Smart Traffic Control System

A Thesis submitted to
School of Electrical and Computer Engineering
Addis Ababa Institute of Technology
Addis Ababa University

in Partial Fulfillment of the Requirements for the Degree of Bachelor of Science
Electrical and Computer Engineering (Comp)

BY: EYUEL WORKU
MIKIYAS TESFAYE
ADVISER: YEABSIRA ASEFA

Addis Ababa, Ethiopia
September 28, 2021

Declaration

We, the undersigned, declare that the Project comprises our own work in compliance with internationally accepted practices; We have fully acknowledged and referred all materials used in this Project work.

Eyuel Worku

Name	Signature

Mikiyas Tesfaye

Name	Signature

Addis Ababa University
Addis Ababa Institute of Technology
School of Electrical and Computer Engineering

This is to certify that the Thesis prepared by **Eyuel Worku** and **Mikiyas Tesfaye**, entitled *Smart Traffic Control System* and submitted in partial fulfillment of the requirements for the degree of Bachelor of Science in Electrical and Computer Engineering (Comp) complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by:

Adviser Yeabsira Asefa Signature _____ Date _____

_____
Dean, School of Electrical and Computer Engineering

ABSTRACT

The number of vehicles on roads is increasing at an alarming rate and leading to traffic jams and accidents. Today's traffic signalling method allocates fixed time for each lane regardless of the distribution of car density on the different roads of the intersection. This implies that congested roads get same time as roads with little or no cars. To remove this inefficiency, a better system that give more passage time for congested roads than roads with little or no traffic. With this kind of a system, time allocation will be more equitable and congested streets will be given more attention. This will relieve the traffic more effectively than today's fixed time based traffic signalling system.

KEYWORDS

Traffic Jams, Congestion , Vehicle

ACKNOWLEDGMENTS

We would like to express our deepest gratitude for Addis Ababa Institute of Technology especially School of Electrical & Computer Engineering, and all our instructors.

We would also like to express our gratitude for Ms. Yeabsira Asefa, our instructor and mentor.

THANK YOU!

CONTENTS

LIST OF FIGURES

ACRONYMS

3D	Three-Dimensional
AI	Artificial Intelligence
API	Application Program Interface
ARM	Advanced Risc Machines
BSD	Berkeley Software Distribution
CLI	Command Line Interface
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
EEPROM	Electronically Erasable Programmable Read-only Memory
GHZ	Gigahertz
GPIO	General-Purpose Input/output
GPU	Graphics Processing Unit
GSM	Global System for Mobile Communications
GUI	Graphical User Interface
HDMI	High-Definition Multimedia Interface
HTTP	Hypertext Transfer Protocol
I2C	Inter-Integrated Circuit
I/O	Input/output
IDE	Integrated Development Environment
KB	Kilobyte
LED	Light Emitting Diode
MB	Megabyte
Mbps	Megabits per second
MHZ	Megahertz
MISO	Multiple Input Single Output
MMX	Multi-Media Extensions
OpenCL	Open Computing Language
OS	Operating System
PCB	Printed Circuit Board
POE	Power over Ethernet
PWM	Pulse Width Modulation
RAM	Random-Access Memory
RNN	Recurrent Neural Network
RX	Receive
RXD	Received Data
SCLK	Serial Clock Signal
SD	Secure Digital
SIM	Subscriber Identity Module
SIMD	Single Instruction, Multiple Data Stream
SMS	Short Message Service

SOC	System on Chip
SSE	Streaming SIMD Extensions
SSH	Secure Shell
STL	Standard Template Library
TTL	Time to Live
TX	Artificial Intelligence
TXD	Transmit Data
UK	United Kingdom
URL	Uniform Resource Locator
USB	Universal Serial Bus
USD	United States Dollar

1 INTRODUCTION

1.1 BACKGROUND

Population in cities is growing at an alarming rate and with that number of vehicles on roads is also growing. This has led to several problems like traffic jams, accidents, and even illnesses due to the frustration and stress of the drivers. One of the many causes of traffic jams and accidents is poor traffic management. Even though present-day traffic signaling system works well when the traffic load on the different lanes of the junction is evenly distributed, it becomes inefficient when that's not the case. This is because today's traffic signaling system is fixed time based and does not take density of traffic on the different lanes into consideration. This shortcoming of today's methods is highly pronounced when there is a large density in one of the lanes and little to no traffic on the other lanes. When this occurs today's method of traffic signaling still would give equal green light time to all the lanes when it should have given the lane with the higher traffic density more time.

1.2 PROBLEM STATEMENT

In modern cities, ineffective traffic flow is one of the issues faced. Specially, in a third world country like Ethiopia, where the development of infrastructures is minimal, and this issue is double fold. The city is usually crowded, and people waste significant amount of their time on the road. One of the reasons for this, besides the lack of sufficient amount of infrastructure or the miss match between available roads and number of cars, is the poor traffic light control. These traffic light control systems assign fixed amount of time for the streets coming into the crossroad. But these streets have varying number of cars; the numbers vary with time, day and other factors. To tackle this issue, a real time system that can count cars from all directions and decide accordingly can be employed.

1.3 OBJECTIVE

General Objective

The main objective of the project is to build an adaptive traffic control system. The system will be responsive to traffic densities on the different lanes of a junction. The proposed system will overcome the shortcomings of today's fixed-time based traffic signaling method and in doing so the proposed system is intended to replace the today's system.

Specific Objectives

The specific objectives of this project are:

- Implementing a vehicle detection algorithm using computer vision to detect density of vehicles
- Implementing an API that can store vehicle density data
- Implementing a website that allows viewing of data in the API
- Implementing an effective scheduling algorithm
- Implementing a crossroad prototype to test and demonstrate implemented program

1.4 SCOPE AND LIMITATION

Due to time and resource limitation, the system we implemented has a smaller scope as compared to our design. Out of the three algorithms considered in our design only two were partially implemented. The object detection algorithm was implemented with a fair accuracy. The reason for the limitation in the effectiveness of this algorithm is the lack of data that is suited to train a Haar cascade, which requires a single car per picture. This is hard to find because most images taken on the street using surveillance cameras or otherwise contain several vehicles which are usually tightly packed. In addition to our limited access to good pictures for training the classifier, the images containing the vehicles to be detected also need to have the same layout as the training pictures. Finding a good dataset is quite difficult when a time constraint is added.

Since the algorithm that is used to detect the amount of space available after the crossroad has not been implemented, this factor has not been considered in the scheduling algorithm. In addition, since the object detection algorithm has not been specifically trained for emergency vehicles, this factor was also not considered in the scheduling algorithm implementation. All the other factors in the design were considered with some assumptions.

