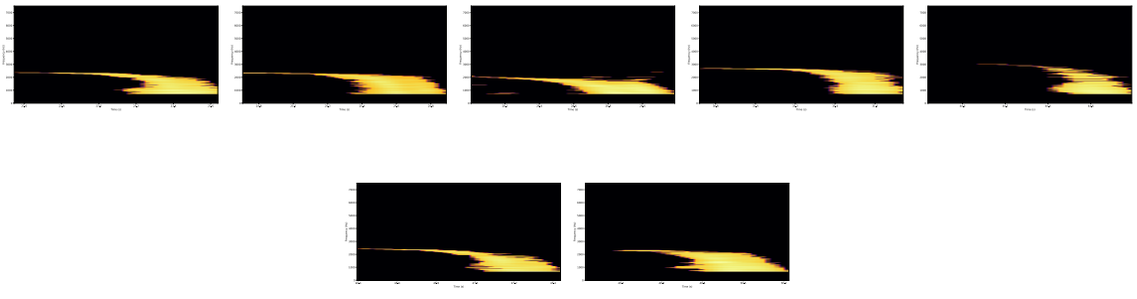


Figure 7.9: All Vehicles Isolated From the Spectrogram

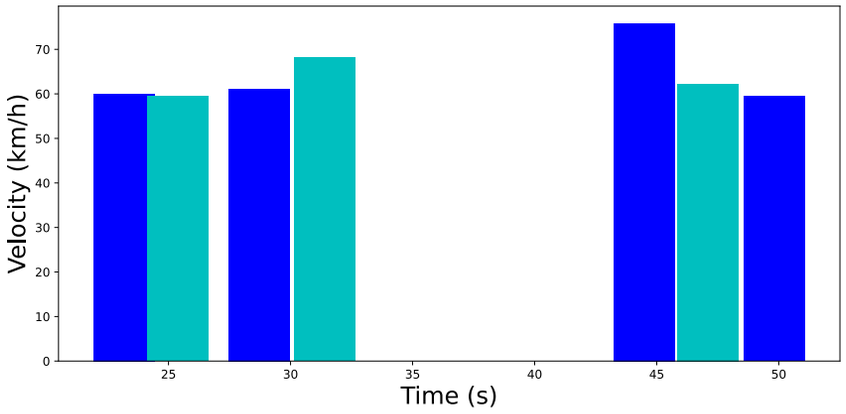
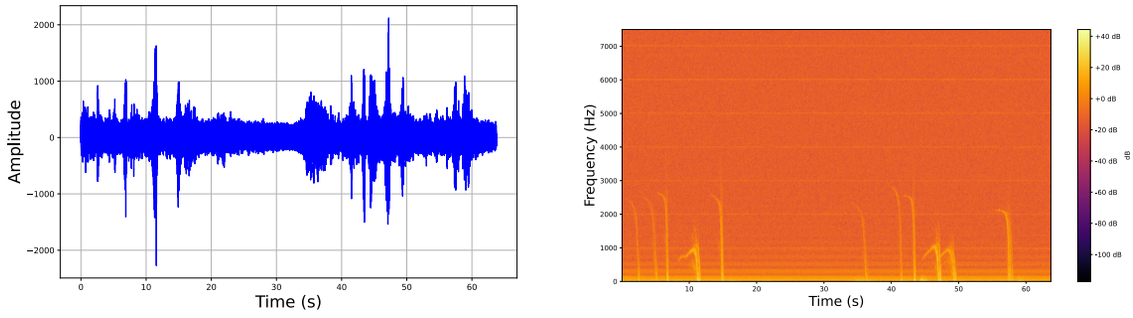


Figure 7.10: Bar Graph Showing Detected Targets and Their Speeds

A table of the data used to plot Fig. 7.10 can be found in Appendix D 11.4.

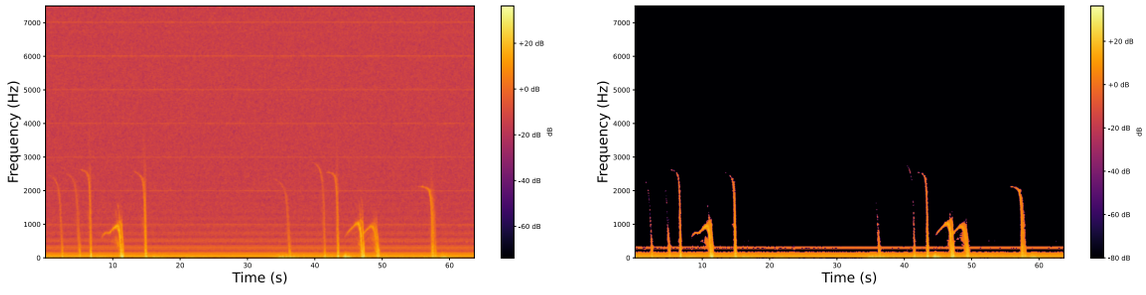
7.3 Setup 1 Test 3 Results



(a) Test 3 Time Domain Plot

(b) Test 3 Spectrogram

Figure 7.11: Test 3 Time Domain and Spectrogram



(a) Test 3 Spectrogram After Gaussian Smoothing

(b) Test 3 Spectrogram After Thresholding

Figure 7.12: Test 3 Spectrogram After Processing

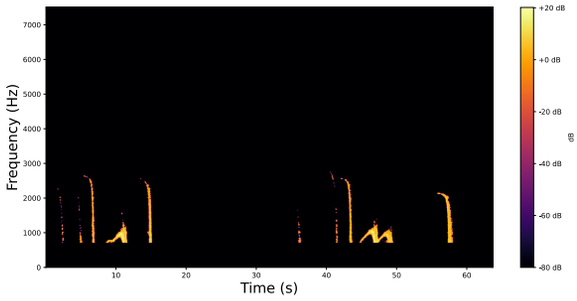


Figure 7.13: Test 3 Spectrogram Plot After Removing Frequencies Below 700Hz and Above 7000Hz

