SMART TRAFFIC MANAGEMENT SYSTEM

PROJECT REPORT
submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology
in

ELECTRONICS AND COMMUNICATION ENGINEERING

of

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

by

JEFFIN GEORGE JOHNSON (RET16EC089)

A N MOHAMMED SHINAS (RET16EC116)

NAVEEN MR (RET16EC121)

NAVIN SHAJU (RET16EC122)

Under the guidance of

Ms.Shyama Sreekumar

Department of Electronics and Communication Engineering
Rajagiri School of Engineering and Technology
Rajagiri Valley, Kakkanad, Kochi, 682039

2020

RSET
RAJAGIRI SCHOOL OF
ENGINEERING & TECHNOLOGY

Rajagiri Valley, Cochin - 682 039

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE

Certified that this document titled ***"Smart Traffic Management System"*** *is a bonafide report of the project presented by* ***JEFFIN GEORGE JOHNSON (Uni.Reg.No:RET16EC089) A.N MOHAMMED SHINAS (Uni.Reg.No:RET16EC116) , NAVEEN MR (Uni.Reg.No:RET16EC121) , NAVIN SHAJU (Uni.Reg.No:RET16EC122)*** *of Eighth Semester Electronics and Communication Engineering in partial fulfillment of the requirements for the award of degree of Bachelor of Technology in Electronics and Communication Engineering from APJ Abdul Kalam Technological University, Kerala, during the academic year 2019-2020.*

Ms. Shyama Sreekumar	**Dr. Rithu James**	
Project Guide	**Head of Department**	**Project Co-ordinator**

Place: Kakkanad

Date:

ACKNOWLEDGEMENTS

We are extremely thankful to our Principal Dr. Sreejith P.S for giving us the consent for this project.

We are also extremely grateful to respected Prof.Dr Rithu James HOD(ECE) for permitting us to utilise all the necessary facilities of this institution.

We express our deep sense of gratitude to our respected guide Ms.Shyama Sreekumar and also to our seminar coordinators Mr Kiran K A and Ms.Swapna Davies for their valuable help and guidance.

Finally we would also like to thank all our dear friends for the support and encouragement they have given us during the entire course of this work.

ABSTRACT

Vehicular traffic is endlessly increasing everywhere in the world and can cause terrible traffic congestion at intersections. Most of the traffic lights today feature a fixed green light sequence, which is usually determined without taking the presence of the Emergency vehicles into account. Therefore, Emergency vehicles such as ambulances, police cars, fire engines,etc. get stuck in traffic jams and are delayed in reaching their destination. This can lead to the loss of property and valuable lives. In general, emergency service vehicles such as ambulances, fire trucks, and police cars use sirens to warn other road users for quick passage, especially for moving through traffic. However, due to well soundproofing techniques in modern cars, drivers may not be aware of the approaching Emergency vehicles, especially when the in-vehicle audio system are used. Consequently, Emergency vehicles may be blocked and may even collide with other vehicles. The proposed system can help prioritize the movement of these emergency vehicles.

List of Figures

List of Tables

Contents

Chapter - 1

Introduction

1.1 Problem Definition

The traffic light control plays a vital role in any intelligent traffic management system. The green light sequence and green light duration are the two key aspects to be considered in traffic light control. In many countries, most traffic lights feature fixed sequences and light length duration. Fixed control methods are however only suitable for stable and regular traffic, but not for dynamic traffic situations. Looking at the present state of practice, the green light sequence is determined without taking the possible presence of emergency vehicles into account. Therefore, emergency vehicles such as ambulances, police cars, fire engines, etc. must wait in traffic at an intersection as depicted in Figure 1.1 which delays their arrival at their destination causing loss of lives and property.

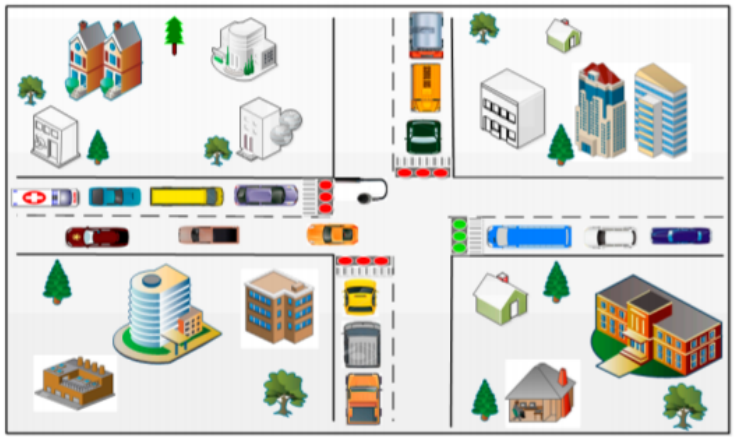


Figure 1.1: Emergency vehicle waiting at a traffic junction

An increased volume of vehicles not only increases the response time of emergency vehicles,but also increases chances for them being involved in accidents. In addition to this the emergency vehicle entering the junction at high speeds on a red light poses danger to traffic on other roads which might further result in some other accidents. As per the latest reports there where an estimated annual mean of 4500 vehicle crashes and 1500 injury crashes involving an ambulance over the past 15 years in India. This implies that there is a serious need for intelligent traffic management system for the effective management of both emergency and normal vehicles.

1.2 Objective

The objective of this project is to design and implement a unique and cost effective traffic management system to control and monitor the traffic at junctions based on real time data. The system should be able to prioritize the movement of emergency vehicles across the junction by detecting the sirens of the emergency vehicles.

1.3 Overview

The hardware of the system consists of a sound sensor (Microphone-Condenser type) which receives all incoming sound signals, a Raspberry Pi 4B which acts as the control system module, a Wi-Fi module (ESP8266) for short range wireless communication, a micro controller (TI MSP 430G2553) for controlling the traffic lights and a set of LED's acting as the traffic lights.

The sound sensor accepts all the incoming sound signals and passes it onto the control system module. The Control system determines whether the incoming signal is a siren or not. The detected sound signals are wirelessly transmitted over the Wi-Fi module to a micro controller for controlling the traffic lights. The project work started off with the study of frequency characteristics of the siren signals of various emergency signals in India. These are given below:

Emergency Vehicle	Frequency(Hz)
Police	635-912
Ambulance	770-960
Fire Engine	650-1550

Table 1.1: Frequency Range of Siren Signals

The next step was identifying a method to detect the sirens of these emergency vehicles from noisy environments. Using Artificial Neural Networks for detection was found to be a very efficient method for detecting sirens from background noises as compared to other techniques that use FFT or pitch detection algorithms. Supervised learning of the Neural Network is done using a Back-propagation algorithm in Python. This is then implemented on Raspberry Pi using two separate programs to train and evaluate the neural network and to execute real time detection.

The next step was interfacing the hardware components such as microphone and Wi-Fi module with Raspberry Pi.The next hardware component that was interfaced was the micro-controller for controlling the traffic junctions. A prototype of a traffic junction was made using a set of cardboards and LED's of red, yellow and green colours were used to model the traffic lights. The results were recorded assuming that only one emergency vehicle passes through the junction.

1.4 Major Contribution

Two separate programs for training and evaluation as well as real time detection were implemented using Python. The siren sound signals as well as the background noise were recorded and stored in two different folders namely Emergency and Non-Emergency respectively. These stored datasets are then used for training the neural network. Training is done based on a technique that sets weights of neurons by comparing the trained data with the tested data. During the training phase the recorded signals are passed through a bandpass filter and preprocessed. Each recorded files are taken as small chunks of 4410 samples (0.1 sec of data) and 34 short term features are extracted using PyAudio analysis and given as input to the Artificial Neural Network. This is followed by a set of 3 hidden layers (256 nodes each) which result in a single output. A Keras sequential Neural Network is used.A Keras Sequential model is appropriate for a plain stack of layers which can be used with multiple inputs and multiple outputs.

These extracted features are then passed through a neural network and certain weights were assigned. These weights obtained for training and evaluation are then compared and adjusted whenever there is a change in the trained signals and reference signals. A Confusion matrix can be made to evaluate the performance of the Neural Network (Figure 8.1). This is obtained from the sklearn Python library. These trained values are converted to HDF5 binary data format and stored in a .h5 file.

The audio signal captured using a microphone is sent wirelessly to a Raspberry-Pi without any loss of data. For that a unidirectional microphone should be used for increased precision. For the purpose of demonstration a condenser microphone is used here. The audio signal is amplified by using an LM386 Opamp (circuit in Appendix 1). This amplified analog data is then converted to digital data using the inbuilt ADC pin in the WiFi module.

Arduino Nano can be used for programming the ESP8266(12E) WiFi module which acts as an access point. Raspberry Pi is connected to the WiFi module. ESP 8266 converts the analog audio signal to digital signal using the inbuilt ADC pin and sends it to Raspberry Pi in 50ms. The Raspberry Pi detects the presence of siren in this signal and also predicts a probability, which is used for controlling traffic signals.

The audio data used for training is a 16bit data, which is sampled at a rate of 44.1kHz. This cannot be transmitted by a WiFi module. For that an external ADC is also required.Resampling and mapping techniques are used to solve this issue.A potential divider circuit is also used at the output of Opamp to reduce the output voltage to 0-1V.With the inbuilt ADC of WiFi module a sampling rate of 5kHz is achieved, which is sufficient enough to transmit emergency sirens lying in the range of 0.4kHz-2.5kHz. Also 10 bit data is mapped to a 16 bit data and then stored as a string. It is then transmitted by the WiFi module to the Raspberry-Pi using HTTP "text/plain" format.Sampling frequency is calculated using the time taken for reading n samples.

After the comprehensive training and evaluation of the Artificial Neural Network, the program is implemented on Raspberry-Pi Model 4B running the latest version of Raspbian OS. The Raspberry Pi 4B Computer with 2GB of RAM is used and is setup using NOOBs.

The Raspberry-Pi is initially connected to the Wi-Fi module which acts as an access point. The transmission of audio data begins when the Raspberry-Pi sends a url request using "urllib.request" to receive data from the IP address of the Wi-Fi module. Features are extracted from the re-sampled data and compared with the trained model network. Feature extraction from the re-sampled data is done using "Pyaudio analysis". Background noises like pedestrian noise, traffic horns and other weather related noise are successfully identified as Non-Emergency Sounds while the unique Emergency Vehicle Siren is accurately detected. A threshold for the probability values can be set to improve the accuracy of the system. If the probability value is above the set threshold, a successful detection can be made.

A series of 3 or more consecutive successful detections can be used to confirm the presence of an Emergency Vehicle which can be followed by sending an interrupt signal from GPIO pins of the Raspberry-Pi to the Traffic Junction Micro-Controller which controls the traffic signals in a loop.

After interfacing the hardware components to the Raspberry Pi ,the data from the microphone is wirelessly transmitted to the Raspberry pi using the WiFi module.The detection of the Siren of emergency Vehicle and the manipulation of traffic lights is done by examining the data.

In Raspberry Pi, the detection of Emergency Vehicle is done by the Neural Network Model, which gives a numerical value between 0 and 1 based on the closeness of the data to the trained value of the Emergency Vehicle. If the value exceeds the threshold value of 0.5, it is detected as an Emergency vehicle and if this value exceeds the threshold value continuously, it sends a signal to the microcontroller. The microcontroller used for controlling traffic lights is MSP430G2553 TI . The Reset pin of microcontroller is kept high by means of a 10K resistor.Shift registers are also used in line with the microcontroller to control the traffic lights.When the signal from the Raspberry pi is received by the microcontroller, an interrupt occurs and the traffic lights turn green in the direction from which the Emergency Vehicle is approaching while the traffic lights across the other junction turn red . After the departure of Emergency vehicle the traffic lights return to their previous state.

1.5 Summary

The first chapter discusses the need for the proposed system followed by objectives which the system has to fulfill as a solution to the problem defined. Next a brief overview of the hardware and software components is given.

Chapter - 2

LITERATURE SURVEY

Alerting sounds such as emergency vehicles, smoke alarms and medical monitoring alarms are of a special importance, as they are usually designed to warn people of hazardous situations, even when these are out of sight. Nowadays, the modern cars are designed with excellent ability of soundproofing, so drivers may not listen to siren sounds of approaching emergency vehicles. This could lead to the delay in providing emergency services or even traffic accidents due to inappropriate communication and cooperation. Various studies and researches have been made in this field aiming to reduce these fatalities. Some of the siren detection sytems are listed in this chapter.

2.1 Ambulance Siren Detector using FFT

Takuya Miyazakia *et al* [3] proposed a system of Siren Detection based on Fast Fourier Transform. In this system FFT is employed twice for feature extraction. It uses dsPIC microcontroller and other Frequency analysis tools for siren detection. This method seems to be very efficient even under Doppler effect. One of the main drawbacks of this system is the significant time delay in detecting siren sound signal.

2.2 Ambulance Siren Detection by Longest Common Subsequence

Jiun-Jian Liaw *et al* [4] presented a system of Siren Detection which uses the sequence of frequency change in the siren sound signal. In this method the input sound signal is divided into is divided into certain segments called frames. The Longest Common Subsequence(LCS) is used to compare the arrangement of frequencies in the frame. The score of LCS is computed using Fuzzy logic and is used to detect the siren of the emergency vehicle.

2.3 Traffic Management for Emergency Vehicle Priority Based on Visual Sensing

Kapileswar Nellore *et al*[7] proposed an approach to schedule emergency vehicles in traffic which combines the measurement of the distance between the emergency vehicle and an intersection using visual sensing methods, vehicle counting and time sensitive alert transmission within the sensor network. This system uses acoustic sensors to collect siren signals and passes these signals onto a frequency measuring controller. The microcontroller measures these frequencies and computes its average.

2.4 Detection of Ambulance Siren Sounds Using Neural networks

Van-Thuan-Tran *et al* [2] presented a system of ambulance siren detection based principles of machine learning. This system mainly includes detection of siren signals using multi-layer perceptron neural network system and a sinusoidal model system. Their experiments show that the proposed method can achieve the identification accuracy of 90% in simulated -15dB noisy sound data and 93.8% in real sound data recorded on a downtown street.