2

LITERATURE REVIEW

2.1 LITERATURE REVIEW

Gaikwad et al [1] proposed a system which worked by measuring the area covered by vehicles. The system run through a couple of main processes acquiring image, transforming it to grayscale and enhancing the image. Green lines were drawn on the road at arbitrary distances which the camera had to detect. If parts of the lines were not detected, then there was a level of traffic on the road and the traffic lights were adjusted to suit the situation.

Ali et al. [2] proposed a system which uses cameras. The system consisted of cameras for image detection, a storage device for the images detected, an object detection algorithm, background object subtraction and shadow removal, blob segmentation and object classification. The system was designed to be integrated with existing video surveillance systems.

For each lane, detected images were analyzed and the blob ratio was used to determine the type of vehicle which had been detected. Based on the number of vehicles detected in all directions, the traffic lights were adjusted to ease congestion. Traffic control systems using cameras required storage space to store images and extra processing power to process images.

Benjamin et al. [3] proposed a system which consisted of two microcontrollers, one as a counter for the sensors' inputs and the second for controlling the traffic lights. Infrared sensors were used at each section of the intersection to determine vehicle density. An RF module was used for communication between the microcontrollers.

The infrared sensor was placed at a distance from the intersection. This enabled vehicles moving towards the intersection to be counted. A suitable distance had to be chosen taking into consideration the volume of traffic that plies that route and the green light on time. Infrared sensors at each section only counted vehicles when that section had a red light.

This was to eliminate an inaccuracy of vehicle count when vehicles are moving out of the intersection during a green light period. All the infrared sensors were connected to the counter. The sensors detected when a vehicle moved past them. The counter recorded and increased the value for that sensor each time a vehicle moved past the sensor.

2.2 THEORETICAL BACKGROUND

2.2.1 *Raspberry Pi*

The Raspberry Pi is a low cost, credit-card sized micro-controller. This micro-controller can connect a computer monitor or TV, standard keyboard and mouse. It's capable of doing everything you would expect a personal computer to do, from browsing the internet and playing high-definition video, to making spreadsheets, word-processing, and playing games. Moreover, it is a capable of performing computing in high level programming languages like Scratch and Python. Raspberry Pi SBCs feature a Broadcom system on chip with an integrated ARM-compatible CPU and on-chip GPU.

Developed in the United Kingdom by the Raspberry Pi Foundation, in association with Broadcom, the Raspberry Pi has evolved over different generation.

[4] The Raspberry Pi Model B was released in February 2012, followed by the simpler and cheaper Model A in 2014. These first-generation boards feature ARM11 processors, are approximately credit-card sized and represent the standard mainline form-factor. Improved A+ and B+ models were released a year later. A "Compute Module" was released in April 2014 for embedded applications.

The Raspberry Pi 2 was released in February 2015 and initially featured a 900 MHz 32-bit quad-core ARM Cortex-A7 processor with 1 GB RAM. Revision 1.2 featured a 900 MHz 64-bit quad-core ARM Cortex-A53 processor. This processor model is also used by Raspberry Pi 3 Model B but has an improved clock speed of 1.2 GHz.

A Raspberry Pi Zero with smaller size and reduced I/O and GPIO capabilities was released in November 2015. On 28 February 2017, the Raspberry Pi Zero W was launched, a version of the Zero with Wi-Fi and Bluetooth capabilities. On 12 January 2018, the Raspberry Pi Zero WH was launched, a version of the Zero W with pre-soldered GPIO headers.

Raspberry Pi 3 Model B was released in February 2016 with a 1.2 GHz 64-bit quad core ARM Cortex-A53 processor, on-board 802.11n Wi-Fi, Bluetooth, and USB boot capabilities. In 2018, the Raspberry Pi 3 Model B+ was launched, on Pi Day. It has a faster 1.4 GHz processor, a three-times faster gigabit Ethernet (throughput limited to ca. 300 Mbps by the internal USB 2.0 connection), and 2.4 / 5 GHz dual-band 802.11ac Wi-Fi (100 Mbps). Other features are POE, USB boot and network boot.

Raspberry Pi 4 Model B was released in June 2019 with a 1.5 GHz 64-bit quad core ARM Cortex A-72 processor, on-board 802.11ac Wi-Fi, Bluetooth 5, full gigabit Ethernet, two USB 2.0 ports, two USB 3.0 ports, and dual-monitor support via a pair of micro–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 initial Raspberry Pi 4 board has a

design flaw where third-party e-marked USB cables, such as those used on Apple MacBooks, incorrectly identify it and refuse to provide power.

Raspberry Pi 400 was released in November 2020. It features a custom board that is derived from the existing Raspberry Pi 4, specifically remodeled with a keyboard attached. The case was derived from that of the Raspberry Pi Keyboard. A robust cooling solution like the one found in Commodore 64 allows the Raspberry Pi 400's Broadcom BCM2711C0 processor to be clocked at 1.8 GHz, which is slightly higher than the Raspberry Pi 4 it's based on. The keyboard-computer features 4 GB of LPDDR4 RAM.

Raspberry Pi Pico was released in January 2021. It was Raspberry Pi's first board based upon a single microcontroller chip; the RP2040, which was designed by Raspberry Pi in the UK. The Pico has 264 KB of RAM and 2 MB of flash memory. It is programmable in MicroPython, CircuitPython and C. It has partnered with Vilros, Adafruit,Pimoroni,Arduino and SparkFun to build Accessories for Raspberry Pi Pico and variety of other boards using RP2040 Silicon Platform. Rather than perform the role of general-purpose computer like the others in the range it is designed for physical computing, similar in concept to an Arduino.[4]

2.2.2 *Computer Vision*

[5] Computer vision is a field of artificial intelligence that enables computers and systems to derive meaningful information from digital images, videos and other visual inputs and take actions or make recommendations based on that information. If AI enables computers to think, computer vision enables them to see, observe and understand.

Computer vision works much the same as human vision, except humans have a head start. Human sight has the advantage of lifetimes of context to train how to tell objects apart, how far away they are, whether they are moving and whether there is something wrong in an image.

Computer vision trains machines to perform these functions, but it has to do it in much less time with cameras, data and algorithms rather than retinas, optic nerves and a visual cortex. Because a system trained to inspect products or watch a production asset can analyze thousands of products or processes a minute, noticing imperceptible defects or issues, it can quickly surpass human capabilities.