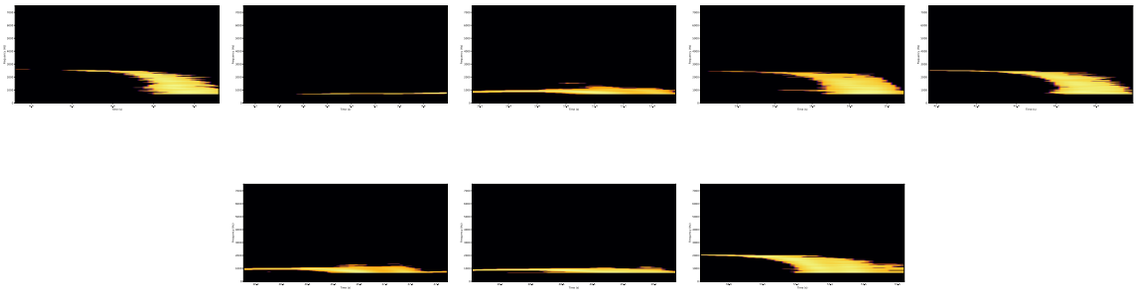


Figure 7.14: All Vehicles Isolated From the Spectrogram

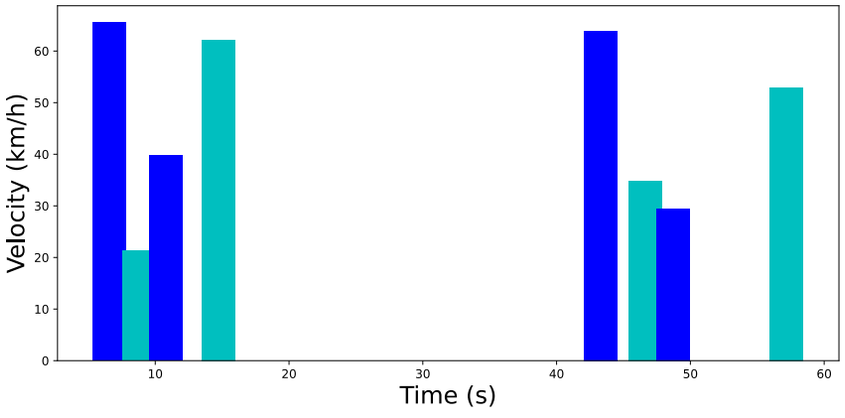


Figure 7.15: Bar Graph Showing Detected Targets and Their Speeds

A table of the data used to plot Fig. 7.15 can be found in Appendix D 11.5.

7.4 Setup 2 Results

The results from the experiments conducted using Setup 2 seen in Section 5.2 are detailed in this section.

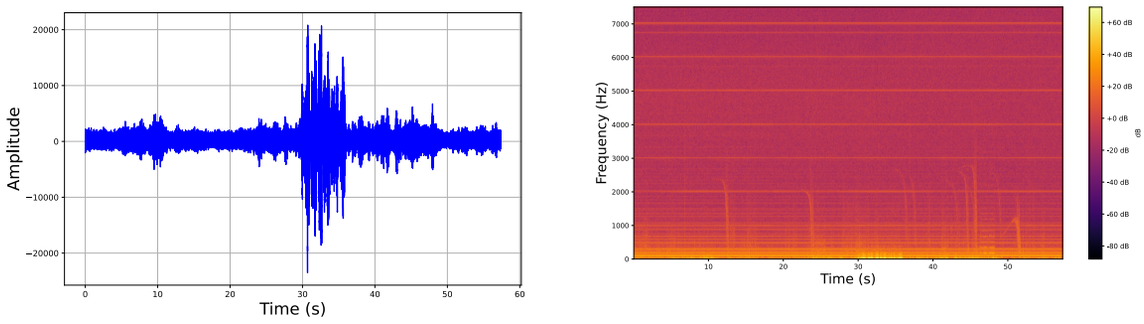


Figure 7.16: Time Domain and Spectrogram Obtained from Setup 2 with the Box Lid On

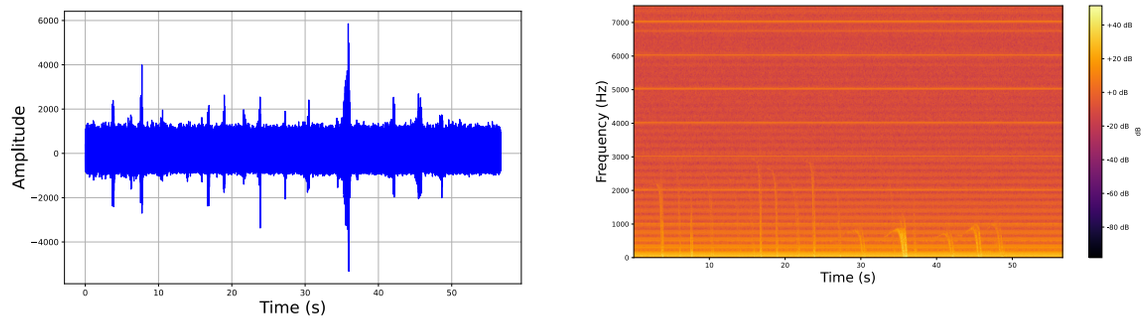


Figure 7.17: Time Domain and Spectrogram Obtained from Setup 2 with the Box Lid Off

Chapter 8

Discussion

This section is a discussion on the results obtained in Chapter 7. Some of the plots shown in that chapter will be re-plotted here to analyze some of the individual targets identified.