Chapter - 3

SYSTEM OVERVIEW

The complete system can be realised in terms of its hardware and software components.The block diagram can be realised as shown below:

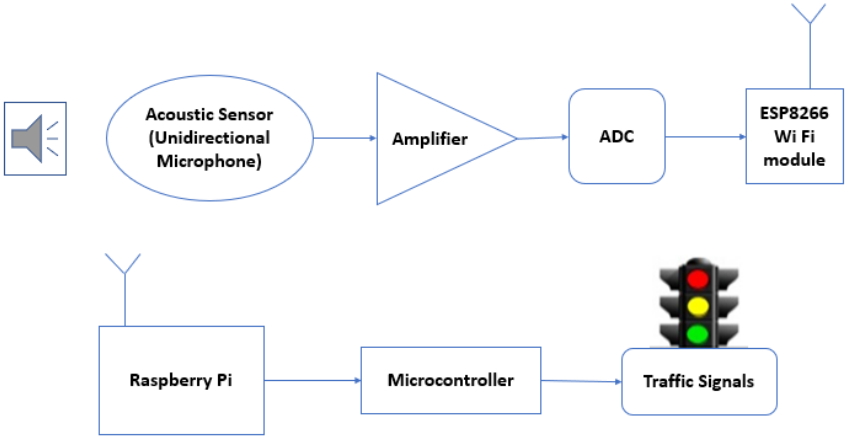


Figure 3.1: Schematic of Smart Traffic Management System

3.1 Hardware

The hardware components of the system are:

1. Microphone - Condenser Type
2. Raspberry Pi 4 Model B
3. Micro controller - TI MSP 430G2553
4. Wi-Fi module - ESP-8266

The microphone accepts all the incoming sound signals and passes it on to the controller module (Raspberry Pi). The controller decides whether the incoming signal is that of an ambulance siren or not.The signal is then transferred onto micro controller through a Wi-Fi module. Based on the received signal the micro controller controls the traffic junction.

3.2 Methodology

The software side of this project mainly deals with the detection of siren of emergency vehicles.This can be performed using two methods namely:

1. Detection using Fast Fourier Transform
2. Detection using Artificial Neural Network

3.2.1 Detection using Fast Fourier Transform

The siren of an ambulance has a varying frequency which follows a certain unique pattern.We take the FFT of these signals and look for the presence of these patterns.

The first step is analysing the pitch frequencies. The second step is analyzing the amplitude change frequencies about each pitch frequencies. Both of the two steps used FFT to analyze the frequencies. In first step, the input signal is sound pressure sampled by a microphone. It is sampled in the sampling rate of 2,400Hz and is applied 64 point FFT. Therefore the amplitudes and phases are obtained about each 37.5 Hz bandwidth. In second step, the input signals are numerical values of amplitudes obtained in first step. For each band, the amplitude is sampled in the sampling rate of 2.34 Hz and is applied 16 point FFT. Therefore the amplitudes and phases of amplitude change are obtained about each 0.14 Hz bandwidth. The block diagram is shown below.

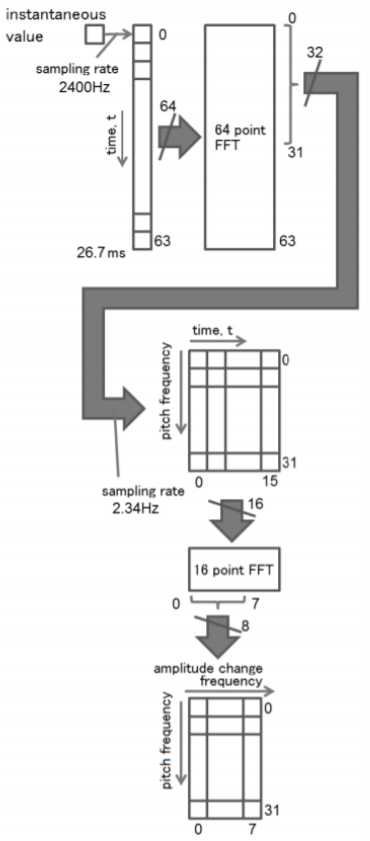


Figure 3.2: Detection using FFT

However this method is less accurate when compared with machine learning methods. There is also significant time delay of 8s when it comes to detection of signals. Moreover noise cancellation is ineffective when compared with neural networks. Hence the second method is preferred over this method.

3.2.2 Detection using Artificial Neural Networks

This method mainly involves Machine learning algorithms. This includes the use of Artificial Neural Networks which are a series of algorithms that recognizes a set of relationships in a certain set data and mimics the way a human brain operates.When compared with the previous method this method has very high accuracy. This method is has a very low time delay around 200ms. Moreover this method is not affected by Doppler effect.

Chapter - 4

SIREN DETECTION USING ARTIFICIAL NEURAL NETWORKS

Siren is a special signal sounded by alarm systems or emergency service vehicles such as fire trucks, police cars and ambulances. When an emergency vehicle performs its task, siren sound is issued to alert other drivers of an officer's need for the right of way on the road. In this section we focus on the detection of siren sounds from ambulance in which the operation modes are basically different from those of other vehicles. The spectrum of an ambulance siren is as shown below: In this project the siren of the ambulance

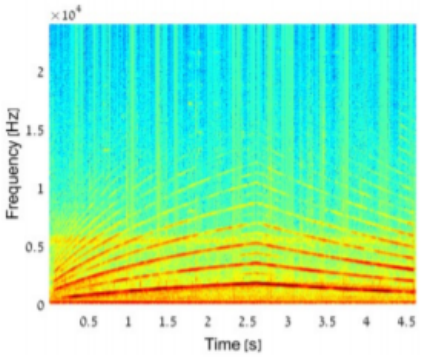


Figure 4.1: Frequency spectrum of an Ambulance Siren

or emergency vehicle is detected by the Raspberry Pi based on the method of Artificial Neural Networks. This method consists of a series of algorithms which recognises the relationship in a certain set of data and mimics the way the human brain operates.

4.1 Feature Extraction From Siren

Selection of features is a key issue when designing an effective signal detector.Each time frame, requires features that comprise the essence of information relevant to classification, are robust to variations in the signal and to noise, and are efficient in terms of computational and space complexity. To find such features, a large set of features that were previously used for different tasks of audio processing are used. These are divided into 3 categories:

- Time domain features
- Frequency domain features
- Wavelet-based features

4.1.1 Time Domain Features

Time domain features include:

- Pitch: Pitch can be defined as the sensation of frequency. A high pitch sound corresponds to a high frequency sound wave and a low pitch sound corresponds to a low frequency sound wave.
- Short Time Energy: Siren sounds tend to have high energy in the given spectral band compared to the energy attributed to noise.
- Zero crossing rate (ZCR): The number of times the sign of a time series changes within a frame. It roughly indicates the frequency that dominates during that frame.

4.1.2 Frequency Domain Features

- MFCC:MFCC or Mel-Frequency Cepstral Coefficients are used to describe the spectral shape of the signal. These coefficients are extracted by applying the discrete cosine transform (DCT) to the log-energy outputs of the nonlinear mel-scale filterbank.
- Spectral flux: Spectral flux is a measure of how quickly the power spectrum of a signal is changing, defined as the Euclidean distance between the spectra of two adjacent frames. The spectral flux can be used to determine the timbre of an audio signal, or for onset detection.

4.1.3 Wavelet-based features

The wavelet coefficients capture time and frequency localized information about the audio waveform. The wavelet transform can be viewed as a multi-level process of filtering the signal using a low-pass (scaling) filter and a high-pass (wavelet) filter.

- Discrete Wavelet transform (DWT): In the DWT, each level is calculated by passing only the previous wavelet approximation coefficients through discrete-time low and high pass filters.
- Wavelet packet transform (WPT): In the WPT, both the detail and approximation coefficients are decomposed to form a full binary tree.

In this method we have used frequency domain features such as MFCC's for siren detection. MFCC's are computed from its spectral representation by setting the upper and lower cutoff frequencies to 600Hz and 1625Hz respectively, since the siren sounds of ambulance have frequencies range from 650 Hz to 1550 Hz.

4.2 Siren Sound Detection

The Siren Sound identification (SSI) system was created by using neural networks including Multi-layer perceptron (MLP) or Long Short Term Memory Recurrent Neural Network (LSTM-RNN),which are trained using the Back Propagation (BP) algorithm and back propagation through time (BPTT) algorithm respectively. In each of the network,the MFCC's vector is used as input of the first layer which connects to a hidden layer in front of an output layer with 3 nodes. MLP is a simple form of feed forward neural network and is easy to implement.
The implementation of a typical neural network for SSI includes three steps:

- Training
- Validation
- Testing.

4.2.1 Training

In the training stage, the extracted MFCC features of training dataset are provided as input to estimate the weight parameters in a neural network.

4.2.2 Validation

In the validation phase, we use the validation dataset to evaluate the performance of trained model, and through this basis to tune the parameters to achieve the best one.

4.2.3 Testing

In the testing phase we pass the unknown audio signal to the fully-trained network to obtain corresponding output, the output with maximum response value is selected as identified result.

Structure of Siren Signal Identification Network is as shown below:

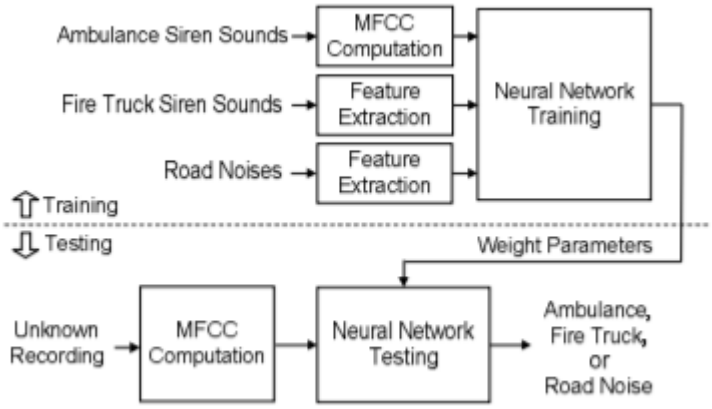


Figure 4.2: Siren Signal Identification System

4.3 Implementation In Python

The above method is implemented in Python version 3.6.8 using API's such as Tensorflow, PyAudio etc.Two separate Python programs were implemented:

- To train and Evaluate the Neural Network
- To execute Real Time Detection

Two separate sets of data were also used for training and evaluation titled Emergency and Non Emergency sound signals. The dataset is pre recorded for training. 70% of the dataset used for training 30% for evaluation of sound signals.

Chapter - 5

SYSTEM HARDWARE

The various hardware components that are used are given as follows:

1. Microphone
2. Raspberry Pi 4 B
3. TI MSP 430
4. ESP 8266
5. LM 386
6. LM 317

5.1 Microphone

Figure 5.1: Unidirectional Microphone

Microphone is a transducer that converts sound to an electronic signals. It accepts all the incoming sound signals and transmits it to the controller. Thus microphone is used as a receiver for sound signals. A unidirectional Condenser type microphone is used in our setup.

5.2 Raspberry Pi 4

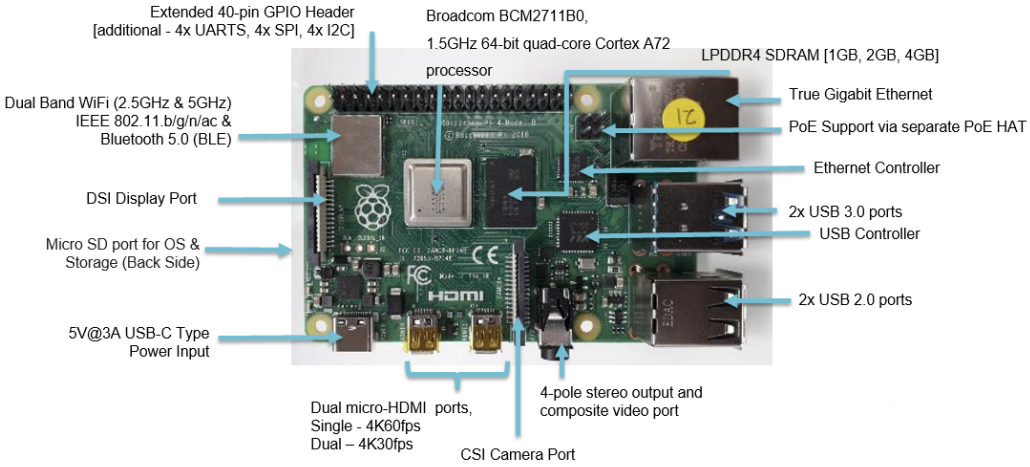


Figure 5.2: Raspberry Pi 4

The Raspberry Pi 4 is the brain of the system and does all the processing involved. It consists of 64 bit quad core ARM Cortex A72 processor running at 1.5GHz.In addition to this it also consists of on-board 802.11ac Wi-Fi, Bluetooth 5, full gigabit Ethernet (throughput not limited), two USB 2.0 ports, two USB 3.0 ports, and dual-monitor support via a pair of micro HDMI (HDMI Type D) ports for up to 4K resolution . The Pi 4 is also powered via a USB-C port, enabling additional power to be provided to downstream peripherals, when used with an appropriate PSU. The Pin-out details of Raspberry Pi 4 is shown below.

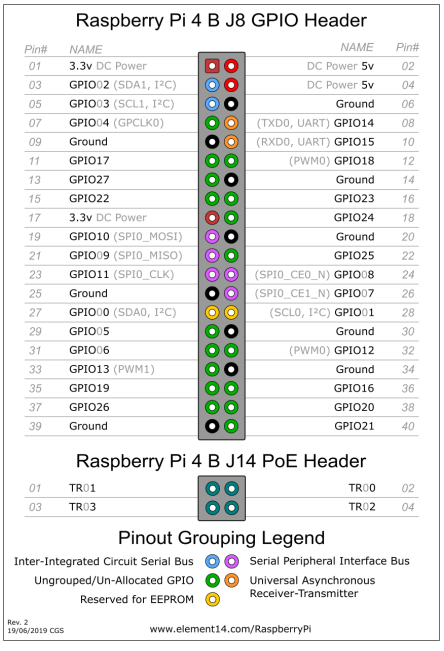


Figure 5.3: Pin-out Details

5.3 TI MSP430

The MSP430 is a mixed-signal micro controller family from Texas Instruments. Built around a 16-bit CPU, the MSP430 is designed for low cost and, specifically, low power consumption embedded applications. The device features a powerful 16-bit RISC CPU, 16-bit registers, and constant generators that contribute to maximum code efficiency. The digitally controlled oscillator (DCO) allows wake-up from low-power modes to active mode in less than 1 s.

Figure 5.4: MSP 430G2553

MSP 430G2553 is the micro-controller that is used over here. The MSP430G2553 series are ultra-low-power mixed signal micro-controllers with built-in 16-bit timers, up to 24 I/O capacitive-touch enabled pins, a versatile analog comparator, and built-in communication capability using the universal serial communication interface. In addition, the MSP430G2553 family members have a 10-bit analog-to-digital ADC.

The basic features of MSP430G2553 are given below:

- Low supply-voltage range: 1.8 V to 3.6 V
- Ultra-low power consumption
- Five power-saving modes
- 16-bit RISC architecture, 62.5 ns instruction cycle time

5.4 ESP 8266

The ESP8266 is a low-cost Wi-Fi microchip, with a full TCP/IP stack and micro-controller capability. This small module allows micro-controllers to connect to a Wi-Fi network and make simple TCP/IP connections.