Computer vision is used in industries ranging from energy and utilities to manufacturing and automotive – and the market is continuing to grow. It is expected to reach USD 48.6 billion by 2022

How it works

Computer vision needs lots of data. It runs analyses of data over and over until it discerns distinctions and ultimately recognize images. For example, to train a computer to recognize automobile tires, it needs to be fed vast quantities of tire images and tire-related items to learn the differences and recognize a tire, especially one with no defects.

Two essential technologies are used to accomplish this:

- Deep learning [6]A subset of machine learning in artificial intelligence that has networks capable of learning unsupervised from data that is unstructured or unlabeled. It imitates the workings of the human brain in processing data and creating patterns for use in decision making. Also known as deep neural learning or deep neural network.[6]
- Convolutional neural network [7]A Deep Learning algorithm which can take in an input image, assign importance to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a Convolutional neural network is much lower as compared to other classification algorithms.[7]

Machine learning uses algorithmic models that enable a computer to teach itself about the context of visual data. If enough data is fed through the model, the computer will "look" at the data and teach itself to tell one image from another. Algorithms enable the machine to learn by itself, rather than someone programming it to recognize an image.

A CNN helps a machine learning or deep learning model "look" by breaking images down into pixels that are given tags or labels. It uses the labels to perform convolutions (a mathematical operation on two functions to produce a third function) and makes predictions about what it is "seeing". The neural network runs convolutions and checks the accuracy of its predictions in a series of iterations until the predictions start to come true. It then starts recognizing or seeing images in a way like humans.

Much like a human making out an image at a distance, a CNN first discerns hard edges and simple shapes, then fills in information as it runs iterations of its predictions. A CNN is used to understand single images. A recurrent neural network (RNN) is used in a similar way for video applications to help computers understand how pictures in a series of frames are related to one another.

Common Application Areas

The following are some of the classes of application areas where computer vision is used.

- *Image classification* sees an image and can accurately predict that a given image belongs to a certain class. For example, a social media company might

want to use it to automatically identify and take the appropriate action if a user uploads objectionable images.

- *Object detection* can use image classification to identify a certain class of image and then detect and tabulate their appearance in an image or video. Examples include detecting damages on an assembly line or identifying machinery that requires maintenance. This projects vehicle detection algorithm belongs to this class.

- *Object tracking* follows or tracks an object once it is detected. This task is often executed with images captured in sequence or real-time video feeds. Autonomous vehicles, for example, need to not only classify and detect objects such as pedestrians, cars and road infrastructure, they need to track them in motion to avoid collisions and obey traffic laws.

- *Content-based image retrieval* uses computer vision to browse, search and retrieve images from large data stores, based on the content of the images rather than metadata tags associated with them. This task can incorporate automatic image annotation that replaces manual image tagging. These tasks can be used for digital asset management systems and can increase the accuracy of search and retrieval.[6]

OpenCV

[8]OpenCV is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc.

OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 18 million. The library is used extensively in companies, research groups and by governmental bodies.

Along with well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that employ the library, there are many startups such as Applied Minds, VideoSurf, and Zeitera, that make extensive use of OpenCV. OpenCV's deployed uses span the range from stitching streetview images together, detecting

intrusions in surveillance video in Israel, monitoring mine equipment in China, helping robots navigate and pick up objects at Willow Garage, detection of swimming pool drowning accidents in Europe, running interactive art in Spain and New York, checking runways for debris in Turkey, inspecting labels on products in factories around the world on to rapid face detection in Japan.

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDA and OpenCL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers.[8]

OpenCV uses an algorithm called Haar cascade classifier to perform detection of different objects. A classifier, in machine learning, is the algorithm used by the machine to categorize data into one or more of set of classes. For example, an email classifier is used to scan emails to filter the spams from the useful emails. Haar cascade classifier is then a classifier that is used to detect object from an image or video. A short abstraction of how this algorithm works is presented in the section to follow.

Haar cascade classifier

A Haar cascade classifier, or a Haar classifier, is a machine learning object detection program that identifies objects in an image or video. This is algorithm is described in detail in a paper prepared by Paul Viola and Michael Jones, titled "Rapid Object Detection using a Boosted Cascade of Simple Features". This article goes into the details of Haar Cascade classifier using mathematics to elaborate the technical details of how it works. Here we have presented a highlight instead.

The algorithm can be explained in three basic stages. We explain each one in detail. The following are the three stages:

- Calculating Haar Features
- Creating Integral Images
- Using Adaboost

i *Calculating Haar Features*
Owing their name to their intuitive similarity with the mathematical Haar wavelets, Haar features were used in the first real-time face detector. Essentially, a Haar feature is a series of calculations that are performed on adjacent rectangular regions at a specific location in a detection window. The calculation involves summing of the intensities of the pixels in each region and looking for the difference between the sums.

There are several types of Haar features. Three of these are the basic ones and are listed below:

a) Edge Features

b) Line Features

c) Four-rectangle Features

The figure 2.1 below shows these three basics Haar Features in the order they are presented above.

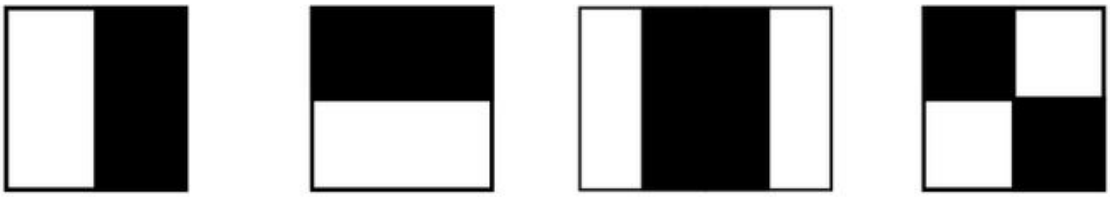

Figure 2.1: Basic Haar Feature

The first two are edge features. As their name indicates, they are used to detect edges in an image. The third one is a line feature used to detect a vertical line. The last one is a four-rectangle feature most likely used to detect a slanted line.

In each of these features, the white bars represent pixels that contain parts of an image that are closer to the light source and would therefore be whiter on a grayscale image (A grayscale image is an image where all the pixels carry only light intensity information). The black bars are the opposite, they are pixels whose image features are farther away from the light source or are obstructed by another object. An example of this could be eyebrows casting slight shadows over the eyes below. Similarly, these features would appear blacker on a grayscale image. The most important reason behind transforming the image to grayscale is this comparison between white and black pixels.