8.1 Setup 1 Discussion

Table 8.1 shows the number of close-lane cars successfully identified as well as the number of far-lane cars successfully filtered out by the setup.

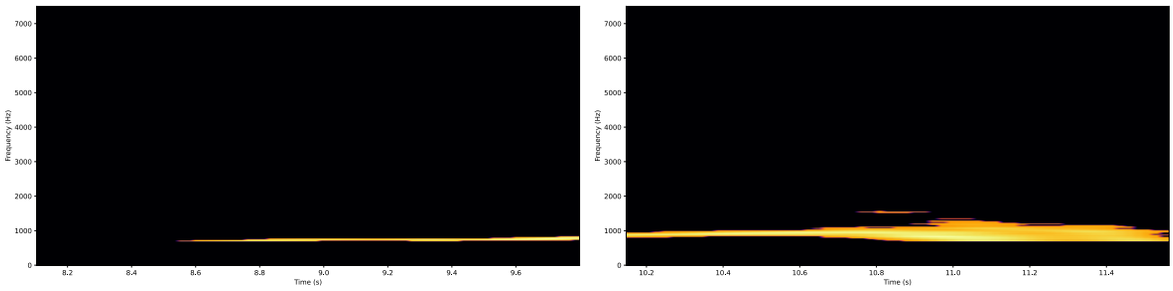
Test Number	Number of Cars in Close Lane	Number of Cars in Far Lane	Number of Cars in Close Lane Identified	Number of Cars in Far Lane Filtered Out
1	20	21	20	21
2	8	3	7	3
3	7	4	8	4

Table 8.1: Evaluation of Setup 1 Results

This table shows that the setup was able to identify all of the close-lane cars perfectly and also managed to filter out all of the far-lane cars in test 1. Algorithm 1 has performed as expected here. Observing Fig. 7.1 and 7.2 it can be seen that the Gaussian smoothing decreased the noise power noticeably. This is evident in the darker colour of the background in the smoothed spectrogram. Furthermore, removing lower frequency signals has removed a lot of extra noise as shown in Fig. 7.3.

Unfortunately, test 2 missed one of the close-lane cars. The car that was missed is the first one in the spectrogram. Fig. 7.7 shows that the first car was almost entirely removed by the thresholding, indicating a weak SNR. This vehicle appears to have been travelling significantly faster than all other vehicles observed (Doppler frequency of $3600Hz$ or a speed of 93km/h). This indicates a potential speed limitation for the system. Unfortunately, further tests are required to verify this.

Test 3 mistook a single car for two cars leading to a false positive. Three of the cars start at very slow velocities and increase before passing the radar as can be seen in Fig. 7.13. This was caused by the cars turning onto the road in front of the radar. The cars turning onto the road were also in the view of the radar for a longer period of time. This led to a false positive reading.



(a) (b)

Figure 8.1: Spectrograms of False Positive Target

Fig. 8.1 is a spectrogram of a single target, however, it can be seen that Algorithm 1 detected it as two targets. This indicates a limitation in Algorithm 1 caused by vehicles remaining in front of the radar for an extended period of time. This can be avoided by placing the sensor more strategically (away from turns/on-ramps) as was done in Tests 1 and 2. However, in high-traffic scenarios, vehicles will be travelling very slowly and will remain in front of the sensor for long periods of time. This is a current limitation of the system and Algorithm 1 must be improved to avoid this.

The average vehicle speeds obtained in Tests 1-3 are given in Table 8.2:

Test Number	Average Vehicle Speed (km/h)
1	62.01
2	63.77
3	46.25

Table 8.2: Average Vehicle Speeds in Tests 1-3

The speeds found in the table agree with the speed limit of the road which was 60km/h. Test 3 obtained much lower speeds, as expected because some of the vehicles were turning onto the road and were thus travelling much slower.

Whilst the average speed is acceptable, there were some anomalies in the speed detection. This is illustrated by observing one of the targets obtained from test 1 in Fig. 7.4:

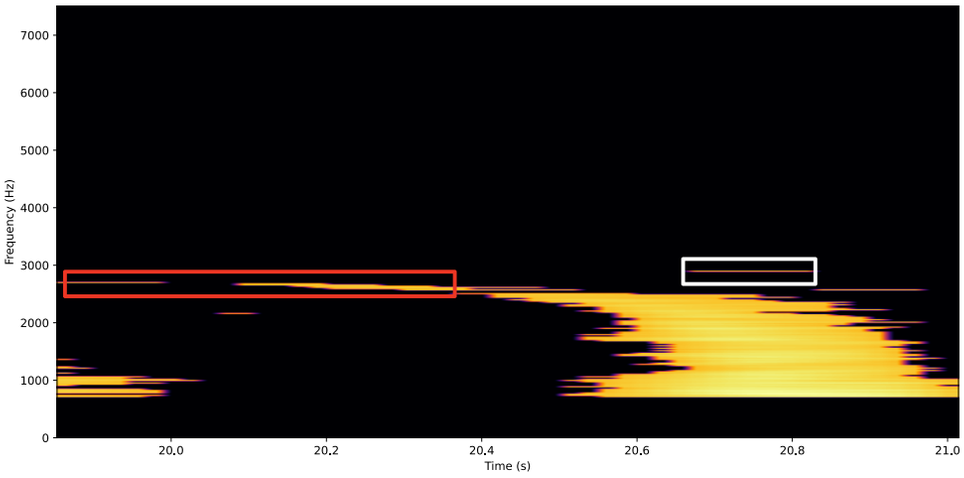


Figure 8.2: Target From Test 1

In Fig. 8.2, the red box highlights the actual velocity of the car whilst the white box highlights the highest velocity in the plot. The algorithm used assigns the highest frequency element in the plot as the velocity of the car. This is a stray frequency likely obtained by scatterings on the car as discussed in the literature review and shown in Fig. 3.13. Fortunately, these stray velocities were still created by the vehicle meaning they were generally within 10% of the actual velocity in the spectrogram. However, these are still marginally incorrect readings and could be improved.

These tests have shown that Setup 1 was able to identify the number of vehicles in a single lane of the road with an accuracy of 94%. Thus, ATP 1 (listed in Table 4.1) has been passed. Furthermore, the system was able to correctly determine the speeds of these targets with an accuracy of 10% and the average speeds in the tests agreed with the expected values passing ATP 2.

This setup was successfully able to filter out neighbouring lanes meaning that multiple systems could be placed next to each lane without interfering with one another, allowing for large roads to be monitored.

8.2 Setup 2 Discussion

Setup 2 was able to run independently on the side of a road entirely off of the Raspberry Pi Zero. It was able to record for a prescribed duration (in these tests 1 minute) and then transmit this data to a nearby laptop over Wifi. The only pitfall regarding this setup was that the Pi was not able to be powered consistently through its GPIO pins. This problem is easily fixed as will be detailed later in Chapter 9. However, this setup did require a laptop to be used as a power supply for the tests conducted. All the components in the system were able to fit in a small box as shown in Fig. 5.22.

Once again looking back at Table 4.1, this system was able to sufficiently pass ATP3. Furthermore, a USB power bank would have been sufficient to power the system but a laptop was used as a power bank was not accessible at the time. However, the system wasn't tested with the power bank and thus the time between charges was not obtained. Thus, the system cannot be said to have passed ATP4.

The system was able to run entirely on the Raspberry Pi passing ATP5.

Observing Fig. 7.16, it can be seen from the spectrogram that placing the system in a box decreased the SNR. The targets appeared much less clear in the plot. Furthermore, there are horizontal bands of noise apparent throughout the plot, especially at frequencies below 1000Hz. These horizontal bands do appear in the plots obtained using Setup 1 (refer to Fig. 7.11) but significantly less frequently.

The test performed with the box lid off obtained similar results as shown in Fig. 7.17. The horizontal bands of noise are still apparent however the background noise as well as the targets appear brighter.

These horizontal bands were likely caused by having the Raspberry Pi and its WiFi module in the same box as the radar. Furthermore, all of the power and data lines likely added to the noise seen. Whilst the system was able to transmit this captured data wirelessly, the captured data could not be passed through Algorithm 1 to identify the vehicles and their speeds due to the increased noise. Thus ATP6 was not entirely passed.

Finally, the entire system used in Setup 2 was priced at $111.29 (R2145). This is over the required price of $100, failing ATP7. The primary reason this system was over budget was due to the cost of the SoundBlaster G3. This soundcard was priced at $69 (R1299) making it the most expensive component in the setup.

Chapter 9

Conclusion

The goal of this project was to design a low-cost traffic monitoring system using Doppler radar. This device was required to count the number of vehicles travelling along a road and record their speeds. This device was also required to be low-cost, unlike most existing traffic monitoring systems on the market.

This report began with a breakdown of the theoretical concepts required to develop a radar traffic monitoring system.

Following the theoretical background, a literature review was conducted, investigating the current means of monitoring traffic. Traffic monitoring solutions applied to low-cost Doppler radars were then investigated to develop an understanding of the capabilities of these devices.

The requirements, specifications and means of testing the final system were discussed. This produced a set of goals that the system was required to meet and a means of verifying whether or not the goals were met.

Using the foundation developed in the literature review, a few low-cost Doppler radars were tested. This was followed by a design process in which a hardware system that complemented the selected radar was developed. These devices were subjected to controlled tests using cars driving at known speeds. Once the hardware system had been developed, software was designed to obtain the data using the hardware and to decode the information in the data.

The system was then subjected to the tests developed in Chapter 4. This involved applying the system in real-world traffic scenarios and observing the results.

Ultimately, the system was able to pass 4 of the 7 tests developed. It was able to count cars and record their speeds in typical traffic conditions. However, the system has numerous drawbacks and fails in some traffic conditions. These drawbacks can be reduced and means of improving the system are explored in Chapter 10. This project can be deemed a success in that it has provided a strong foundation in the journey towards developing a low-cost traffic monitoring system and a clear means of reaching this destination.

Chapter 10

Future Work

This chapter focuses on the flaws in the developed traffic monitoring system and explores methods of improving this system.