It is a 32 bit RISC microprocessor running at 80MHz. The ESP8266 consists of memory of 32KiB instruction RAM and 80KiB user data RAM.It also consists of 16 GPIO pins.

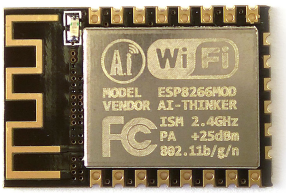


Figure 5.5: ESP8266 Wi-Fi Module

The Pin-out details of ESP8266 are given below:

- VCC, Voltage (+3.3 V; can handle up to 3.6 V)
- GND, Ground (0 V)
- RX, Receive data bit X
- TX, Transmit data bit X
- EN, Enable (Always HIGH using 3.3k)
- RST, Reset (Always HIGH using 3.3k)
- GPIO 0, General-purpose input/output No. 0 (Always HIGH using 3.3k)
- GPIO 15, General-purpose input/output No. 15 (Always LOW using 3.3k)
- ADC, 10 bit data; Voltage Range 0 - 1V

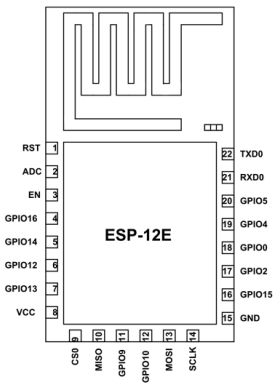


Figure 5.6: ESP8266 Pin-out Details

5.5 LM 386

The LM386 is an integrated circuit containing a low voltage audio power amplifier. This is mainly used to amplify the incoming sound signals required for transmission .The IC consists of an 8 pin dual in-line package (DIP-8) and can output 0.25 to 1 watts of power depending on the model using a 9-volt power supply.

Figure 5.7: LM 386 Op Amp

The IC LM386 audio amplifier consists of 8-pins where each pin of this IC is discussed below:

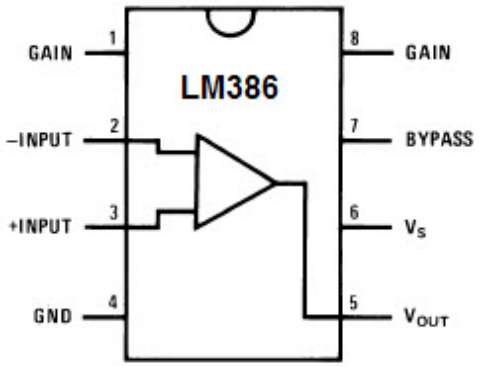


Figure 5.8: LM 386 Pin Diagram

- Pin-1 is gain pin, used adjust the amplifier gain by connecting this IC to an external component capacitor.
- Pin-2 is the non-inverting pin, is used to provide the audio signal.
- Pin-3 is the inverting terminal and it is normally connected to ground.
- Pin-4 is a ground pin connected to the ground terminal of the system.
- Pin-5 is the output pin, used to provide amplified output audio.

- Pin-6 is connected to the power.
- Pin-7 bypass pin is used to connect a decoupling capacitor.
- Pin-8 is the gain setting pin.

5.6 LM 317

The LM 317 is a positive linear voltage regulator. It is mainly used to provide a constant voltage of 3.3V at the micro-controller.

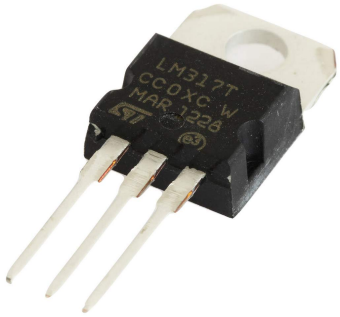


Figure 5.9: LM 317 Linear Voltage Regulator

The pin diagram of LM 317 is shown below:

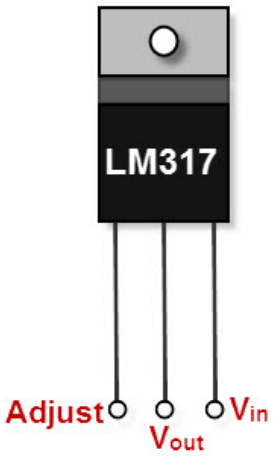


Figure 5.10: Pin Configuration of LM 317

Chapter - 6

IMPLEMENTATION

6.1 Training and Evaluation of Artificial Neural Network

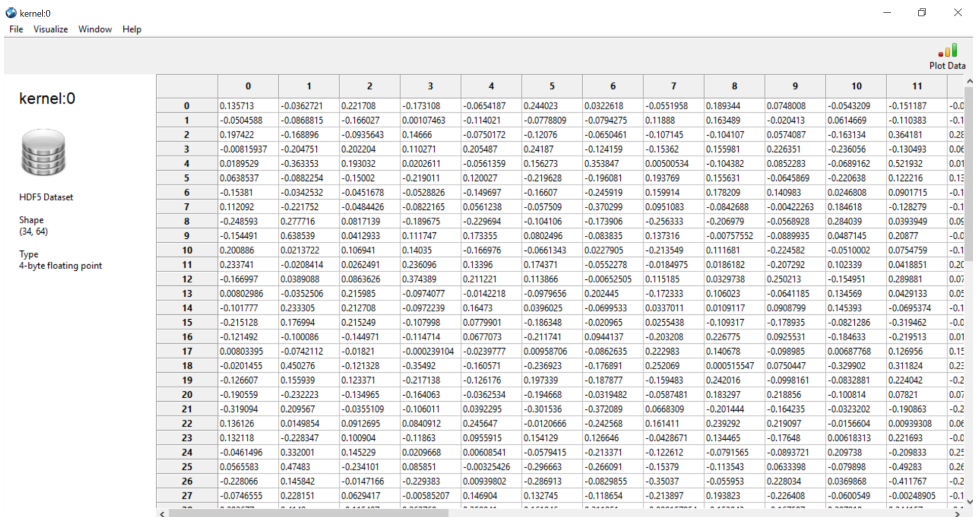
Training and Testing of Artificial Neural Networks were done in Python version 3.6.8 using the following API's:

- Librosa: Librosa is the package used for audio analysis. It is basically used in music generation(using LSTM's), Automatic Speech Recognition *etc.*
- Matplotlib: It is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter.
- PyAudio: PyAudio provides Python bindings for PortAudio, the cross-platform audio I/O library. PyAudio can be used to play and record audio signals on a variety of platforms.
- TensorFlow: TensorFlow is a Python library for fast numerical computing. It is a foundation library that can be used to create Deep Learning models directly or by using wrapper libraries that simplify the process built on top of TensorFlow.
- Keras: Keras is an open-source neural-network library written in Python. It is designed to enable fast experimentation with deep neural networks.

Two separate programs for training and evaluation as well as real time detection were implemented using Python. The siren sound signals as well as the background noise were recorded and stored in two different folders namely Emergency and Non-Emergency respectively. These stored datasets are then used for training the neural network. Training is done based on a technique that sets weights of neurons by comparing the trained data with the tested data. During the training phase the recorded signals are passed through a bandpass filter and preprocessed. Each recorded files are taken as small chunks of 4410 samples (0.1 sec of data) and 34 short term features are extracted using PyAudio analysis and given as input to the Artificial Neural Network. This is followed by a set of 3 hidden layers (256 nodes each) which result in a single output. A Keras sequential Neural Network is used.A Keras Sequential model is appropriate for a plain stack of layers which can be used with multiple inputs and multiple outputs.A Dense layer is just a regular layer of neurons in a neural network. Each neuron receives input from all the neurons in

the previous layer, thus densely connected. Keras supports the early stopping of training via a callback called EarlyStopping. This callback allows you to specify the performance measure to monitor, the trigger, and once triggered, it will stop the training process. The EarlyStopping callback is configured when instantiated via arguments. Epochs are an arbitrary cutoff, generally defined as "one pass over the entire dataset" used to separate training into distinct phases, which is useful for logging and periodic evaluation. When using evaluation_data or evaluation_split with the fit method of Keras models, evaluation will be run at the end of every epoch.Keras handles the problem in a modular way, and several different back-end engines can be plugged seamlessly into Keras.At this time, Keras has two back-end implementations available: the TensorFlow back-end and the Theano back-end. TensorFlow is an open-source symbolic tensor manipulation framework developed by Google, Inc.
The rectified linear activation function (called ReLU) has been shown to lead to very high-performance networks. This function takes a single number as an input, returning 0 if the input is negative, and the input if the input is positive. A sigmoid function is a type of activation function, and more specifically defined as a squashing function. Squashing functions limit the output to a range between 0 and 1, making these functions useful in the prediction of probabilities. Same process is applied to the sound signals that are recorded for evaluation.

These extracted features are then passed through a neural network and certain weights were assigned. These weights obtained for training and evaluation are then compared and adjusted whenever there is a change in the trained signals and reference signals. A Confusion matrix can be made to evaluate the performance of the Neural Network (Figure 8.1). This is obtained from the sklearn Python library. These trained values are converted to HDF5 binary data format and stored in a .h5 file.



	0	1	2	3	4	5	6	7	8	9	10	11
0	0.135713	-0.0362721	0.221708	-0.173108	-0.0654187	0.244023	0.0322618	-0.0551958	0.189344	0.0748008	-0.0543209	-0.151187
1	-0.0504588	-0.0868815	-0.166027	0.00107463	-0.114021	-0.0778809	-0.0794275	0.11888	0.163489	-0.020413	0.0614669	-0.110383
2	0.197422	-0.168896	-0.0935643	0.14666	-0.0750172	-0.12076	-0.0650461	-0.107145	-0.104107	0.0574087	-0.163134	0.364181
3	-0.00815937	-0.204751	0.202204	0.110271	0.205487	0.24187	-0.124159	-0.15362	0.155981	0.226351	-0.236056	-0.130493
4	0.0189529	-0.363353	0.193032	0.0202611	-0.0561359	0.156273	0.353847	0.00500534	-0.104382	0.0852283	-0.0689162	0.521932
5	0.0638537	-0.0882254	-0.15002	-0.219011	0.120027	-0.219628	-0.196081	0.193769	0.155631	-0.0645869	-0.220638	0.122216
6	-0.15381	-0.0342532	-0.0451678	-0.0528826	-0.149697	-0.16607	-0.245919	0.159914	0.178209	0.140983	0.0246808	0.0901715
7	0.112092	-0.221752	-0.0484426	-0.0822165	0.0561238	-0.057509	-0.370299	0.0951083	-0.0842688	-0.00422263	0.184618	-0.128279
8	-0.248593	0.277716	0.0817139	-0.189675	-0.229694	-0.104106	-0.173906	-0.256333	-0.206979	-0.0568928	0.284039	0.0393949
9	-0.154491	0.638539	0.0412933	0.111747	0.173355	0.0802496	-0.083835	0.137316	-0.00757552	-0.0889935	0.0487145	0.20877
10	0.200886	0.0213722	0.106941	0.14035	-0.166976	-0.0661343	0.0227905	-0.213549	0.111681	-0.224582	-0.0510002	0.0754759
11	0.233741	-0.0208414	0.0262491	0.236096	0.13396	0.174371	-0.0552278	-0.0184975	0.0186182	-0.207292	0.102339	0.0418851
12	-0.166997	0.0389088	0.0863626	0.374389	0.211221	0.113866	-0.00652505	0.115185	0.0329738	0.250213	-0.154951	0.289881
13	0.00802986	-0.0352506	0.215985	-0.0974077	-0.0142218	-0.0979656	0.202445	-0.172333	0.106023	-0.0641185	0.134569	0.0429133
14	-0.101777	0.233305	0.212708	-0.0972239	0.16473	0.0396025	-0.0699533	0.0337011	0.0109117	0.0908799	0.145393	-0.0695374
15	-0.215128	0.176994	0.215249	-0.107998	0.0779901	-0.186348	-0.020965	0.0255438	-0.109317	-0.178935	-0.0821286	-0.319462
16	-0.121492	-0.100086	-0.144971	-0.114714	0.0677073	-0.211741	0.0944137	-0.203208	0.226775	0.0925531	-0.184633	-0.219513
17	0.00803395	-0.0742112	-0.01821	-0.000239104	-0.0239777	0.00958706	-0.0862635	0.222983	0.140678	-0.098985	0.00687768	0.126956
18	-0.0201455	0.450276	-0.121328	-0.35492	-0.160571	-0.236923	-0.176891	0.252069	0.000515547	0.0750447	-0.329902	0.311824
19	-0.126607	0.155939	0.123371	-0.217138	-0.126176	0.197339	-0.187877	-0.159483	0.242016	-0.0998161	-0.0832881	0.224042
20	-0.190559	-0.232223	-0.134965	-0.164063	-0.0362534	-0.194668	-0.0319482	-0.0587481	0.183297	0.218856	-0.100814	0.07821
21	-0.319094	0.209567	-0.0355109	-0.106011	0.0392295	-0.301536	-0.372089	0.0668309	-0.201444	-0.164235	-0.0323202	-0.190863
22	0.136126	0.0149854	0.0912695	0.0840912	0.245647	-0.0120666	-0.242568	0.161411	0.239292	0.219097	-0.0156604	0.00939308
23	0.132118	-0.228347	0.100904	-0.11863	0.0955915	0.154129	0.126646	-0.0428671	0.134465	-0.17648	0.00618313	0.221693
24	-0.0461496	0.332001	0.145229	0.0209668	0.00608541	-0.0579415	-0.213371	-0.122612	-0.0791565	-0.0893721	0.209738	-0.209833
25	0.0565583	0.47483	-0.234101	0.085851	-0.00325426	-0.296663	-0.266091	-0.15379	-0.113543	0.0633398	-0.079898	-0.49283
26	-0.228066	0.145842	-0.0147166	-0.229383	0.00939802	-0.286913	-0.0829855	-0.35037	-0.055953	0.228034	0.0369868	-0.411767
27	-0.0746555	0.228151	0.0629417	-0.00585207	0.146904	0.132745	-0.118654	-0.213897	0.193823	-0.226408	-0.0600549	-0.00248905

Figure 6.1: Portion Of the Generated h5 file

6.2 Audio Capturing and Wireless Transmission

A unidirectional microphone is used to capture the sound signals at a distance of around 40 to 60 meters away from the traffic junction. The range can be extended using Long Range Wi-Fi Modules. These signals are amplified using an amplifier circuit. An Op-Amp LM 386 is used. It is interfaced as shown below along with the Wi-Fi module:

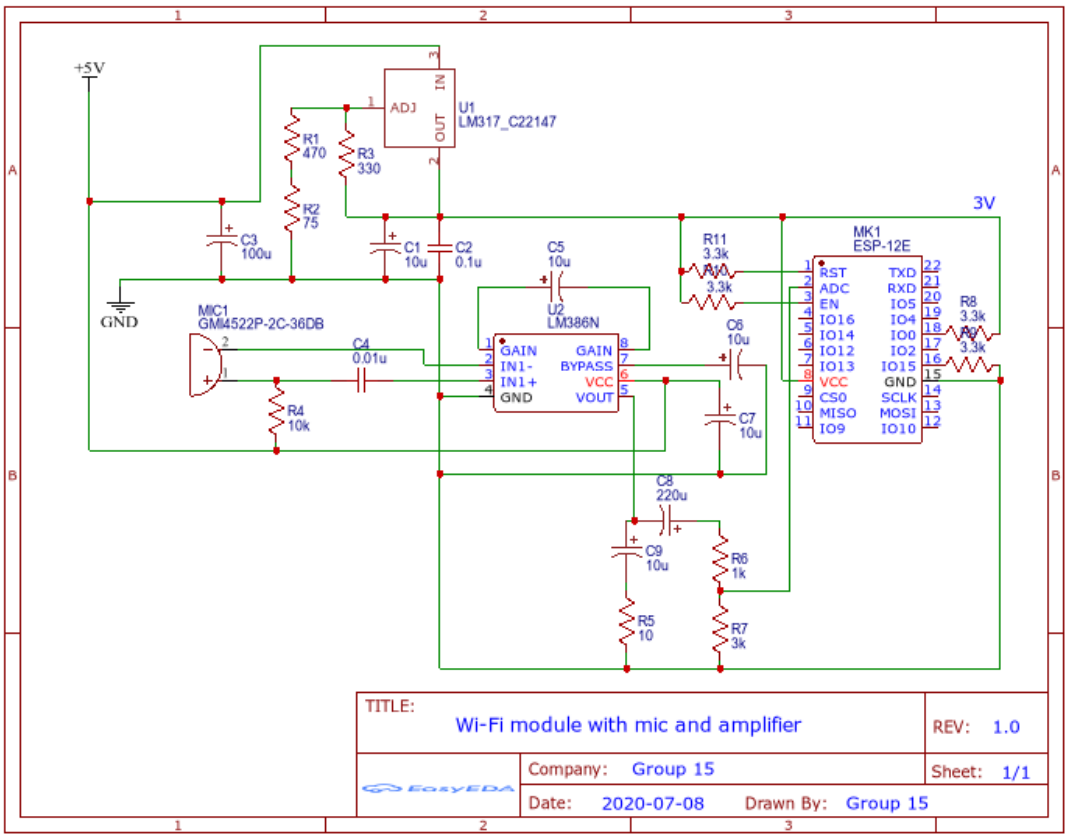


Figure 6.2: Schematic of interfaced microphone and amplifier circuit

The amplified signal may range from 0 to 5V peak to peak and a voltage divider circuit is used to convert this to 0 to 1V peak to peak for the Wi-Fi module. This signal is given to the in-built ADC of ESP 8266 which can be represented in a 10-bit binary format. We were able to achieve a maximum sampling rate of 9000 with the Wi-Fi module. A external 16 bit ADC with sampling rate of 44.1kHz can be used to improve the prediction rate. The 10 bit sampled data is mapped to 16 bit data and is converted to a string and stored. For continuous data transmission, it is converted to text/plain format and sent wirelessly. The reading of data from the ADC begins when a request is received from the Raspberry-Pi.

6.3 Data Reception on Raspberry-Pi

The Wi-Fi module acts as an access point and the Raspberry-Pi is always connected to it. Python uses "urllib.request" to receive data from the IP address of the Wi-Fi module. The received is stored in a **numpy** array and used for Re-sampling using "scipy" library.

We used a sampling rate of 5000 samples per second for continuous data reception. Re-sampling is done at 44.1kHz.
Feature extraction from the re-sampled data is done using "Pyaudio analysis" and is compared with the trained model HDF file and predict the probability. A threshold can be set to improve the detection of the system. If the probability value is above the set threshold, a successful detection can be made.

Real Time Detection of the Ambulance Siren can be performed and the following result is obtained:

```
Non Emergency vehicles-0.12972277402877808
Non Emergency vehicles-[0.11899991]
Non Emergency vehicles-0.2129722237586975
Non Emergency vehicles-[0.25668728]
Ambulance is comming. Set the Green signal quick- 0.5538880228996277
Non Emergency vehicles-[0.4268408]
Ambulance is comming. Set the Green signal quick- 0.5857581496238708
Ambulance is comming. Set the Green signal quick- [0.61391425]
Ambulance is comming. Set the Green signal quick- 0.6575718522071838
Ambulance is comming. Set the Green signal quick- [0.67775506]
Non Emergency vehicles-0.25032487511634827
Non Emergency vehicles-[0.24032766]
Non Emergency vehicles-0.17970412969589233
Non Emergency vehicles-[0.15270136]
Non Emergency vehicles-0.17972829937934875
Non Emergency vehicles-[0.17462234]
Non Emergency vehicles-0.2088310867547989
Non Emergency vehicles-[0.15966268]
Non Emergency vehicles-0.12292230874300003
Non Emergency vehicles-[0.1274829]
Non Emergency vehicles-0.1987697035074234
Non Emergency vehicles-[0.20545556]
Non Emergency vehicles-0.13049758970737457
Non Emergency vehicles-[0.11808577]
```

Figure 6.3: Real Time Detection of Ambulance Siren

When 3 consecutive detections are made, we can confirm the presence of an emergency vehicle and send an interrupt signal to the traffic junction micro-controller which will then give a caution signal to the current green path and sets the corresponding side of the emergency vehicle to green. Another microphone can be placed at the junction to confirm the emergency vehicle has passed after which the traffic control resumes its loop.

Chapter - 7

CHALLENGES FACED

7.1 Detection Of Siren from Noisy Environments

Differentiating the siren sound signals from background noise was one of the major challenges of this project as the frequency spectrum of the siren sound is composed of multiple peaks in its frequency spectrum as shown in the figure shown below. This was resolved by employing the method of Artificial Neural Networks.

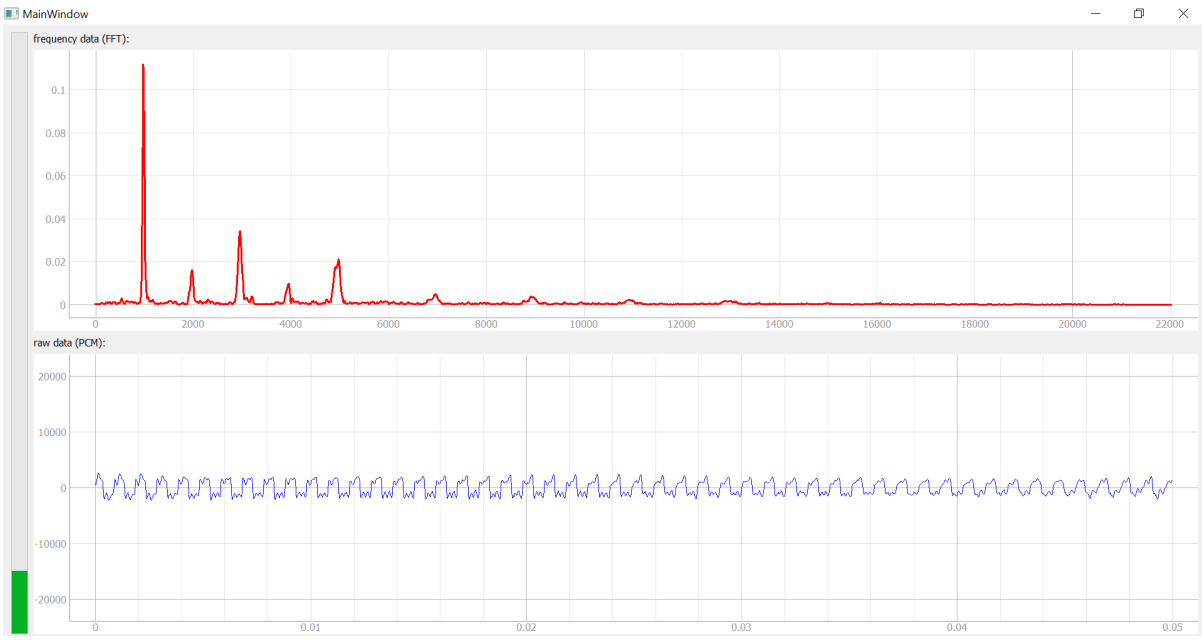


Figure 7.1: FFT Of Ambulance Siren

7.2 Wireless Transmission of the detected signals

Wireless transmission of the detected signals from Raspberry Pi through ESP 8266 was another major challenge encountered in the project as the siren sound signal was not fully transmitted during the initial phase. Upon searching the web we got to know the correlation between the sampling rate of wi-fi module and the python program. This was resolved by reducing the sampling rate from 7000Hz to 5000Hz of the python program.

7.3 Cost Factor

Cost was another factor in deciding the components for our system. Our system requires highly unidirectional microphone and a very good amplifier in order to amplify the incoming sound signals.

Chapter - 8

RESULTS AND OBSERVATIONS

8.1 Software Simulation

Sound of the ambulance siren and background noises were all recorded and stored as **.wav** files in two separate folders namely **Emergency** and **Non-Emergency** respectively. These recorded sound signals are then used for training and evaluation. During training and evaluation a confusion matrix is created and these trained values are stored in an **HDF file**.

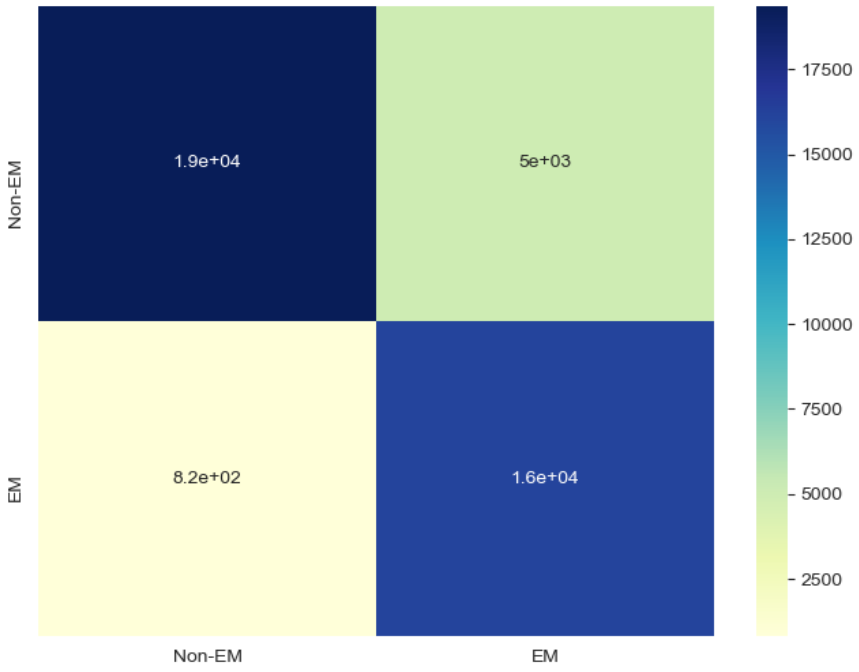


Figure 8.1: Confusion matrix of the trained signals. X-axis represents the predicted values and Y-axis represents the true values

A confusion matrix is a table that is often used to describe the performance of a

classification model (or "classifier") on a set of test data for which the true values are known. It allows the visualization of the performance of an algorithm. It allows easy identification of confusion between classes e.g. one class is commonly mislabeled as the other.
Using the .h5 file generated during the training phase real time detection is carried out.The results obtained are shown below:

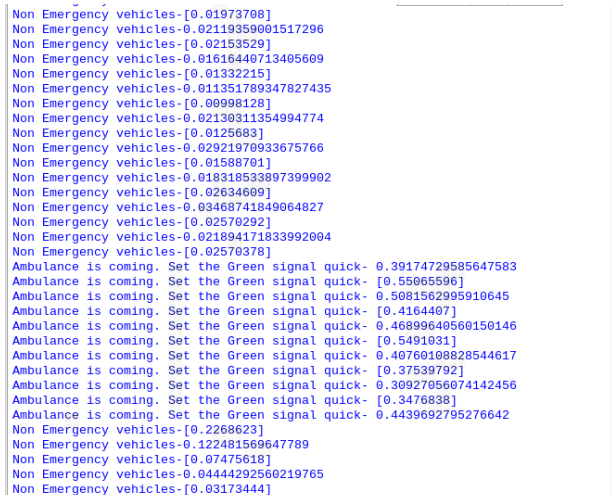


Figure 8.2: Real Time Detection

8.2 Interfacing Hardware Components

The whole experimental setup is as shown below:

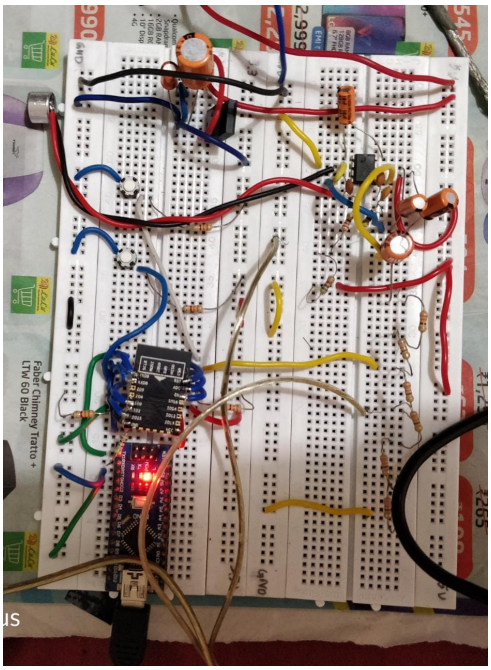


Figure 8.3: Experimental Setup

8.3 Output

The initial state of the traffic junction is shown below:

Figure 8.4: Traffic junction Before Detecting Siren

The state of the traffic junction after detecting the siren is given below:

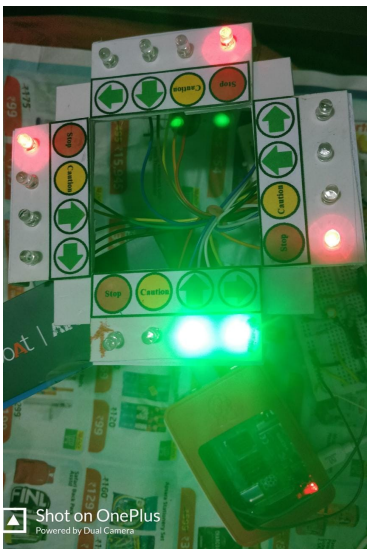


Figure 8.5: Traffic junction After Detecting Siren

Chapter - 9

CONCLUSION AND FUTURE SCOPE

9.1 Conclusion

The designed system was successful in detecting the Siren of the emergency vehicle and thereby scheduling the movement of emergency vehicles through the junction. Three different Ambulance siren sound signals of different frequency range were successfully trained and detected using the method of Artificial Neural Networks. The ambulance siren sound signals were detected with an accuracy of around 84% by the Siren Detection System.