Note that Haar like features are scalable but they cannot be a one-by-one pixel. This is because each cell of the frame represents a single pixel, which can only contain the value for a single color. Since the entire function of these frames is to compare the "white-ness" and "black-ness" of neighboring pixels it cannot be scaled down to a single pixel. The smallest scale for the basic features is:

- Edge Features: 1x2 or 2x1
- Line Features: 1x3 or 3x1
- Four-Rectangle Features: 2x2

To explain these basic features. Consider a face detection algorithm.
a) [9] Edge features: Think of the forehead and the eyebrows or eyes. The forehead is an exposed, flat surface of the face. This allows it to reflect more

light, so it is often "lighter". Eyebrows and the eye area are generally darker. The algorithm would read the lighter shade of the forehead and the transition to the darker eyebrows as an "edge" boundary. Which means it would place the inverted version of the second feature from the image above.

b) Line features: In a line feature, the pattern can go white-black-white just like the third feature in the image above or black-white-black. Think about a nose. The top edge of your nose that stretches from the bridge to the nose tip, while not as flat as the forehead, is still reflective and the closest point on the face to a light source that might be in front of the subject, so it will naturally be brighter and stand out. The area around the nostrils typically bends away from the light making them darker. This pattern would be picked up as a line feature, opposite of the one given in the picture above. Another interesting way that line features are being utilized is in eye-tracking technology. A darker iris sandwiched between the white space of your eye on either side of it.

c) Four-Rectangle Features: This is good for finding diagonal lines and highlights in an image. This is used best on a micro scale. Depending on the lighting, it can pick out the edges of the jaw, chin, wrinkles, etc... These typically are features that aren't as important in general face-detection as there are so many of them, as well as so many variations in every individual's face, that it would lead to an algorithm that was too slow and might only detect the faces of certain people. In other words, too specialized.[9]

Figure 2.2 shows an image transformed to greyscale with few Haar features placed. Note that, the contrast of the images to the right have been turned up to make the difference clear.

Figure 2.2: Haar Features on a face

In the above right most picture, we can see that the algorithm uses edge and line features to classify different transitions on a person's face. The transition

from forehead to the eyebrow, eyes to chicks, upper lip to mouth and the jaw to the chin are classified using edge features. The nose on the other hand is classified using a line feature that uses black-white-black pattern.

As stated above, a picture transformed to gray scale would not look like the picture in figure 2.2. This is not a transformation that happens in the process. Instead, it would look for features on an image like this figure 2.3 below.

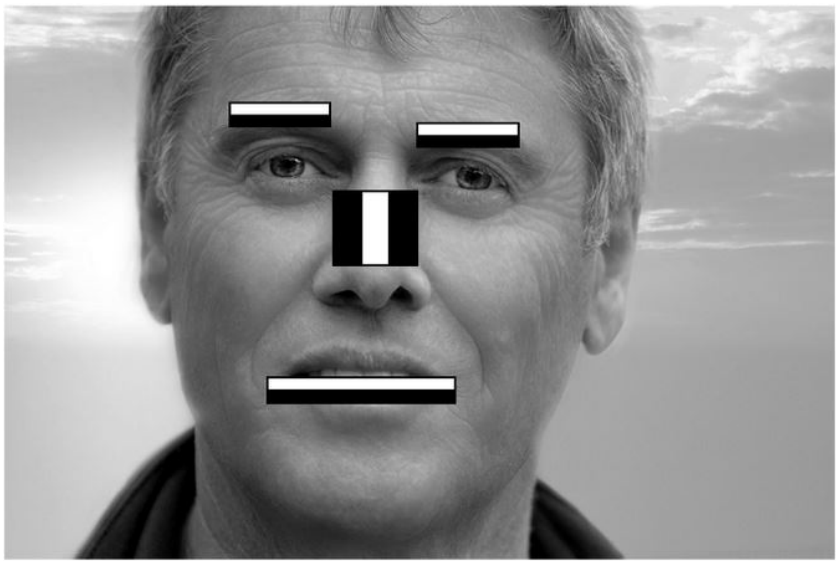

Figure 2.3: Haar Features on Grayscale image

To classify part of an image as black and white and assign the appropriate feature, the algorithm does the evaluation as follows. Consider an image with intensity values ranging from 0 to 1. The algorithm is taught that, in a Haar-like feature, if the difference between the means of the light and dark areas is above some threshold value, treat them as black and white. Let's look at an example using a 6x5 pixels dimension segment of an image.

Let's assume the threshold to be 0.25. We can see that the pixels to the left of the dark line are lighter while the ones to the right darker. The mean of the lighter side is 0.58667 and the mean of the darker side is 0.28. The difference between these values is 0.30667, which is greater than the threshold value. Therefore, the algorithm treats this segment as a feature that looks like the first Haar feature in figure 2.1.

The need for the second stage arises from the problem explained next. [10] Since these frames are scalable, there are thousands of combinations and places where features fit within even a tiny image. A 24x24 pixel image contains over 180,000 possible features. The frames within that image can be between 1x2 (or 2x1) pixels and 24x24 pixels.

To better illustrate this problem, consider this example. Let's say that the amount of time it takes for a computer to calculate the mean of a single

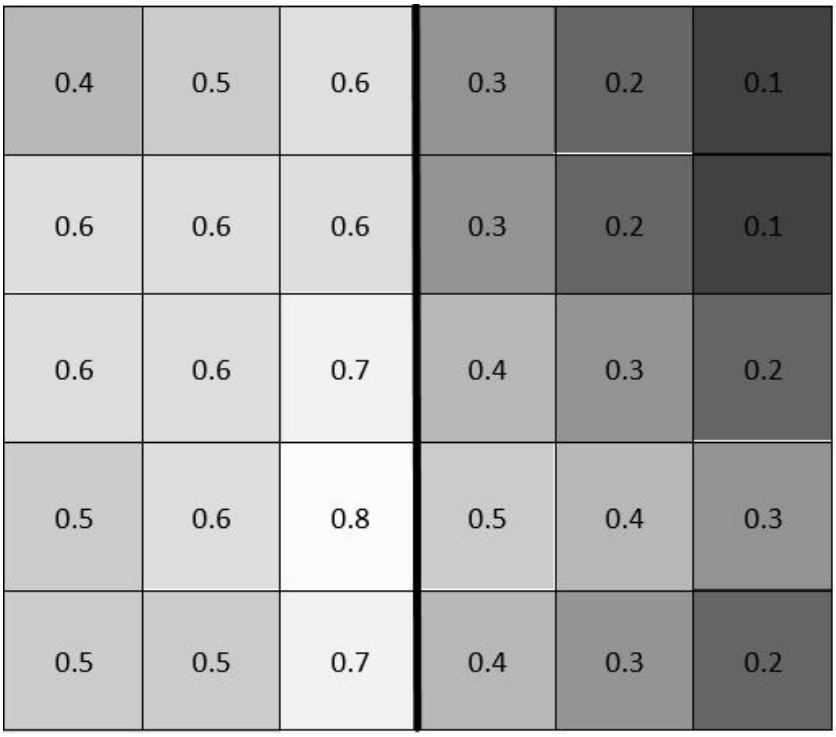

Figure 2.4: Image segment

feature is 1 millisecond (this would be incredibly slow, but let's run with it). Since there are over 180,000 features in a 24x24 pixel image, it would take 180,000ms, 3 minutes, to calculate all the features in it. That's not including processing power and memory used up to complete it