The algorithm used to identify targets in the spectrogram had one large flaw. The system failed in the event that a vehicle was in front of the radar for an extended period of time leading to a larger spectrogram. This resulted in the target being misidentified as two separate targets as seen in Fig. 8.1. To account for this, an improved target identification algorithm can be developed. Fortunately, one such algorithm was explored in the Literature Review and illustrated in Fig. 3.14. Using such an algorithm, where both the start and a subsequent lower point on the target spectrogram are used for identification will likely perform better than only using the base of the spectrogram to identify it.

The algorithm also occasionally failed with the speed extraction from the spectrogram, as seen in Fig. 8.2. The improved target identification algorithm could help avoid this issue by taking the speed obtained at the start of the spectrogram instead of taking the highest speed in the plot.

The flaws apparent in Setup 2 (discussed in Section 8.2) can be easily improved. The Raspberry Pi should be powered through its micro USB port instead of through its GPIO pins. This can be done by soldering a male Micro USB connector onto a battery module or by using a portable power bank.

Furthermore, the issue of noise from other components in the same housing can be removed by placing the power supply and Raspberry Pi in a separate housing to the radar module. Once again, a similar solution was investigated in the Literature Review 3.2.3.

The system has the potential to classify the vehicles. A method of doing this using the scatterings reflected off of the vehicle was observed in the Literature Review and seen in Fig. 3.13. Furthermore, the soundcard used in this setup has two input channels. This allows for a second radar system to be sampled using the same device.

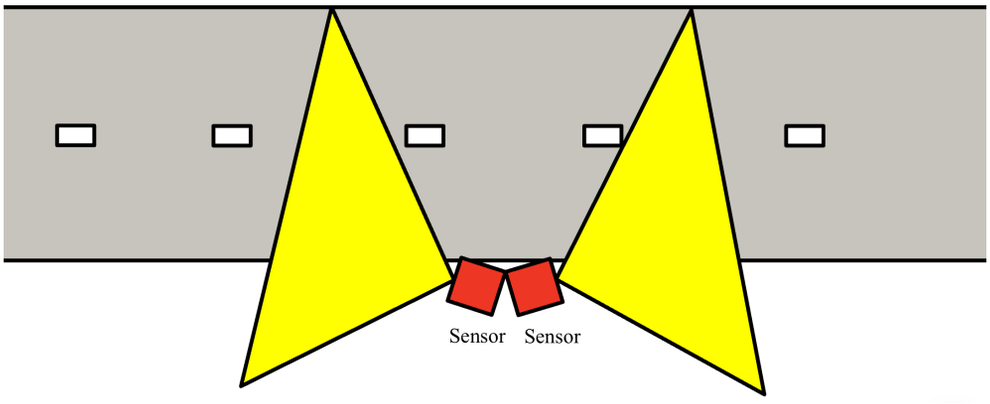


Figure 10.1: Two Radar Setup

This system will be able to detect the scatterings from the front and back of the vehicle, potentially allowing for an improved classification accuracy than the one seen in existing literature. Furthermore, the additional information from the second radar can be used to verify the results obtained from the first one, allowing for a more accurate system.

Lastly, the final system was slightly over budget. The primary cause of this was the cost of the SoundBlaster G3 ($68). To reduce this cost, a high-quality external ADC could potentially be used instead and interfaced directly with the Raspberry Pi. This would require a thorough investigation into external ADCs and methods of interfacing them with the Raspberry Pi. The ADS1262 [31] is a 32-bit ADC that can be interfaced with the Raspberry Pi and costs only $18.27 (R344.39). Alternatively, an investigation into improving the sampling quality of the low-cost soundcard (the SteelSound VHV2) using post-processing has the potential to achieve good results.

Bibliography

[1] X. Bao, Y. Zhan, C. Xu, K. Hu, C. Zheng, and Y. Wang, "A novel dual microwave doppler radar based vehicle detection sensor for parking lot occupancy detection," *IEICE Electronics Express*, vol. 14, no. 1, pp. 20161087–20161087, 2017.

[2] J. @Scicoding, "How to do spectrogram in python." "https://scicoding.com/how-to-do-spectrogram-in-python/", 2023. Accessed on October 24th, 2023.

[3] S. Prototyping, "Hb100 doppler module (102077)." "https://www.smart-prototyping.com/HB100-Doppler-Module", 2023. Accessed on October 26th, 2023.

[4] V. C. Nguyen, D. K. Dinh, V. A. Le, and V. D. Nguyen, "Length and speed detection using microwave motion sensor," in *2014 International Conference on Advanced Technologies for Communications (ATC 2014)*, vol. 1, pp. 371–376, 2014.

[5] Amazon, "Cdm324 microwave module, 24ghz 15m doppler radar smart microwave body induction module single channel switch intelligent sensor module." "https://www.amazon.com/Akozon-CDM324-Induction-Channel-Microwave/dp/B07FMQ37L7", 2023. Accessed on October 26th, 2023.

[6] Z. Liu, C. Li, H. Wang, Y. Wang, Y. Hui, G. Mao, and P. Zhao, "A novel cost-effective iot-based traffic flow detection scheme for smart roads," in *2020 IEEE/CIC International Conference on Communications in China (ICCC)*, pp. 1196–1201, IEEE, 2020.

[7] B. Sensors, "Radar movement alarm unit rsm2650." "https://static.rapidonline.com/pdf/50-7299_v1.pdf", 2014. Accessed on October 26th, 2023.