9.2 Future Scope

The present traffic management system is designed based on the assumption that only one emergency vehicle crosses the traffic junction at a single point of time. This can be further improved by developing python codes for real time scenarios like multiple emergency vehicles crossing the traffic junction at the same point of time, traffic junctions near hospitals *etc.*
Another improvement which can be done is the implementation of a 4 way traffic junction where the inter-junction communication is facilitated by IoT based communication network.

REFERENCES

[1] Dean Carmel, Ariel Yeshurun, Yair Moshe Signal and Andrew and Erna Viterbi; "Detection of Alarm Sounds in Noisy Environments"; Image Processing Laboratory (SIPL) Faculty of Electrical Engineering, Technion – Israel Institute of Technology; IEEE-2017 - 25th European Signal Processing Conference (EUSIPCO)

[2] Van-Thuan Tran, Yu-Cheng Yan, Wei-ho Tsai ; "Detection Of Ambulance And Fire Truck Siren Sounds Using Neural Networks"; Department of Electronic Engineering, National Taipei University of Technology, Taiwan; Proceedings of 51st Research World International Conference, Hanoi, Vietnam, 26th -27th July 2018.

[3] Takuya Miyazakia , Yuhki Kitazonoa , Manabu Shimakawab; "Ambulance Siren Detector using FFT on dsPIC"; Proceedings of the 1st IEEE/IIAE International Conference on Intelligent Systems and Image Processing 2013.

[4] Jiun-Jian Liaw, Wen-Shen Wang , Hung-Chi Chu , "Recognition of the Ambulance Siren Sound in Taiwan by the Longest Common Subsequence".2013 IEEE International Conference on Systems, Man, and Cybernetics

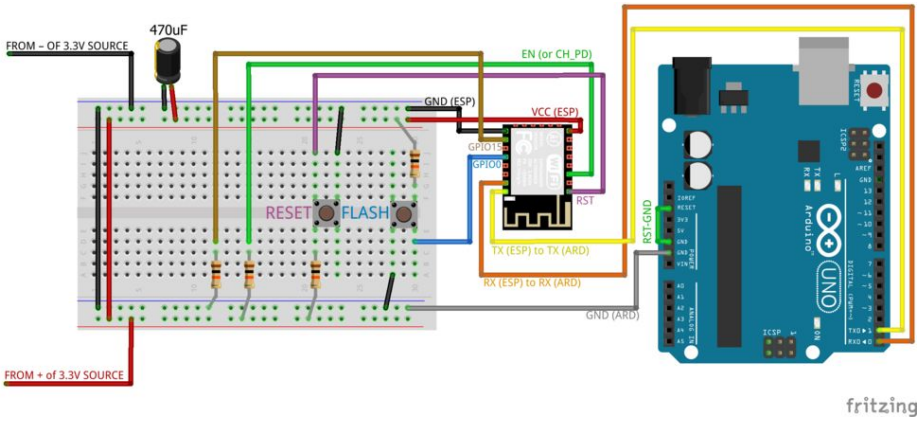
[5] Rohini Priya P., Anju Joy J., Sumathy G; " Traffic light pre-emption control system for Emergency vehicles."; SSRG Int. J. Electron. Commun. Eng. 2015; 2:21–25.

[6] Sandhya, K. S., Karthikeyan, B. (2017). "Automatic traffic diversion system using traffic signals." 2017 International Conference on Nextgen Electronic Technologies: Silicon to Software (ICNETS2).

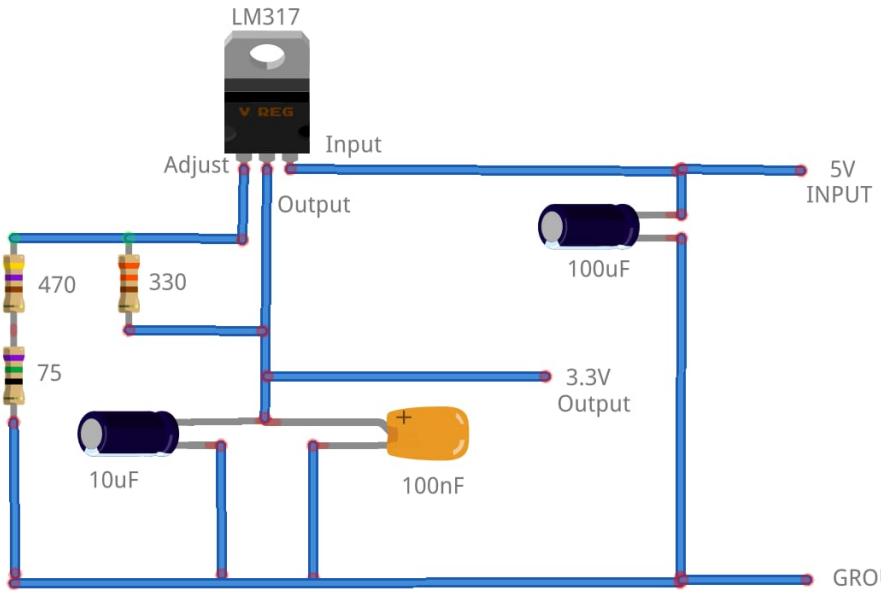
[7] Kapileswar Nellore and Gerhard P. Hancke;"Traffic Management for Emergency Vehicle Priority Based on Visual Sensing" ; Published: 10 November 2016 MDPI

Appendix A
Circuit Diagrams

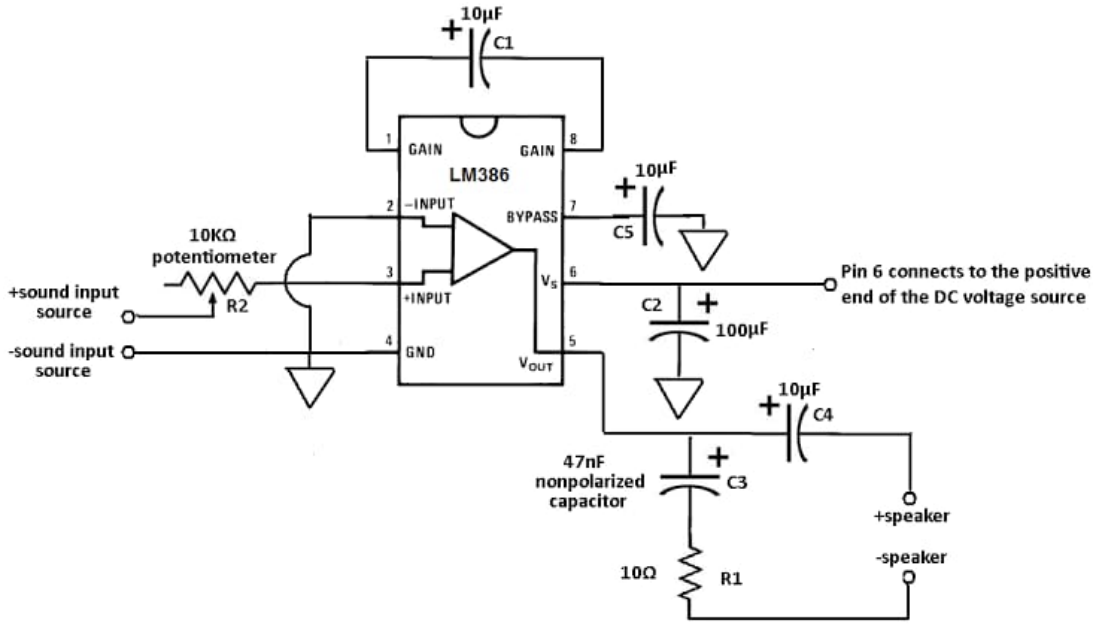
Interfacing Wi-Fi Module



Interfacing LM 317



Interfacing LM 386



Appendix B

IEEE Paper

Smart Traffic Management System

An Emergency Vehicle Detection System for Traffic Junctons

Jeffin George Johnson*, A N Mohammed Shinas†, Naveen M R‡, and Navin Shaju§
Department of Electronics and Communication Engineering
Rajagiri School of Engineering and Technology, Ernakulam, India
Email: *jeffingjohnson@gmail.com †shin.shinas98@gmail.com ‡nnnaveenmr1@gmail.com §navisk98@gmail.com

***Abstract*—Vehicular traffic is endlessly increasing everywhere in the world and can cause terrible traffic congestion at intersections. Most of the traffic lights today feature a fixed green light sequence,therefore the green light sequence is determined without taking the presence of the emergency vehicles into account. Therefore, emergency vehicles such as ambulances, police cars, fire engines,etc. get stuck in a traffic jam and are delayed in reaching their destination. This can lead to the loss of property and valuable lives. In general, emergency service vehicle such as ambulances, fire trucks, and police cars use sirens to warn other road users for quick passage, especially for moving through the traffic. However, due to the well soundproofing techniques in modern cars, drivers may not be aware of the approach of emergency vehicles, especially when in-vehicle audio systems are used.As a consequence, emergency vehicles may be blocked and even collided with other vehicles. The proposed system can help prioritize the movement of these emergency vehicles.**

I. Introduction

The traffic light control plays a vital role in any intelligent traffic management system. The green light sequence and green light duration are the two key aspects to be considered in traffic light control. In many countries, most traffic lights feature fixed sequences and light length duration. Fixed control methods are however only suitable for stable and regular traffic, but not for dynamic traffic situations. Looking at the present state of practice, the green light sequence is determined without taking the possible presence of emergency vehicles into account. Therefore, emergency vehicles such as ambulances, police cars, fire engines, etc. must wait in traffic at an intersection which delays their arrival at their destination causing loss of lives and property.

The objective of this project is to design and implement a unique and cost effective traffic management system to control and monitor the traffic at junctions based on real time data.The system should be able to prioritize the movement of emergency vehicles across the junction by detecting the sirens of the emergency vehicles.

Multiple methods exist to detect emergency vehicles from a traffic environment such as optical and camera-based techniques. However these methods utilize a camera and other image processing techniques that may be expensive and less accurate. Acoustic methods use the sirens of emergency vehicles to detect them from a noisy environment. Using Artificial Neural Networks for detection was found to be a very efficient method for detecting sirens from background noises as compared to other techniques that use FFT or pitch detection algorithms. Supervised learning of the Neural Network is done using a Back-propagation algorithm in Python. This is then implemented on Raspberry Pi using two separate programs to train and evaluate the neural network and to execute real time detection.

The hardware of the system consists of a sound sensor (Unidirectional Microphone) which receives all incoming sound signals, a Raspberry Pi 4B which acts as the control system module, a Wi-Fi module for short range wireless communication, a micro controller (TI MSP 430G2553) for controlling the traffic lights and a set of LED's acting as the traffic lights. The sound sensor accepts all the incoming sound signals that are wirelessely transmitted through a Wi-Fi module to the control system module. The Control system determines whether the incoming signal is a siren or not and sends a control signal to a micro controller for controlling the traffic lights.

II. System Overview

The complete system can be realised in terms of its hardware and software components.The block diagram can be realised as shown below:

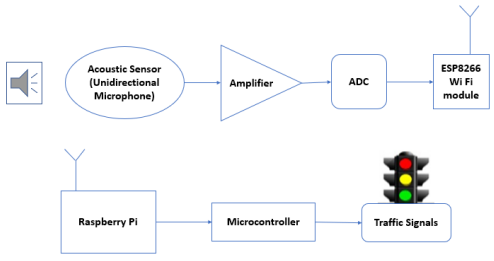


Fig. 1. Schematic of Smart Traffic Management System

The hardware components of the system are:

- Microphone Condenser Type (Unidirectional)
- Raspberry Pi 4 Model B
- Micro controller - TI MSP 430G2553
- Wi-Fi module - ESP-8266

The software side of this project mainly deals with the detection of siren of emergency vehicles and wireless transmission of audio. This is done using Python and Arduino IDE. The detection method involves Machine learning algorithms. This includes the use of Artificial Neural Networks which are

a series of Back-propagated algorithms that use Supervised learning techniques. This method has very high accuracy and very low time delays around 200ms. Moreover this method is not affected by Doppler effect.

III. Siren Detection Using Artificial Neural Networks

In this section we focus on the detection of siren sounds from ambulance in which the operation modes are basically different from those of other vehicles.The spectrum of an ambulance siren is as shown below:

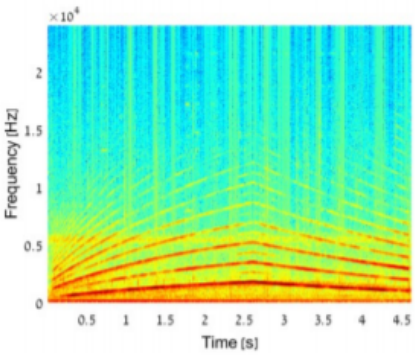


Fig. 2. Frequency spectrum of an Ambulance Siren

Feature Extraction From Siren

Selection of features is a key issue when designing an effective signal detector.Each time frame, requires features that comprise the essence of information relevant to classification, are robust to variations in the signal and to noise, and are efficient in terms of computational and space complexity. To find such features, a large set of features that were previously used for different tasks of audio processing are used. These are divided into 3 categories:

- Time domain features
- Frequency domain features
- Wavelet-based features

A. Time Domain Features

Time domain features include: Pitch, Short Time Energy, Zero crossing rate (ZCR), mean, median, standard deviation (std), etc.

B. Frequency Domain Features

Frequency Domain Features include MFCC or Mel-Frequency Cepstral Coefficients and Spectral flux. MFCC are used to describe the spectral shape of the signal.These coefficients are extracted by applying the discrete cosine transform (DCT) to the log-energy outputs of the nonlinear mel-scale filter-bank

C. Wavelet-based features

The wavelet coefficients capture time and frequency localized information about the audio waveform. The wavelet transform can be viewed as a multi-level process of filtering the signal using a low-pass (scaling) filter and a high-pass (wavelet) filter. They are Discrete Wavelet transform (DWT) and Wavelet packet transform (WPT).

In this method we have used frequency domain features such as MFCCs for siren detection. MFCCs are computed from its spectral representation by setting the upper and lower cutoff frequencies to 600Hz and 1625Hz respectively, since the siren sounds of ambulance have frequencies range from 650 Hz to 1550 Hz.

Siren Sound Detection

The Siren Sound identification (SSI) system was created by using neural networks including Multi-layer perceptron (MLP) or Long Short Term Memory Recurrent Neural Network (LSTM-RNN), which are trained using the back propagation (BP) algorithm and back propagation through time (BPTT) algorithm respectively. In each of the network, the MFCCs vector is used as input of the first layer which connects to a hidden layer in front of an output layer with 3 nodes. MLP is a simple form of feed forward neural network and is easy to implement.