The amount of time explodes even more when you consider that, these days, you're most likely going to be using an image with a resolution of 1920x1080 pixels. To reduce the amount of time and resources used up, we need to find a way to reduce the complexity of the task. How did Viola and Jones, authors of the paper "Rapid Object Detection using a Boosted Cascade of Simple Features", accomplish this to such a degree that their algorithm could detect faces in real-time on a digital camera? [10] Enter the integral images.

ii *Creating Integral Images*

An integral image is an image that takes the value of a pixel as the sum of pixel values from the original image. The value of a pixel is equal to the sum of all the pixels located to the top and left of that pixel, including that pixel itself. To understand this, lets consider an image of 10x8 pixels dimension.

This image uses values from 0 to 255 to represent the intensity values of its pixels. To get the integral image, the algorithm moves through all the pixels and assign a new value for that pixel. This value is the sum of pixel values that are in a rectangular, where the pixel we are evaluating for is located at the bottom right of this rectangle. The rectangle extends to the images left

0	1	3	5	2	7	10	7	10	9
3	7	6	9	3	8	1	8	5	8
1	11	0	7	13	2	14	2	13	1
14	3	2	7	1	0	9	7	2	12
1	5	15	3	6	6	5	1	10	6
8	1	2	6	7	3	2	11	0	15
7	7	6	0	9	5	10	3	8	1
12	5	6	10	11	3	6	7	9	1

Figure 2.5: Original Image

and top side, bounded by a horizontal and vertical line starting from the pixel on evaluation. The image below illustrates this.

On figure 2.6, the sum of pixel values in the shaded region will be the corresponding value of that pixel, pixel with value 2, in the integral image. The corresponding value would then be 51. The algorithm goes through all the pixels and performs the same operation. The integral image looks like the one given on figure 2.7.

Previously we have stated that, for a 24x24 pixel image. There are 180,000 features. Which means that, the algorithm must perform addition for different sized frames again and again to find the means of each side of a feature. This takes a lot of processing time. Here is where the integral image comes in. The integral image reduces the number of additions that needs to be performed to a maximum of four. Whether the dimension of the selected area is 24x24 pixels or 1000x1000 pixels, only four numbers are required to find the area's sum. To make this clear, let's use the 10x10 image example used before.

Consider the 3x5 pixels region in the red rectangle. We can see that the sum of the pixel values in this region is 83. Finding this value required 14 addition operations. Let's see how many addition operations it takes using the integral image.

Clearly the value inside the blue square is not the right answer. This is because, it is the sum of all the pixels the left and top of it. That means it contains pixel values to the left and top of the shaded region. To get the right value, we need to subtract the sum of the regions outside of the shaded one.

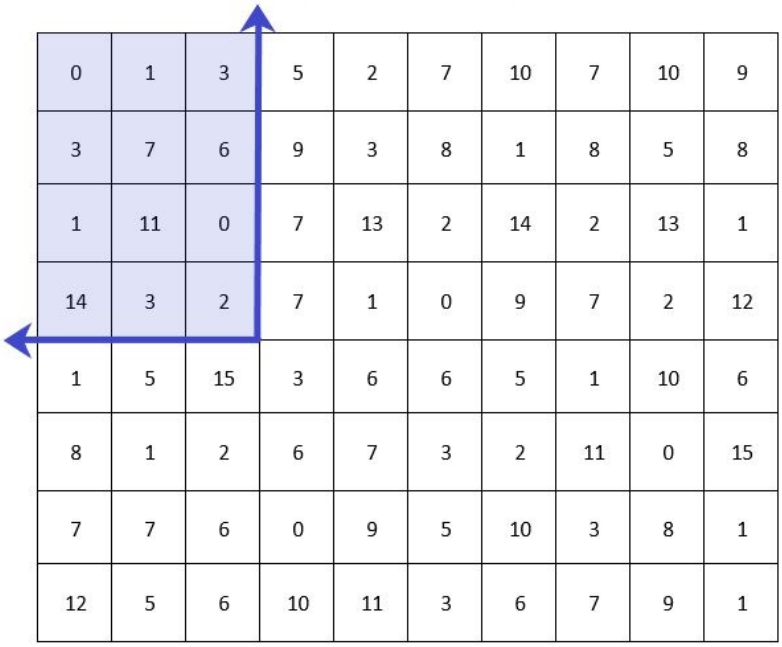

Figure 2.6: Evaluating Pixel Integral Value

To get the sum of the pixels inside the red rectangle, all we need is to do is subtract the sum of the pixel values in these two regions from the integral value of the pixel inside the blue square in figure 2.9. One other thing to consider is that these two regions have overlapping area. To account for double subtraction of this region, we need to add this region's sum value once. Therefore, the pixels to consider are as given on figure 2.12 below.

So, instead of performing 14 addition operations, the algorithm now will perform only 4. That is 215-72-80+20 = 83. Note that we add the region with sum 20, because the region with sum value 72 and 80 both contain this region.

Reducing from 14 addition operation to 4 might not seem like much improvement, but what you must consider is larger frames. For example, consider a 24x24 dimension frame. It contains a total of 576 pixels, which would mean 575 addition operations. In this case, it saves time required to perform 571 to 574 addition operations. Additionally, such kind of operation is performed for so many features within a single image. This process drastically reduces the complexity of the task, which reduces the amount of time and resources needed to compute.

iii *Using Adaboost* Boosting algorithms are one of the most widely used algorithm in data science competitions used to improve the accuracy of models. AdaBoost, short for Adaptive Boosting, is a popular boosting technique which helps you combine multiple "weak classifiers" into a single "strong classifier". It essentially chooses the best features and trains the classifiers to use them.