[8] T. Butterfield, "Building radar speed camera and traffic logger with a raspberry pi." "https://blog.durablescope.com/post/BuildASpeedCameraAndTrafficLogger/", 2015. Accessed on October 26th, 2023.

[9] J. Fang, H. Meng, H. Zhang, and X. Wang, "A low-cost vehicle detection and classification system based on unmodulated continuous-wave radar," in *2007 IEEE Intelligent Transportation Systems Conference*, pp. 715–720, 2007.

[10] Amazon, "Sen0192 microwave motion sensor module non-contact detection long detection distance and high sensitivity with light dc5v." "https://www.amazon.

in/Microwave-Non-Contact-Detection-Distance-Sensitivity/dp/B09FZ3PDYS", 2023. Accessed on October 27th, 2023.

[11] uRAD, "urad radar industrial." "https://urad.es/en/product/urad-radar-industrial/", 2023. Accessed on October 28th, 2023.

[12] M. A. Richards, J. Scheer, W. A. Holm, and W. L. Melvin, "Principles of modern radar," *N/A*, 2010.

[13] P. S. Thomas Zawistowski, "An introduction to sampling theory." "http://www2.egr.uh.edu/~glover/applets/Sampling/Sampling.html", N/A. Accessed on October 25th, 2023.

[14] ElectronicsTutorials, "Analogue to digital converter." "https://www.electronics-tutorials.ws/combination/analogue-to-digital-converter.html", 2021. Accessed on October 25th, 2023.

[15] F. Z. Meinard Muller, "Stft: Influence of window function." "https://www.audiolabs-erlangen.de/resources/MIR/FMP/C2/C2_STFT-Window.html#:~:text=The%20Hann%20window%20is%20a,introduces%20some%20smearing%20of%20frequencies", N/A. Accessed on October 24th, 2023.

[16] V. Velardo, "Short-time fourier transform explained easily." "https://www.youtube.com/watch?v=-Yxj3yfvY-4", 2020. Accessed on October 24th, 2023.

[17] Tektronix, "Understanding fft overlap fundamentals." "https://www.tek.com/en/documents/primer/understanding-fft-overlap-processing-fundamentals-0#:~:text=Overlapping%20transforms%20works%20somewhat%20like,of%20frequency%20changes%20with%20time.", 2023. Accessed on October 24th, 2023.

[18] 3Blue1Brown, "But what is a convolution?." "https://www.youtube.com/watch?v=KuXjwB4LzSA&t=1s", 2022. Accessed on October 24th, 2023.

[19] P. Lab, "Moving average filter - theory and software implementation - phil's lab 21r." "https://www.youtube.com/watch?v=rttn46_Y3c8", 2021. Accessed on October 25th, 2023.

[20] F. P. of Computer Vision, "Hough transform — boundary detection." "https://www.youtube.com/watch?v=XRBc_xkZREg&t=636s", 2021. Accessed on October 26th, 2023.

[21] A. Czyżewski, S. Cygert, G. Szwoch, J. Kotus, D. Weber, M. Szczodrak, D. Koszewski, K. Jamroz, W. Kustra, A. Sroczyński, *et al.*, "Comparative study on the effectiveness of various types of road traffic intensity detectors," in *2019 6th International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, pp. 1–7, IEEE, 2019.

[22] C. M. Melgoza, K. George, and J. Miho, "Comparison of cw radar systems for radar applications using object detection and real-time tracking," in *2022 IEEE 13th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON)*, pp. 0342–0346, 2022.

[23] S. Nayar, "Hough transform — boundary detection." "https://www.youtube.com/watch?v=XRBc_xkZREg", 2021. Accessed on October 29th, 2023.

[24] DigiKey, "Esp32-s2-devkitm-1." "https://www.digikey.com/en/products/detail/espressif-systems/ESP32-S2-DEVKITM-1/13180196", 2023. Accessed on October 29th, 2023.

[25] pico Technology, "Picoscope® 2000 series like a benchtop oscilloscope, only smaller and better." "https://www.picotech.com/oscilloscope/2000/picoscope-2000-overview", 2023. Accessed on October 29th, 2023.

[26] IC's and R. Gadgets, "Cdm324 24ghz microwave human body motion sensor module radar induction switch sensor." "https://www.icstation.com/cdm324-24ghz-microwave-human-body-motion-sensor-module-radar-induction-switc html", 2023. Accessed on October 19th, 2023.

[27] V. Kaul, "Line in vs. mic in (audio signals explained for dummies)." "https://producerhive.com/ask-the-hive/line-in-vs-mic-in/#:~:text=Line%2Din%20and%20mic%2Din%20are%20audio%20inputs%2C%20but,interfaces%2C%20amplifiers%2C%20and%20speakers.". Accessed on October 29th, 2023.

[28] CreativeLabs, "Soundblaster g3." "https://en.creative.com/p/sound-blaster/sound-blaster-g3", 2023. Accessed on October 19th, 2023.

[29] Takealot, "7.1 sound channel usb 2.0 steel sound 5hv2 sound card steel sound 12 channel equalizer." "https://www.takealot.com/7-1-sound-channel-usb-2-0-steel-sound-5hv2-sound-card-steel-soun/PLID44878116", 2023. Accessed on October 19th, 2023.

[30] RaspberryPi, "Raspberry pi zero w." "https://www.raspberrypi.com/products/raspberry-pi-zero-w/", 2023. Accessed on October 29th, 2023.