The implementation of a typical neural network for SSI includes three steps:

- Training
- Validation
- Testing.

1) Training: In the training stage, the extracted MFCC features of training dataset are provided as input to estimate the weight parameters in a neural network.

2) Validation: In the validation phase, we use the validation dataset to evaluate the performance of trained model, and through this basis to tune the parameters to achieve the best one.

3) Testing: In the testing phase we pass the unknown audio signal to the fully-trained network to obtain corresponding output, the output with maximum response value is selected as identified result.

Structure of Siren Signal Identification Network is as shown below:

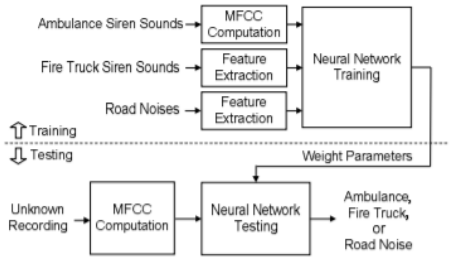


Fig. 3. Siren Signal Identification System

Implementation In Python

The above method is implemented in Python version 3 using APIs such as Tensorflow, Keras etc.Two separate Python programs were implemented:

- To train and Evaluate the Neural Network
- To execute Real Time Detection

Two separate sets of data were also used for training and evaluation titled Emergency and Non Emergency sound signals.The dataset is pre recorded for training. 70% of the dataset used for training 30% for evaluation of sound signals.

IV. Hardware Implementation

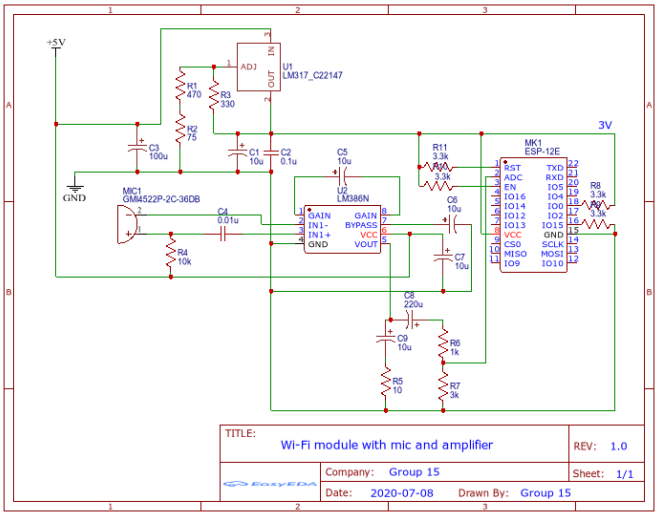


Fig. 4. Interfacing Microphone and Wi-Fi Module

The audio from the microphone is given to an amplifier circuit that uses LM 386 to amplify the received signals. This is converted from 16 bits to 10 bit for the ADC of the Wi-Fi Module (ESP-8266). The audio is sent wirelessly to the Raspberry-Pi where the detection program will be running. The sampling rate used is 5000 samples per second which can be increased up to 8000 samples per second by compromising the time delay.

V. Results and Observations

Software Simulation

Sounds of the ambulance siren and background noises were all recorded and stored as **.wav** files in two separate folders namely **Emergency** and **Non-Emergency** respectively.These recorded sound signals are then used for training and evaluation. During training and evaluation a confusion matrix is created (as shown in Figure 5) and these trained values are stored in **.h5 file** (HDF5 format).

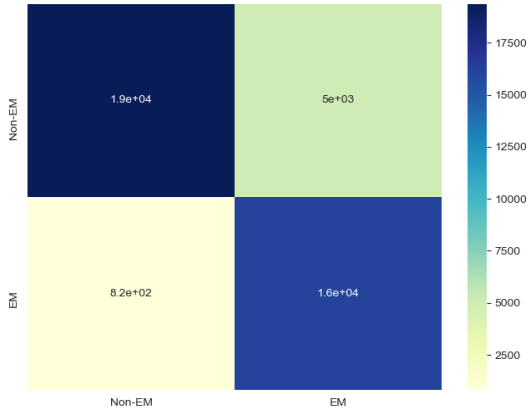


Fig. 5. Confusion matrix of the trained signals. X-axis represents the predicted values and Y-axis represents the true values

A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. It allows the visualization of the performance of an algorithm.It allows easy identification of confusion between classes e.g. one class is commonly mislabeled as the other. The X-axis represents the predicted values and Y-axis represents the true values.
Using the h5 file generated during the training phase real time detection is carried out.The results obtained are shown below:

```
Non Emergency vehicles-[0.01973708]
Non Emergency vehicles-0.02119359001517296
Non Emergency vehicles-[0.02153529]
Non Emergency vehicles-0.016164407134056 09
Non Emergency vehicles-[0.013322151]
Non Emergency vehicles-0.011351789347827435
Non Emergency vehicles-[0.00998128]
Non Emergency vehicles-0.02130311354994774
Non Emergency vehicles-[0.0125683]
Non Emergency vehicles-0.029219709336757 66
Non Emergency vehicles-[0.01588701]
Non Emergency vehicles-0.018318533897399902
Non Emergency vehicles-[0.02634609]
Non Emergency vehicles-0.03468741849064827
Non Emergency vehicles-[0.02570292]
Non Emergency vehicles-0.021894171833992004
Non Emergency vehicles-[0.02570378]
Ambulance is coming. Set the Green signal quick- 0.39174729585647583
Ambulance is coming. Set the Green signal quick- [0.55065596]
Ambulance is coming. Set the Green signal quick- 0.5081562995910645
Ambulance is coming. Set the Green signal quick- [0.4164407]
Ambulance is coming. Set the Green signal quick- 0.4689964056015014 6
Ambulance is coming. Set the Green signal quick- [0.5491031]
Ambulance is coming. Set the Green signal quick- 0.4076010882854461 7
Ambulance is coming. Set the Green signal quick- [0.37539792]
Ambulance is coming. Set the Green signal quick- 0.3092705607414245 6
Ambulance is coming. Set the Green signal quick- [0.3476838]
Ambulance is coming. Set the Green signal quick- 0.4439692795276642
Non Emergency vehicles-[0.2268623]
Non Emergency vehicles-0.122481569647789
Non Emergency vehicles-[0.07475618]
Non Emergency vehicles-0.04444292560219765
Non Emergency vehicles-[0.03173444]
```

Fig. 6. Real Time Detection - Software Simulation

Output

The components were interfaced as shown and the output after detecting the siren is shown (Figure 7 and Figure 8). The Wi-Fi module acts as an access point and the Raspberry-Pi is connected to it for wireless transmission.

After detection, a control signal is given as an interrupt from the Raspberry-Pi to the micro-controller to set the corresponding side of the junction to green to allow the emergency vehicle to pass through.

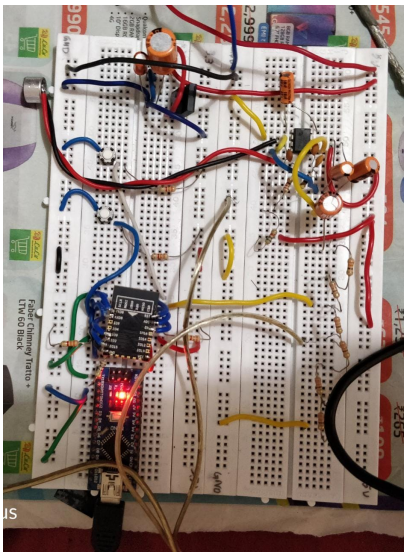


Fig. 7. Experimental Setup

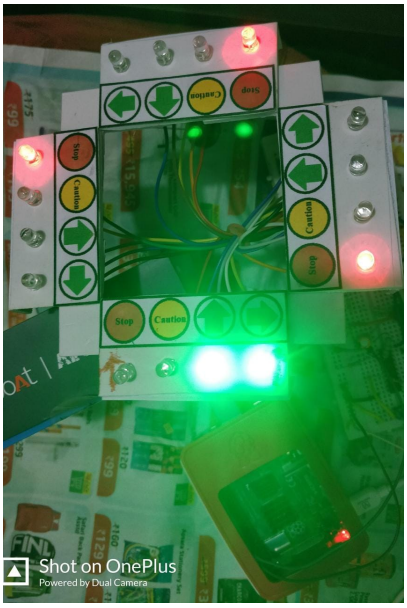


Fig. 8. Traffic junction After Detecting Siren

After the vehicle passes through, the junction returns to its original loop.

VI. Conclusion and Future Scope

The designed system was successful in detecting the Siren of the emergency vehicles and thereby scheduling the movement of emergency vehicles through the junction.Three different siren sound signals of different frequency ranges were successfully trained and detected using Artificial Neural Networks.The siren sound signals were detected with an accuracy of around 84% by the Siren Detection System.
The present traffic management system is designed based on the assumption that only one emergency vehicle crosses the traffic junction at a single point of time.This can be further improved by developing python codes for real time scenarios like multiple emergency vehicles crossing the traffic junction at the same point of time. Another improvement which can be done is the implementation of a 4 way traffic junction where the inter-junction communication is facilitated by IoT based communication network.

Acknowledgment

The authors acknowledge the invaluable guidance of Ms. Shyama Sreekumar, Associate Professor, Department of Electronics and Communication Engineering, Rajagiri School of Engineering and Technology, that assisted us from the conception of the project till it's complete execution and for being for being their go-to person for all their programming mishaps. The authors also thank the lab faculties and all the technical staffs of the Department of Electronics and Communication Engineering, Rajagiri School of Engineering and Technology for their support and cooperation. The authors extend a special thanks to Prof. Dr. Rithu James, Head of Department, Department of Electronics and Communication Engineering, Rajagiri School of Engineering and Technology, for permitting them to utilise all the necessary facilities of the institution.

References

[1] Dean Carmel, Ariel Yeshurun, Yair Moshe Signal and Andrew and Erna Viterbi; "Detection of Alarm Sounds in Noisy Environments"; Image Processing Laboratory (SIPL) Faculty of Electrical Engineering, Technion – Israel Institute of Technology; IEEE-2017 - 25th European Signal Processing Conference (EUSIPCO)

[2] Van-Thuan Tran, Yu-Cheng Yan, Wei-ho Tsai ; "Detection Of Ambulance And Fire Truck Siren Sounds Using Neural Networks"; Department of Electronic Engineering, National Taipei University of Technology, Taiwan; Proceedings of 51st Research World International Conference, Hanoi, Vietnam, 26th -27th July 2018.

[3] Takuya Miyazakia , Yuhki Kitazonoa , Manabu Shimakawab; "Ambulance Siren Detector using FFT on dsPIC"; Proceedings of the 1st IEEE/IIAE International Conference on Intelligent Systems and Image Processing 2013.

[4] Jiun-Jian Liaw, Wen-Shen Wang , Hung-Chi Chu , "Recognition of the Ambulance Siren Sound in Taiwan by the Longest Common Subsequence". 2013 IEEE International Conference on Systems, Man, and Cybernetics

[5] Rohini Priya P., Anju Joy J., Sumathy G; " Traffic light pre-emption control system for Emergency vehicles."; SSRG Int. J. Electron. Commun. Eng. 2015; 2:21–25.

[6] Sandhya, K. S., Karthikeyan, B. (2017). ”Automatic traffic diversion system using traffic signals.” 2017 International Conference on Nextgen Electronic Technologies: Silicon to Software (ICNETS2).

Appendix C

Presentation Slides

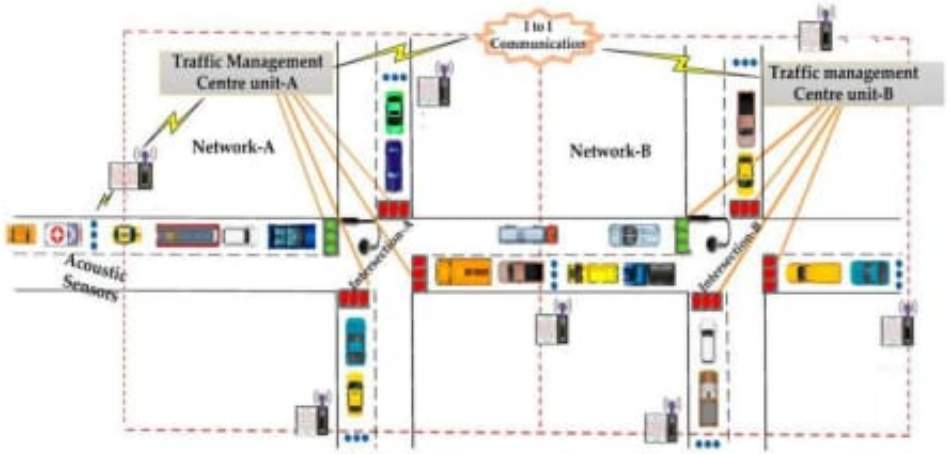
1

Objectives:

- To detect sirens of emergency vehicles from noisy environments.
- Design and implementation of a Python based Machine Learning Program for the detection.
- Model a Raspberry- Pi to microcontroller interface to control traffic junctions.