0	1	4	9	11	18	28	35	45	54
3	11	20	34	39	54	65	80	95	112
4	23	32	53	71	88	113	130	158	176
18	40	51	79	98	115	149	173	203	233
19	46	72	103	128	151	190	215	255	291
27	55	83	120	152	178	219	255	295	346
34	69	103	140	181	212	263	302	350	402
46	86	126	173	225	259	316	362	419	472

Figure 2.7: Integral Image

A weak classifier is a classifier that performs poorly, but still performs better than random guessing. The accuracy of these classifiers is slightly greater than 50%, which is small. Take for example classifying a person as male or female based on their height. A weak classifier for this would be if take anyone under 170cms to be female and anyone above it as male. This classifier would classify so many people in a wrong category, but its still better than randomly assigning a class. A strong classifier is on the other hand is a model for classification that achieves arbitrarily good performance, a lot better than random guessing.

Weak learners are created by moving a window over the input image, and computing Haar features for each subsection of the image. This difference is compared to a learned threshold that separates non-objects from objects. This is basically what we have seen in the first section.

AdaBoost can be applied to any classification algorithm, so it's really a technique that builds on top of other classifiers as opposed to being a classifier itself. This algorithm has two main functions.

- It chooses the training set for each new classifier that is trained based on the results of the previous classifier
- It determines how much weight should be given to each classifier's proposed answer when combining the results

0	1	3	5	2	7	10	7	10	9
3	7	6	9	3	8	1	8	5	8
1	11	0	7	13	2	14	2	13	1
14	3	2	7	1	0	9	7	2	12
1	5	15	3	6	6	5	1	10	6
8	1	2	6	7	3	2	11	0	15
7	7	6	0	9	5	10	3	8	1
12	5	6	10	11	3	6	7	9	1

Figure 2.8: Selected area

0	1	4	9	11	18	28	35	45	54
3	11	20	34	39	54	65	80	95	112
4	23	32	53	71	88	113	130	158	176
18	40	51	79	98	115	149	173	203	233
19	46	72	103	128	151	190	215	255	291
27	55	83	120	152	178	219	255	295	346
34	69	103	140	181	212	263	302	350	402
46	86	126	173	225	259	316	362	419	472

Figure 2.9: Selected Region Shaded

0	1	3	5	2	7	10	7	10	9
3	7	6	9	3	8	1	8	5	8
1	11	0	7	13	2	14	2	13	1
14	3	2	7	1	0	9	7	2	12
1	5	15	3	6	6	5	1	10	6
8	1	2	6	7	3	2	11	0	15
7	7	6	0	9	5	10	3	8	1
12	5	6	10	11	3	6	7	9	1

Figure 2.10: Region 1

0	1	3	5	2	7	10	7	10	9
3	7	6	9	3	8	1	8	5	8
1	11	0	7	13	2	14	2	13	1
14	3	2	7	1	0	9	7	2	12
1	5	15	3	6	6	5	1	10	6
8	1	2	6	7	3	2	11	0	15
7	7	6	0	9	5	10	3	8	1
12	5	6	10	11	3	6	7	9	1

Figure 2.11: Region 2

0	1	4	9	11	18	28	35	45	54
3	11	20	34	39	54	65	80	95	112
4	23	32	53	71	88	113	130	158	176
18	40	51	79	98	115	149	173	203	233
19	46	72	103	128	151	190	215	255	291
27	55	83	120	152	178	219	255	295	346
34	69	103	140	181	212	263	302	350	402
46	86	126	173	225	259	316	362	419	472

Figure 2.12: 4 points of consideration

3

SYSTEM COMPONENTS

3.1 HARDWARE

3.1.0.1 *Breadboard*

The breadboard is a way of connecting electronic components to each other without having to solder them together. They are often used to test a circuit design before creating a PCB.The holes on the breadboard are connected in a pattern. Figure 3.1 shows this pattern.

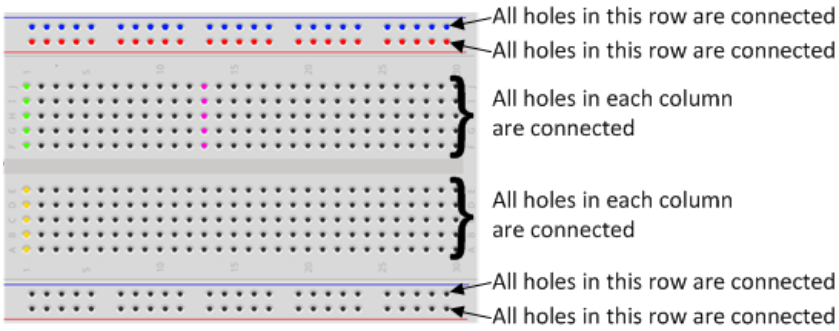

Figure 3.1: Breadboard

3.1.0.2 *Jumper Cables*

Jumper wires are used on breadboards to make connections. For this project, we used two kinds of jumper cables. Wires pulled out of an Ethernet cable and a standard jumper cable. The standard jumper cables have "Male" and "Female" types. This category comes from the way the tips of these cables are oriented. There are also jumper cables with both "Male" and "Female" type tips. The pins on the raspberry pi are compatible with "Female" tips.

3.1.0.3 *LED*

An LED is an electronic component which emits light when it receives electricity. Compared to incandescent bulbs, LEDs are more efficient as most of the electricity is converted into light does not heat. Different LEDs can give off different color of light. It merely depends on the chemical composition of the semiconducting material.

LED has one leg shorter than the other. The longer leg known as the anode is always connected to the positive supply of the circuit. The shorter leg known as the cathode is connected to the negative side of the power supply, known as ground.

Figure 3.2: Ethernet Wires

This is because the LED is a kind of diode thus electrical polarity must be taken into consideration.

The conventional current enters the LED, a polarized electrical device through its anode and leaves through the cathode. The current in an LED or other diodes rises exponentially with the applied voltage so a small change in voltage can cause a large change in current. Current through the LED must be regulated by an external circuit such as a constant current source to prevent damage[11]. Since most common power supplies are nearly constant-voltage sources, LED fixtures must include a power converter, or at least a current-limiting resistor.

Unlike a traditional incandescent lamp, an LED will light only when voltage is applied in the forward direction of the diode. No current flows and no light is emitted if voltage is applied in the reverse direction.