[31] DigiKey, "Ads1262." "https://www.digikey.com/en/products/detail/ADS1262IPWR/296-48477-1-ND/8567522?curr=usd&utm_campaign=buynow&utm_medium=aggregator&utm_source=octopart", 2023. Accessed on October 29th, 2023.

Chapter 11

Appendix

11.1 Appendix A

The code used in Algorithm 1:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.io import wavfile
from scipy.signal import spectrogram

# Load the WAV file
filename = 'openTest2.wav'  # Replace with your WAV file's path
sample_rate, audio_data = wavfile.read(filename)

# Create a spectrogram with adjusted parameters
frequencies, times, Sxx = spectrogram(audio_data, fs=sample_rate, nperseg=2048,
    noverlap=1024)

# Apply Gaussian smoothing
Sxx_smoothed = gaussian_filter(10 * np.log10(Sxx), sigma=(0.5, 3))

# Apply thresholding
threshold_db = -3  # Threshold in dB
Sxx_thresholded = np.where(Sxx_smoothed >= threshold_db, Sxx_smoothed, -80)

# Set all frequencies below 200 Hz and above 7000 Hz to -80 dB
lower_freq_limit = 700
upper_freq_limit = 7000

# Find the frequency indices corresponding to the limits
lower_freq_index = np.argmin(np.abs(frequencies - lower_freq_limit))
upper_freq_index = np.argmin(np.abs(frequencies - upper_freq_limit))

# Set the values outside the frequency limits to -80 dB
Sxx_thresholded[:lower_freq_index, :] = -80
```

```
Sxx_thresholded[upper_freq_index:, :] = -80

# Find the frequency index corresponding to the target frequency (750 Hz)
target_frequency = 750  # Hz
freq_index = np.argmin(np.abs(frequencies - target_frequency))

# Define the minimum group size and the power threshold
min_group_size = 10
power_threshold = 0  # dB

# Initialize variables to keep track of the group
group_start_time = None
group_size = 0

# Loop through the spectrogram at the specified frequency
for t in range(Sxx_thresholded.shape[1]):
    if Sxx_thresholded[freq_index, t] >= power_threshold:
        if group_start_time is None:
            group_start_time = t
            group_size = 1
        else:
            if t == group_start_time + group_size:
                group_size += 1
            else:
                if group_size >= min_group_size:
                    # Print the time range of the group
                    start_time = times[group_start_time]
                    end_time = times[group_start_time + group_size]
                    print(f"Group with {group_size} points >= {power_threshold} dB
                        at times: {start_time} - {end_time}")

                    # Plot the full spectrogram for this group
                    group_start_index = group_start_time - 30
                    group_end_index = group_start_time + group_size
                    group_spectrogram = Sxx_thresholded[:, group_start_index:
                        group_end_index]

                    # Find the indices of non-zero elements
                    nonzero_indices = np.argwhere(group_spectrogram != -80)

                    if len(nonzero_indices) > 0:
                        # Find the maximum indices for non-zero elements
                        max_indices = np.max(nonzero_indices, axis=0)

                        #print("Largest indices of non-zero values:", tuple(
                            max_indices))
                        print("max freq = ", frequencies[max_indices[0]])
                    else:
                        print("No non-zero values found in the array.")
```

```
79                      plt.figure(figsize=(12, 6))
80                      plt.imshow(group_spectrogram, aspect='auto', cmap='inferno',
81                                 origin='lower', extent=[times[group_start_index],
                                       times[group_end_index], frequencies.min(),
                                       frequencies.max()])
82                      plt.colorbar(label='dB')
83                      plt.title(f'Spectrogram for Group (Threshold > {power_threshold
                            } dB)')
84                      plt.xlabel('Time (s)')
85                      plt.ylabel('Frequency (Hz)')
86                      plt.tight_layout()
87                      plt.show()
88
89                  group_start_time = t
90                  group_size = 1
91
92  # Check if the last group meets the minimum size requirement
93  if group_size >= min_group_size:
94      start_time = times[group_start_time]
95      end_time = times[group_start_time + group_size]
96      print(f"Group with {group_size} points >= {power_threshold} dB at times: {
            start_time} - {end_time}")
97
98      # Plot the full spectrogram for the last group
99      group_start_index = group_start_time - 25
100     group_end_index = group_start_time + group_size
101     group_spectrogram = Sxx_thresholded[:, group_start_index:group_end_index]
102
103     # Find the indices of non-zero elements
104     nonzero_indices = np.argwhere(group_spectrogram != -80)
105
106     if len(nonzero_indices) > 0:
107         # Find the maximum indices for non-zero elements
108         max_indices = np.max(nonzero_indices, axis=0)
109
110         #print("Largest indices of non-zero values:", tuple(max_indices))
111         print("max freq = ", frequencies[max_indices[0]])
112     else:
113         print("No non-zero values found in the array.")
114
115     plt.figure(figsize=(12, 6))
116     plt.imshow(group_spectrogram, aspect='auto', cmap='inferno',
117                origin='lower', extent=[times[group_start_index], times[
                       group_end_index], frequencies.min(), frequencies.max()])
118     plt.colorbar(label='dB')
119     plt.title(f'Spectrogram for Last Group (Threshold > {power_threshold} dB)')
120     plt.xlabel('Time (s)')
121     plt.ylabel('Frequency (Hz)')
122     plt.tight_layout()
123     plt.show()
```

Listing 11.1: Algorithm 1

11.2 Appendix B

The code used in Algorithm 2:

```
import pyaudio
import wave
import time
import RPi.GPIO as GPIO
import subprocess

# Parameters for audio recording
FORMAT = pyaudio.paInt16  # Audio format (16-bit PCM)
CHANNELS = 1              # Number of audio channels (mono)
RATE = 48000              # Sample rate (samples per second)
RECORD_DURATION = 120     # Duration of each recording in seconds
BUFFER_SIZE = 4096

# Define GPIO pin for the shutdown button
SHUTDOWN_BUTTON_PIN = 18

# This function will be called when the button is pressed
def shutdown_button_callback(channel):
    print("Button pressed. Shutting down...")
    subprocess.call(['sudo', 'poweroff'])

# Initialize PyAudio
audio = pyaudio.PyAudio()

# List available input devices and their indices
for i in range(audio.get_device_count()):
    device_info = audio.get_device_info_by_index(i)
    device_name = device_info['name']
    print(f"Device {i}: {device_name}")

# Choose the desired input device by index
desired_device_index = 3  # Replace with the index of your chosen microphone

# Set up the GPIO for the button
GPIO.setmode(GPIO.BCM)
GPIO.setup(SHUTDOWN_BUTTON_PIN, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.add_event_detect(SHUTDOWN_BUTTON_PIN, GPIO.FALLING, callback=
    shutdown_button_callback, bouncetime=2000)

while True:
    frames = []

    # Get the current time and date for the filename
    timestamp = time.strftime("%Y%m%d_%H%M%S")

    # Open the audio stream before entering the recording loop
    stream = audio.open(format=FORMAT, channels=CHANNELS,
```

```
47                          rate=RATE, input=True, input_device_index=
                                desired_device_index, frames_per_buffer=BUFFER_SIZE)
48
49      # Record audio data for the specified duration
50      start_time = time.time()
51      while time.time() - start_time < RECORD_DURATION:
52          data = stream.read(BUFFER_SIZE)
53          frames.append(data)
54
55      # Close the audio stream after finishing recording
56      stream.stop_stream()
57      stream.close()
58
59      # Generate a unique filename with a timestamp for when recording began
60      filename = f"recording_{timestamp}.wav"
61
62      print(f"Recording complete. Saving to WAV as '{filename}'...")
63
64      # Save the recorded audio to the unique filename
65      with wave.open(filename, 'wb') as wf:
66          wf.setnchannels(CHANNELS)
67          wf.setsampwidth(audio.get_sample_size(FORMAT))
68          wf.setframerate(RATE)
69          wf.writeframes(b''.join(frames))
70
71      # Transmit the recorded file to your laptop using SCP
72      subprocess.call(['scp','-i','home/raspberry/.ssh/id_rsa', filename, '
            your_username@your_laptop_ip:/path/on/laptop'])
73
74      # Optionally, remove the file from the Raspberry Pi to save space
75      subprocess.call(['rm', filename])
```

Listing 11.2: Algorithm 2

11.3 Appendix C

Table of data used to plot Figure. 7.5:

Target Number	Target Speed (Km/h)	Time of Detection (seconds after recording start)
1	57.3	9.3
2	67.7	16.7
3	65.0	18.3
4	73.14	20.5
5	64.4	21.8
6	63.3	27.3
7	62.8	42.7
8	54.6	91.2
9	68.8	93.5
10	54.6	95.3
11	58.4	100.0
12	57.9	110.3
13	40.9	124.3
14	51.3	126.8
15	68.2	135.0
16	66.0	137.1
17	53.5	139.2
18	71.0	143.2
19	74.8	152.1
20	66.6	157.3

Table 11.1: Data Obtained from Setup 1 Test 1

11.4 Appendix D

Table of data used to plot Figure. 7.10:

Target Number	Target Speed (Km/h)	Time of Detection (seconds after recording start)
1	60.0	23.2
2	59.5	25.4
3	61.1	28.7
4	68.2	31.4
5	75.9	44.5
6	62.2	47.1
7	59.5	49.8

Table 11.2: Data Obtained from Setup 1 Test 2

11.5 Appendix E

Table of data used to plot Figure. 7.15:

Target Number	Target Speed (Km/h)	Time of Detection (seconds after recording start)
1	65.5	6.6
2	21.3	8.8
3	39.8	10.8
4	62.2	14.7
5	63.9	43.3
6	34.9	46.6
7	29.5	48.7
8	52.9	57.2

Table 11.3: Data Obtained from Setup 1 Test 3

11.6 Appendix F: GitHub Repository

A GitHub Repository with Algorithm 1 and Algorithm 2 as well as code to generate all of the data plots in this report can be found at:

https://github.com/MishayNaidoo/Traffic-Monitoring-Using-Doppler-Radar.git

11.7 Appendix G: Ethics Clearance

PRE-SCREENING QUESTIONNAIRE OUTCOME LETTER

STU-EBE-2023-PSQ000537

2023/08/06

Dear Mishay Naidoo,

Your Ethics pre-screening questionnaire (PSQ) has been evaluated by your departmental ethics representative. Based on the information supplied in your PSQ, it has been determined that you do not need to make a full ethics application for the research project in question.

You may proceed with your research project titled:

Traffic monitoring using low cost Doppler radars

Please note that should aspect(s) of your current project change, you should submit a new PSQ in order to determine whether the changed aspects increase the ethical risks of your project. It may be the case that project changes could require a full ethics application and review process.

Regards,

Faculty Research Ethics Committee