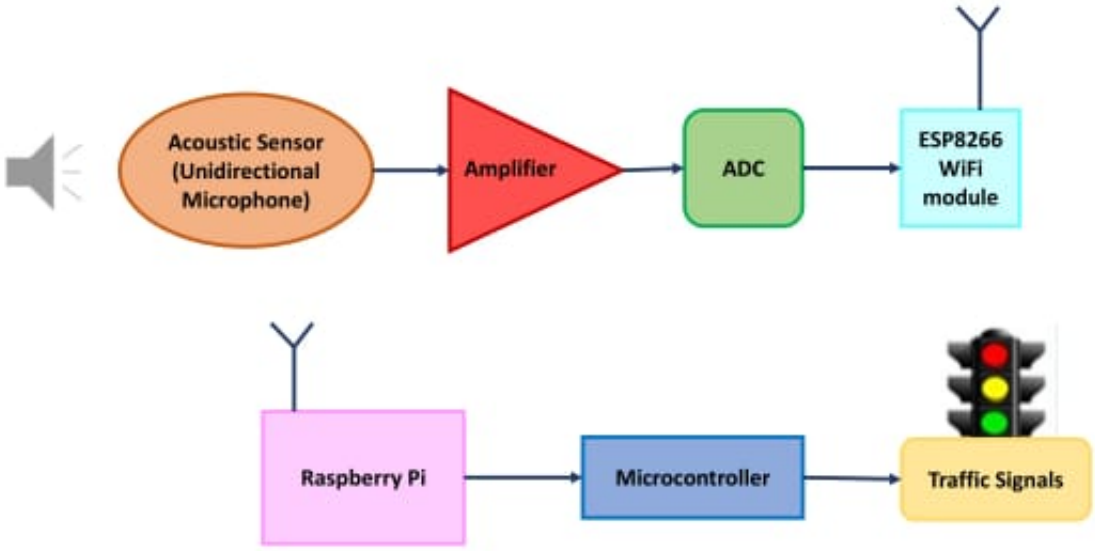
2

Project Overview:



3

Block Diagram:



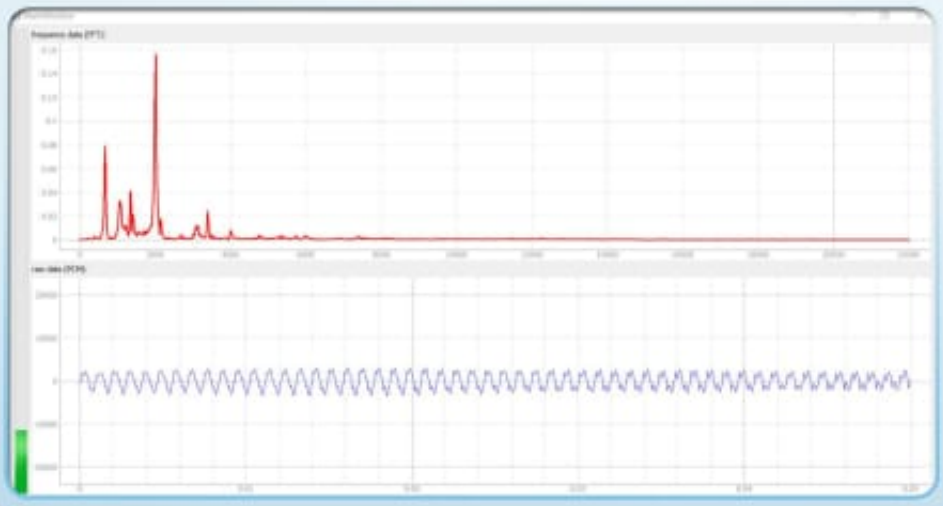
4

Methodology:

- Siren Detection using Artificial Neural Networks

5

FFT of Ambulance Siren



6

Hardware Overview:

- Raspberry Pi Model 4B
- Wi-Fi Module – (**ESP8266**)
- Microphone (Condenser Type)
- Op-Amp (**LM386**)
- Microcontroller (**TI MSP430G2553**)

7

Software Overview:

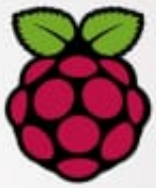
- **Python 3.7 :** Ambulance detection using ANN.
- **Arduino IDE :** To program Wi-Fi module.
- **Code Composer Studio**: To program traffic signal microcontroller.

8

Work Flow:

1. SETTING UP OF RASPBERRY-PI
2. TRAINING OF ARTIFICIAL NEURAL NETWORK USING THE REQUIRED RECORDED SOUND FILES.
3. SETTING UP A MICROPHONE AND AMPLIFIER CIRCUIT WITH WI-FI.
4. INTERCONNECTION OF WI-FI MODULE TO RASPBERRY PI
5. SETTING UP OF 4-WAY TRAFFIC LIGHT JUNCTION MODEL.
6. REAL-TIME DETECTION

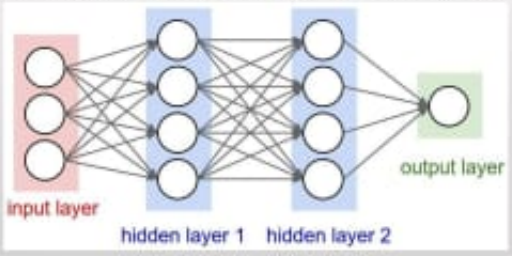
Implementation in Python using Raspberry-Pi



- The audio files are pre recorded at 16 bit 44.1kHz sampling rate, for training of neural network.
- 2 sets of audios used for training:
  - Emergency Sounds
  - Non-Emergency Sounds
- 2 separate python programs were implemented:
  - To Train and Evaluate the Neural Network
  - To Execute Real time detection
- After training, the data can be used for Real-Time detection through the Wi-Fi module

Artificial Neural Networks:

- **Artificial Neural Network** - uses series of algorithms , recognizes relationships in a set of data, mimics the way the human brain operates.
- We used **Keras Sequential** neural network
- **34** input, hidden layers(256,256,256) and single output.
- Supervised learning method with Back-Propagation(BP) Algorithm.



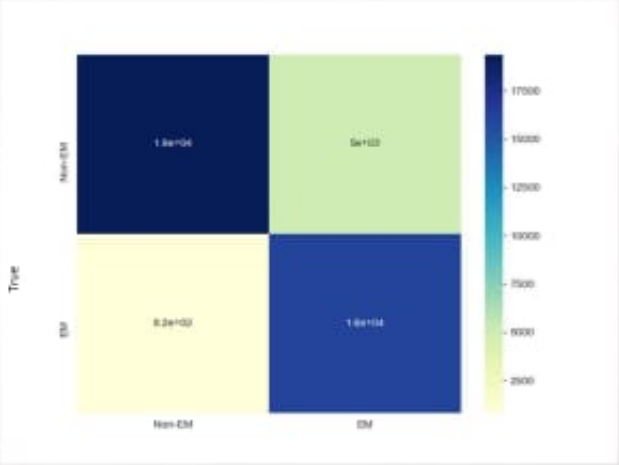
Training

- We used around 150 pre-recorded audio files for ambulance sirens.
- 450 files of Non-Emergency sounds/noises.
- Extraction of audio features are done using ***pyaudioanalysis***
- We take small chunks of these audio files and extract 34 features like
  - Time domain features
  - Frequency domain features
  - Wavelet-based features
- This 34 features are given as input to the Neural Network for training.
- The resulting data is stored as a Hierarchical Data Format (HDF) file.
- Using this file we can evaluate any presence of Emergency vehicles.

Confusion Matrix

- A confusion matrix is a plot used to describe the performance of a classification model on a set of test data for which the true values are known.
- It allows the visualization of the performance of an algorithm.
- This is the Confusion Matrix obtained from the Evaluation Stage
- Accuracy while evaluation: 88%

  Precision: 1.0

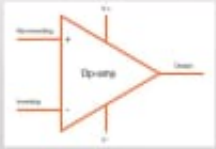


Confusion matrix of the trained signals. X-axis represents the predicted values and Y-axis represents the true values

Real-Time detection

Real time detection is done step by step as follows:

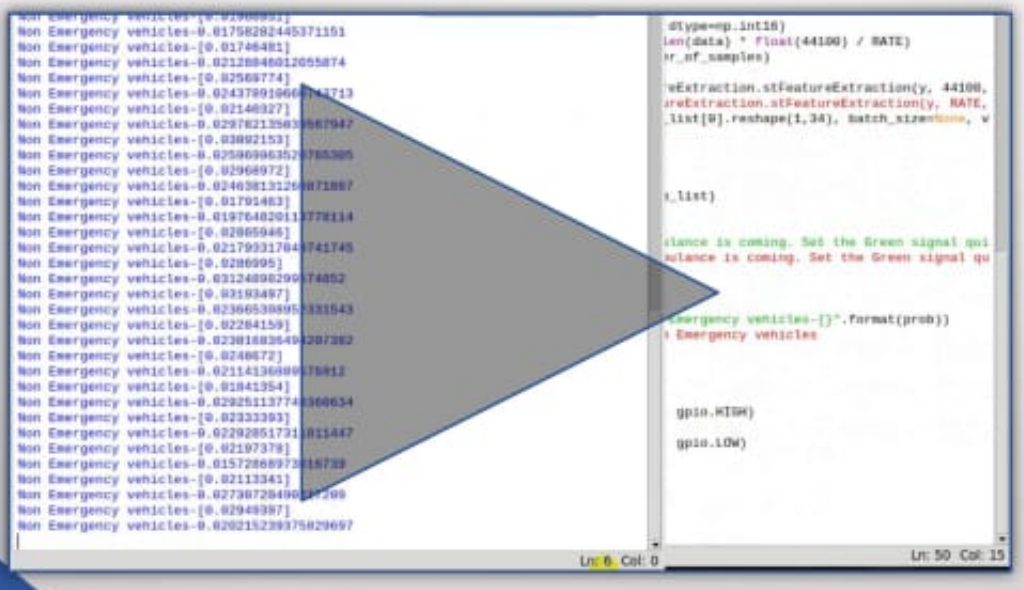
- The audio is captured by a unidirectional microphone around 100m away from junction.
- It is amplified using an amplifier circuit(Op-Amp LM386).
- The signal is given to the Wi-Fi module (ESP8266) where the inbuilt ADC converts the data
- 5000 sampling rate was obtained, sufficient for detection.
- Raspberry-Pi is connected to the Wi-Fi module.



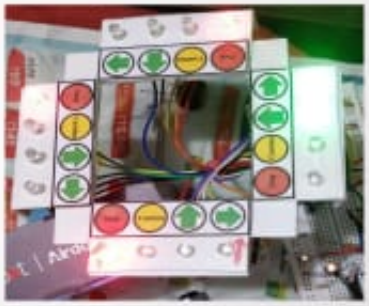
Real time detection

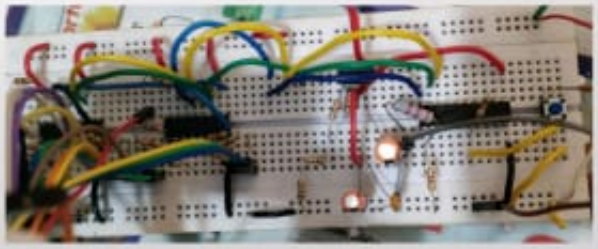
- When the Raspberry-Pi gives a URL Request, the Wi-Fi module sends one second duration of audio to the Raspberry-Pi.
- Raspberry-Pi receives the audio data and Converts it into a ***numpy*** array, which is then resampled to the sample rate of trained data.
- This data is then compared with the HDF file.
- The predicted value is used to confirm the presence of an EM vehicle.
- We can set the threshold value for comparison to this predicted value.
- If the predicted value is above threshold it is an EM vehicle else detected as noise.

Software Simulation:



- The presence of an emergency vehicle is confirmed after three consecutive EM detections take place.
- The Raspberry Pi sends a HIGH signal to the microcontroller controlling the Traffic Junction.
- The model Traffic Signal Junction is controlled by a **MSP430G2553** microcontroller.
- 2 shift registers are used for controlling 16 LEDs.



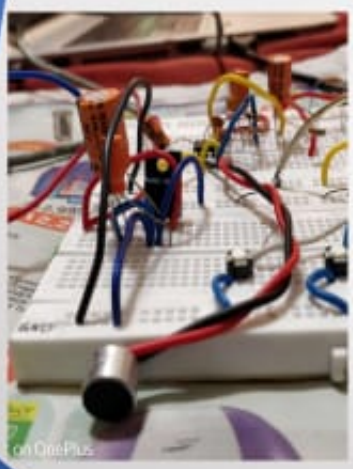


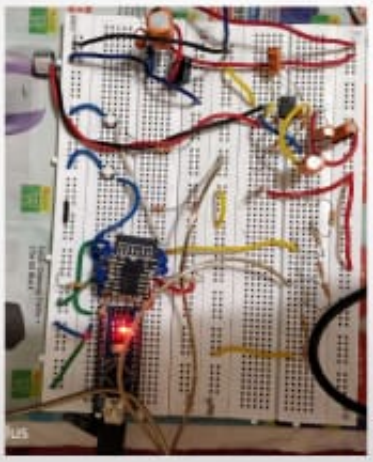
17

- When a HIGH signal is received from the Raspberry Pi, the traffic signal sets a green signal to the corresponding side while all other paths are turned to red.
- To indicate the arrival of an Emergency vehicle, an LED display can be shown to indicate it to the other drivers.
- The loop resumes when ambulance has passed.

18

Result & Observations:





19

Video Demonstration:

20

Advantages:

- High Accuracy
- Robustness to noise & low amplitude signals
- Large Flexibility in training data; Can be used for various sirens;
- Not affected by Doppler Effect.
- Very low time delay (500 ms)

Future Scope:

- Implementation on a 4-Way Traffic junction and an IOT-based Inter-Junction Communication Network
- Using ADC with higher sampling frequencies(fs=44.1khz) will increase precise prediction.
- Using a unidirectional mic will help from false detection.
- Using better programming methods we can reduce the delay of data transmission.
- Using long range wireless communication for longer distance junctions will ensure smooth flow of emergency vehicles.

Work Division:

A N Mohammed Shinas	Traffic Light System Setup and Control
Naveen M R	Training & Testing of Artificial Neural Network
Navin Shaju	Microphone & Wi-Fi Module Setup and Control
Jeffin George Johnson	Raspberry Pi 4 Computer Setup and Control

References:

- Dean Carmel, Ariel Yeshurun, Yair Moshe Signal and Andrew and Erna Viterbi; ***"Detection of Alarm Sounds in Noisy Environments";*** Image Processing Laboratory (SIPL) Faculty of Electrical Engineering, Technion – Israel Institute of Technology; **IEEE-2017** - 25th European Signal Processing Conference (EUSIPCO);
- Van-Thuan Tran, Yu-Cheng Yan, Wei-ho Tsai ; ***"Detection Of Ambulance And Fire Truck Siren Sounds Using Neural Networks";*** Department of Electronic Engineering, National Taipei University of Technology, Taiwan; Proceedings of 51st Research World International Conference, Hanoi, Vietnam, **26th -27th July 2018;**
- Takuya Miyazakia , Yuhki Kitazonoa , Manabu Shimakawab; ***"Ambulance Siren Detector using FFT on dsPIC";*** Proceedings of the 1st **IEEE/IIAE** International Conference on Intelligent Systems and Image Processing **2013;**

- Kadar Muhammad Masum, A., Kalim Amzad Chy, M., Rahman, I., Nazim Uddin, M., & Islam Azam, K. (2018). ***An Internet of Things (IoT) based Smart Traffic Management System: A Context of Bangladesh***. **2018** International Conference on Innovations in Science, Engineering and Technology (ICISET).
- Rohini Priya P., Anju Joy J., Sumathy G; ***"Traffic light pre-emption control system for Emergency vehicles."***; SSRG Int. J. Electron. Commun. Eng. **2015**; 2:21–25.
- Sandhya, K. S., & Karthikeyan, B. (2017). ***"Automatic traffic diversion system using traffic signals."*** **2017** International Conference on Nextgen Electronic Technologies: Silicon to Software (ICNETS2).

The End

Appendix D

Questionnaire

Q. **What is Back Propagation algorithm used in Neural Networks?**

Ans. Back propagation algorithm is an algorithm is used to effectively train a neural network through a method called chain rule. In simple terms, after each forward pass through a network, backpropagation performs a backward pass while adjusting the model's parameters (weights and biases).