3.1.0.4 *Resistor*

Resistors are needed to connect LEDs to the GPIO pins of the Raspberry Pi. This is because the LEDs would draw more current burning out the Raspberry Pi in the process. Thus, resistors in the circuit ensure the Raspberry Pi will not be damaged.The measure of resistance is called the Ohm (Ω), and the larger the resistance, the more it limits the current. The value of a resistor is marked with colored bands along the length of the resistor body. It does not matter which way round you connect the resistors. Current flows in both ways through them.

3.1.0.5 *GSM Module*

A GSM modem or GSM module is a hardware device that uses GSM mobile telephone technology to provide a data link to a remote network. From the view of the mobile phone network, they are essentially identical to an ordinary mobile phone, including the need for a SIM to identify themselves to the network. GSM modems

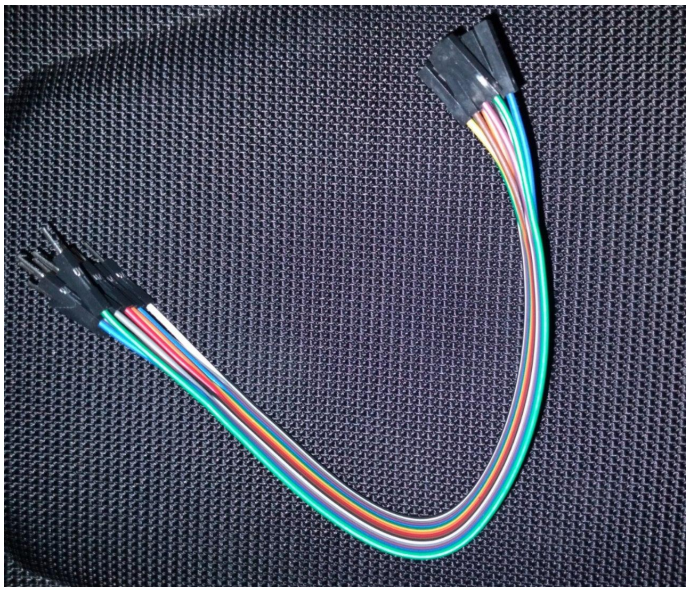

Figure 3.3: Standard Jumper Cables

typically provide TTL-level serial interfaces to their host. They are usually used as part of an embedded system.

3.1.0.6 *Diode*

The SIM800L requires a power voltage of 3.4V to 4.4V, with an optimal voltage of 4V. The raspberry pi provides a voltage of either 3.3V or 5V. Which means both voltage values that the raspberry pi provides are not conducive for the GSM module, specifically the SIM800L. To power the module with the appropriate voltage, we use a diode.

A diode is known for dropping voltage due to the resistive nature of the depletion layer. For a silicon diode, this voltage drop is approximately 0.7volts. Adding to its simplicity and inexpensiveness, the fact that we are looking to decrease the raspberry pi voltage by at least 0.6 volts makes this device a good choice. The 1N4000 series are the common diodes found in the market. For this project we use 1N4007 diode.

3.1.0.7 *Capacitor*

The power consumption of a GSM module varies; low power consumption when the device is ideal and peak power consumption when the device is performing communication like sending or receiving SMS. To account for this variation in power consumption, we can use the capacitor.

One of the most known function of the capacitor is power stabilization. It keeps the power of a circuit junction stable by charging themselves when there is too much power and discharge when there is shortage. Therefore, we connect a capacitor of 6.3V, 100 µF in parallel with Vcc and ground pin of the GSM module.

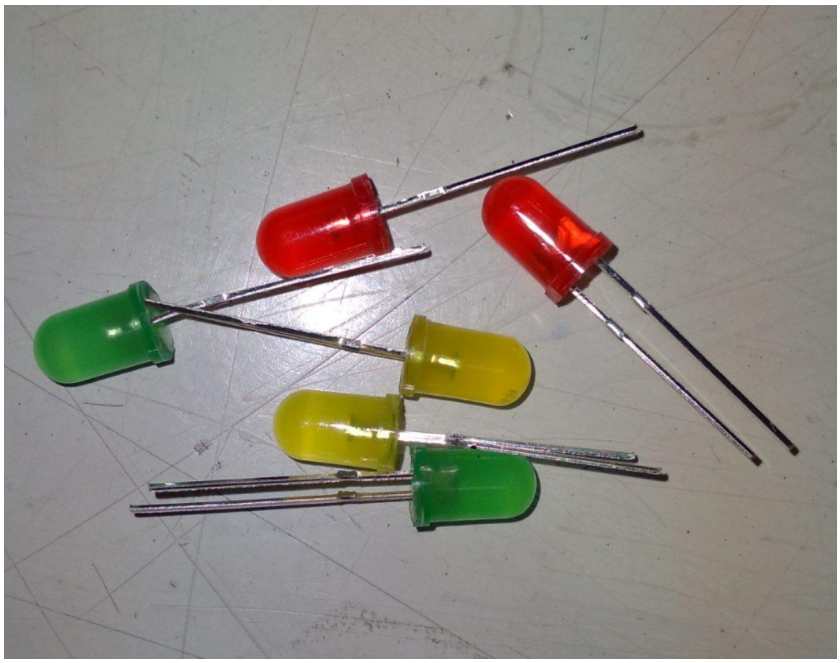

Figure 3.4: LEDs

3.1.0.8 *Camera*

The raspberry pi needs a device that can give real time image of the crossroad so that it can use it as an input to run its vehicle detection algorithm. For a practical application of this project, a camera dedicated for such kind of purpose called surveillance camera can be used. In this project though, we will not be interfacing this device with the raspberry pi.

3.1.0.9 *Raspberry Pi*

This project utilizes the Raspberry Pi 2 Model B. The Raspberry Pi 2 Model B is the second-generation Raspberry Pi. It is based on the BCM2836 system-on-chip (SoC), which includes a quad-core ARM Cortex-A7 processor and a powerful GPU. It weights 42g and has a dimension of 5 x 4 x 3 inches. The raspberry pi 2 model B is illustrated on the figure 3.9 below.