Q. **What is a confusion matrix?**

Ans. A confusion matrix is a table that is often used to describe the performance of a classification model on a set of test data for which the true values are known.It allows the visualisation of the performance of an algorithm.It allows easy identification of confusion between classes.

Q. **What are some of the Time domain features used for feature extraction?**

Ans.Time domain features include :Pitch, Short Time Energy ,Zero Crossing Rate, mean, median ,standard deviation *etc.*

Q. **How well does the system filter out road noises?**

Ans. The Artificial Neural Network is trained using two sets of data:Emergency vehicle sirens and Non-Emergency vehicle Sirens.All road noises including road honks,pedestrian noises and weather related noises are used to train ANN and identify as Non-emergency vehicles.When both sounds are mixed together,unique features of siren are extracted to clearly identify a sound and make a decision.

Q. **What are the disadvantages of the method of Siren detection using FFT?**

Ans.The disadvantages are:

- Less accurate
- Significant time delay of 8s when it comes to detection of signals
- Noise cancellation is ineffective when compared with neural networks.

Q. **How long does the system override the usual traffic signal functioning?**

Ans. Depending on the junction a preset time is given for the vehicle to pass through the junction.The interrupt to the traffic signal exists as long as the emergency vehicle is present in the junction.The central module monitors whether the vehicle has passed and allots additional time if required.

Q. **How is the problem involving multiple emergency vehicles approaching the junction resolved?**

Ans. When multiple emergency vehicles approach a junction, preference is given to vehicle that arrives first.The central control unit monitors whether the vehicles have passed through the junction before allowing the traffic signal to resume.

Q. **What are the different stages involved in the implementation of neural network for Siren Detection?**

Ans. The three different stages invloved are:

- Training
- Validation
- Testing

Appendix E
Program Code

TRAINING AND EVALUATION

```
# Import the libraries
# Python's in-built modules
import os
import glob
import time

# External modules
import numpy as np
import pandas as pd
from tqdm import tqdm

# Librosa for audio
import librosa
import librosa.display

# For designing the band-pass filter
from scipy.signal import butter, lfilter, hilbert

# Plotting modules
import matplotlib.pyplot as plt
import matplotlib.style as ms
import matplotlib.ticker as ticker
import seaborn as sns
sns.set_style("whitegrid")
import matplotlib.gridspec as gridspec

# Pyaudionalaysis functions
from pyAudioAnalysis import audioBasicIO
from pyAudioAnalysis import audioFeatureExtraction

# Scikit-learn modules
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.utils import shuffle
from sklearn.externals import joblib
```

```
from sklearn.metrics import confusion_matrix

# Keras and TensorFlow modules
from keras.models import Sequential from keras.layers import Dense from keras.callbacks
import EarlyStopping from keras.models import load_model import tensorflow as tf from
keras.backend import common as K K.set_image_dim_ordering('th')
# Supress Tensorflow error logs
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'

# fix random seed for reproducibility
from numpy.random import seed
seed(1)

import tensorflow
tensorflow.random.set_seed(2)


def butter_bandpass_filter(data, lowcut=500, highcut=1500, fs=8000, order=5):
"'
Bandpass filter to remove noise
Helps the algorithm to focus only on the relevant frequency band

    "'
nyq = 0.5 * fs
low = lowcut / nyq
high = highcut / nyq

    b, a = butter(order, [low, high], btype='band')
y = lfilter(b, a, data)
return y

def preprocess(y):
"'
Extracts the envelope of the audio signal

    "'
y_filt = butter_bandpass_filter(y)
analytic_signal = hilbert(y_filt)
amplitude_envelope = np.abs(analytic_signal)
return amplitude_envelope


def read_files(files):
X = []
for fn in tqdm(files):
y, sr = librosa.load(fn, sr=8000)
y = preprocess(y)
```

```
features = audioFeatureExtraction.stFeatureExtraction(y, sr, 0.10*sr, .05*sr)
X.extend(features)
return X

def get_data(path_em, path_nonem):
# # Make the data file path user defined
# path_em = '../data/balanced/cleaned_emergency/'
# path_nonem = '../data/balanced/nonEmergency/'

em_files = glob.glob(os.path.join(path_em, '*.wav'))
nonem_files = glob.glob(os.path.join(path_nonem, '*.wav'))

print("Reading the Em class files")
X_em = read_files(em_files)

#N_em = len(em_files) # Remove data imbalance
print("Reading the Non-Em class files")
X_nonem = read_files(nonem_files)

return X_em, X_nonem


def prepare_data_train(X_em, X_nonem):
X_em = np.array(X_em)
X_nonem = np.array(X_nonem)

X = np.vstack((X_em, X_nonem))
Y = np.hstack((np.ones(len(X_em)), np.zeros(len(X_nonem))))

scaler = StandardScaler()
scaler.fit_transform(X)

X, Y = shuffle(X, Y, random_state=7)

# Save scaler for testing later
# Use sklearn's inbuilt saving tool
scaler_filename= "scaler.save"
joblib.dump(scaler, scaler_filename)
print("Saved scaler!")

return X, Y, scaler

def prepare_data_test(X_em, X_nonem, scaler):
X_em = np.array(X_em)
X_nonem = np.array(X_nonem)

X = np.vstack((X_em, X_nonem))
```

```
Y = np.hstack((np.ones(len(X_em)), np.zeros(len(X_nonem))))

# # Correct the expression below
# X = (X - scaler.mean_)/scaler.std_
scaler.fit_transform(X)

X, Y = shuffle(X, Y, random_state=7)

return X, Y



    def build_model():
model = Sequential([
Dense(64, input_dim=34, activation='relu'),
Dense(256, activation='relu'),
Dense(256, activation='relu'),
Dense(256, activation='relu'),
Dense(1, activation='sigmoid')
])

model.summary()

model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

return model



    def run_model(model, X_train, Y_train, X_test, Y_test):
earlystop = EarlyStopping(monitor='val_loss', min_delta=0.001, patience=10, verbose=0,
mode='auto')
callbacks_list = [earlystop]
history = model.fit(X_train, Y_train, epochs=200, validation_data=(X_test, Y_test), batch_size=5000,
callbacks=callbacks_list)
model.save("Trained_data.h5")
print("Saved model to disk!")
return history

def plot_model_history(model_history):
fig, axs = plt.subplots(1,2,figsize=(15,5))



# summarize history for accuracy
axs[0].plot(range(1,len(model_history.history['accuracy'])+1),model_history.history['accuracy'])
axs[0].plot(range(1,len(model_history.history['val_accuracy'])+1),model_history.history['val_accuracy']
axs[0].set_title('Model Accuracy')
```

```
axs[0].set_title('Model Accuracy')
axs[0].set_ylabel('Accuracy')
axs[0].set_xlabel('Epoch')
axs[0].set_xticks(np.arange(1,len(model_history.history['accuracy'])+1),len(model_history.history['accu
axs[0].legend(['train', 'val'], loc='best');

# summarize history for loss
axs[1].plot(range(1,len(model_history.history['loss'])+1),model_history.history['loss'])
axs[1].plot(range(1,len(model_history.history['val_loss'])+1),model_history.history['val_loss'])
axs[1].set_title('Model Loss')
axs[1].set_ylabel('Loss')
axs[1].set_xlabel('Epoch')
axs[1].set_xticks(np.arange(1,len(model_history.history['loss'])+1),len(model_history.history['loss'])/10
axs[1].legend(['train', 'val'], loc='best');

plt.savefig('model_history.png')

    def clip_level_prediction(model, X_test, Y_test):
Y_pred = model.predict_classes(X_test)
cm = confusion_matrix(Y_pred, Y_test)
df_cm = pd.DataFrame(cm, index = ['Non-EM', 'EM'],
columns = ['Non-EM', 'EM'])
plt.figure(figsize = (8,6))
sns.heatmap(df_cm, annot=True, cmap='YlGnBu');
plt.savefig('Clip_confusion_matrix.png')

def predict_probability(y, scaler, sr):
y = preprocess(y)
features_list = audioFeatureExtraction.stFeatureExtraction(y, sr, 0.10*sr, .05*sr)
scaler.transform(features_list)
count = 0
N = 10 # Window size
th = 0.5 # Minimum probabilty value for Em presence

model = load_model('Trained_data.h5')

prob_list = [ ]
class_list = [ ]

for i in range(N):
p = model.predict(features_list[i].reshape(1,34), batch_size=None, verbose=0)
p = p.flatten()
prob_list.append(p)
prob = np.mean(prob_list)

if prob > th:
#print("Em")
```

```
class_list.append(1)
else:
#print("Non-em")
class_list.append(0)

for i in range(N,len(features_list)):
prob_list.pop(0)
p = model.predict(features_list[i].reshape(1,34), batch_size=None, verbose=0)
p = p.flatten()
prob_list.append(p)
prob = np.mean(prob_list)
#print(prob)
if prob > th:
#print("Em")
class_list.append(1)
else:
#print("Non-em")
class_list.append(0)

return class_list
# Test Accuracy
def predict_output(y, scaler, sr):
class_list = predict_probability(y, scaler, sr)

if np.mean(class_list) > 0.5:
return 1
else:
return 0


    def main():

    # # TO DO:
# Save extracted train and test data into npz or hdfs format

    # train data
train_path_em = 'For_training/Emergency/'
train_path_nonem = 'For_training/nonEmergency/'

    print("Training data")
Em_data, Nonem_data = get_data(train_path_em, train_path_nonem)

    X_train, Y_train, scaler = prepare_data_train(Em_data, Nonem_data)

    # test data
test_path_em = 'For_evaluation/Emergency/'
test_path_nonem = 'For_evaluation/nonEmergency/'
```

```
print("Test data")
Em_data, Nonem_data = get_data(test_path_em, test_path_nonem)

X_test, Y_test = prepare_data_test(Em_data, Nonem_data, scaler)

# Build the model
model = build_model()

    # Run the training
history = run_model(model, X_train, Y_train, X_test, Y_test)

# Plot the loss and accuracy curves
#plot_model_history(history)

# Get audio clip level evaluation results
clip_level_prediction(model, X_test, Y_test)

    scaler_filename = "scaler.save"
scaler = joblib.load(scaler_filename)

test_em_files = glob.glob(os.path.join(test_path_em, '.wav'))
test_nonem_files = glob.glob(os.path.join(test_path_nonem, '.wav'))

tot_em, correct_em, tot_nonem, correct_nonem = 0, 0, 0, 0

print("Evaluating Em class test data")
for test_file in tqdm(test_em_files):
y, sr = librosa.load(test_file, sr=8000)
classes = predict_output(y, scaler, sr=8000)
if classes == 1:
correct_em += 1
tot_em += 1

print("Evaluating NonEm class test data")
for test_file in tqdm(test_nonem_files):
y, sr = librosa.load(test_file, sr=8000)
classes = predict_output(y, scaler, sr=8000)
if classes == 0:
correct_nonem += 1
tot_nonem += 1

print("Correct Em:
nTotal Em:
nCorrect NonEm:
nTotal NonEm: ".format(correct_em, tot_em, correct_nonem, tot_nonem))
```

```
print("=== EVALUTION METRICS ===")
print("Accuracy: ".format(float(correct_em + correct_nonem)/(tot_em + tot_nonem)))
print("Precision: ".format(float(correct_em)/(tot_nonem - correct_nonem + correct_em)))
print("Recall: ".format(float(correct_em)/tot_em))

if __name__ == "__main__":
main()
```

REAL TIME AMBULANCE DETECTION

```
import os
import numpy as np
import pyaudio
from scipy.signal import butter, lfilter, hilbert, resample
from pyAudioAnalysis import audioFeatureExtraction
from keras.models import load_model
import urllib.request
import RPi.GPIO as gpio


gpio.setmode(gpio.BCM)
gpio.setup(18, gpio.OUT)
###### Data Pre-processing functions ###

def butter_bandpass_filter(data, lowcut=500, highcut=1500, fs=44100, order=5):
nyq = 0.5 * fs
low = lowcut / nyq
high = highcut / nyq

b, a = butter(order, [low, high], btype='band')
y = lfilter(b, a, data)
return y

def preprocess(y):
y_filt = butter_bandpass_filter(y)
analytic_signal = hilbert(y_filt)
amplitude_envelope = np.abs(analytic_signal)
return amplitude_envelope

    ##################################################
RATE = 5000
CHUNK=4410
model = load_model('Trained_data.h5')
gpio.setmode(gpio.BCM)
gpio.setup(18, gpio.OUT)

input=True, frames_per_buffer=CHUNK)
```

```
th = 0.5
prob_list = []
count = 0
var = 0

try:
while True:
count = 1
with urllib.request.urlopen('http://192.168.137.204') as f:
audio=f.read().split()
data = np.array(audio, dtype=np.int16)
number_of_samples = round(len(data) * float(44100) / RATE)
data = resample(data, number_of_samples)
y = preprocess(data)
features_list = audioFeatureExtraction.stFeatureExtraction(y, 44100, number_of_samples,
number_of_samples)
#features_list = audioFeatureExtraction.stFeatureExtraction(y, RATE, CHUNK, CHUNK)
p = model.predict(features_list[0].reshape(1,34), batch_size=None, verbose=0)
p = p.flatten()
prob_list.append(p)

if count%1 == 0:
prob = np.mean(prob_list)

if prob >= th:
print("Ambulance is coming. Set the Green signal quick- ".format(prob))
var += 1
else:
print("Non Emergency vehicles-".format(prob))
var = 0

prob_list = [prob]

if var > 2:
gpio.output(18, gpio.HIGH)
else:
gpio.output(18, gpio.LOW)
except KeyboardInterrupt:
stream.stop_stream()
stream.close()
p.terminate()
```

Appendix F

Mapping The Project Objectives with POs and PSOs

Sl. No	PROJECT OBJECTIVES	POs	PSOs
1.	To identify the problems of movement of Emergency vehicles across a traffic junction and to come up with a viable solution.	PO4, PO5	PSO2
2.	To research the world wide web and perform literature survey on various papers regarding the identified solution.	PO2	-
3.	To identify different algorithms for Siren detection and to study them in detail and apply it for the defined problem.	PO1,PO5, PO6, PO9	-
4.	To study machine learning algorithms using tools like Python.	PO4, PO3	-
5.	To identify the hardware and software requirements for the project.	PO7	PSO1
6.	To work together as a team to overcome the challenges faced and to convey our ideas and progress to the evaluation panel.	PO3, PO8, PO9,PO10	-
7.	To use the knowledge acquired during the four years of study to the problem.	PO12	-
8.	To manage the entire expertise and time plan of the project.	PO10	PSO1
9.	To design, build and debug the required system.	PO3, PO6 PO8, PO1	PSO3
10.	To prepare presentation, reports and paper with minimum plagiarism and providing necessary acknowledgements.	PO2, PO9	PSO2