The specifications of this model are:

- 900 MHz quad-core ARM Cortex-A7 CPU
- 1 GB of RAM
- Video Core IV 3D graphics core
- Ethernet port
- HDMI output
- Audio output
- RCA composite video output

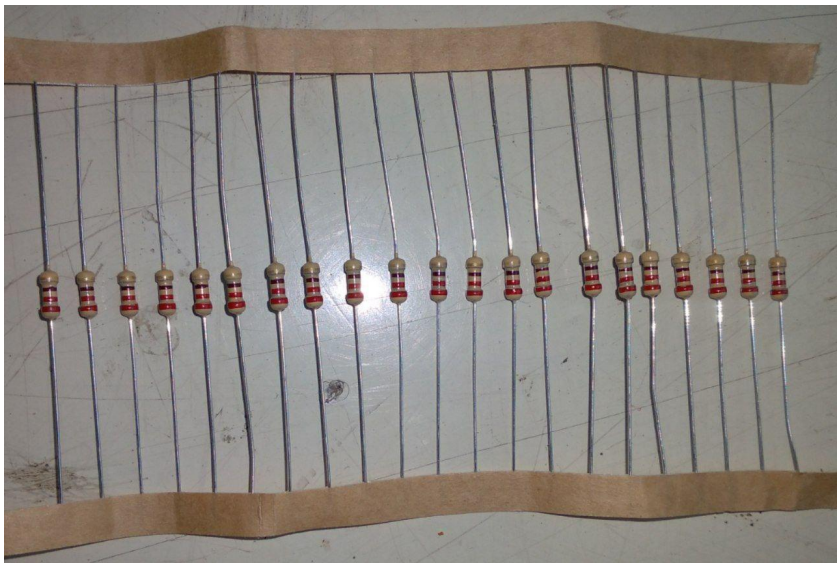

Figure 3.5: Resistors

- Four USB ports
- MicroSD card reader
- 40 pins

[12] A powerful feature of the Raspberry Pi is the row of GPIO (general-purpose input/output) pins along the top edge of the board. A 40-pin GPIO header is found on all current Raspberry Pi boards (unpopulated on Pi Zero and Pi Zero W). Prior to the Pi 1 Model B+ (2014), boards comprised a shorter 26-pin header. Figure 3.10 below illustrates the details of the pins.

We can do programming by either opening the CLI or an IDE under the PI icon on top left. To do this, it is essential to know the pin configuration. Figure 3.10 below illustrates this.

Voltages

Two 5V pins and two 3V3 pins are present on the board, as well as a number of ground pins (0V), which are unconfigurable. The remaining pins are all general purpose 3V3 pins, meaning outputs are set to 3V3 and inputs are 3V3-tolerant.

Outputs

A GPIO pin designated as an output pin can be set to high (3V3) or low (0V).

Inputs

A GPIO pin designated as an input pin can be read as high (3V3) or low (0V). This is made easier with the use of internal pull-up or pull-down resistors. Pins GPIO2 and GPIO3 have fixed pull-up resistors, but for other pins this can be configured in software.

More

As well as simple input and output devices, the GPIO pins can be used with a variety of alternative functions, some are available on all pins, others on specific pins.

Figure 3.6: GSM SIM800L

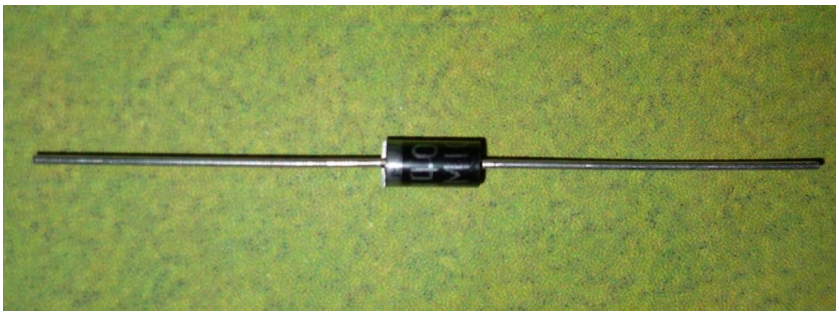

Figure 3.7: 1N4007 Diode

- PWM (pulse-width modulation)
  - Software PWM available on all pins
  - Hardware PWM available on GPIO12, GPIO13, GPIO18, GPIO19
- SPI
  - SPI0: MOSI (GPIO10); MISO (GPIO9); SCLK (GPIO11); CE0 (GPIO8), CE1 (GPIO7)
  - SPI1: MOSI (GPIO20); MISO (GPIO19); SCLK (GPIO21); CE0 (GPIO18); CE1 (GPIO17); CE2 (GPIO16)
- I2C
  - Data: (GPIO2); Clock (GPIO3)
  - EEPROM Data: (GPIO0); EEPROM Clock (GPIO1)
- Serial
  - TX (GPIO14); RX (GPIO15)

Figure 3.8: Capacitor

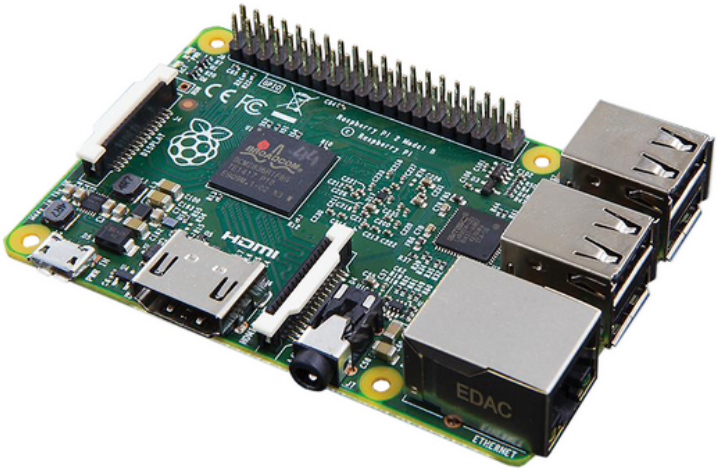

Figure 3.9: Raspberry Pi 2 Model B

Warning: while connecting simple components to the GPIO pins is perfectly safe, it's important to be careful how you wire things up. LEDs should have resistors to limit the current passing through them. Do not use 5V for 3.3V components. Do not connect motors directly to the GPIO pins, instead use an H-bridge circuit or a motor controller board.[12]

3.2 SOFTWARE AND TOOLS

3.2.0.1 *Raspberry Pi OS*

Raspberry Pi needs an Operating System to operate. Raspberry pi OS, previously known as Raspbian, is the officially supported operating system. To add this operating system, a microSD card is required, hence the microSD card reader on the list of specifications stated above. Earlier models Raspberry Pi used a full-sized SD card.

Because of a hardware limitation in the Raspberry Pi Zero, 1 and 2, the boot partition on the SD card must be 256GB or smaller otherwise the device will not boot up. Later models of Raspberry Pi 2 with a BCM2837 SoC, Raspberry Pi 3, 4 and 400 do not have this limitation. This does not affect Raspberry Pi OS, which always uses a small boot partition.

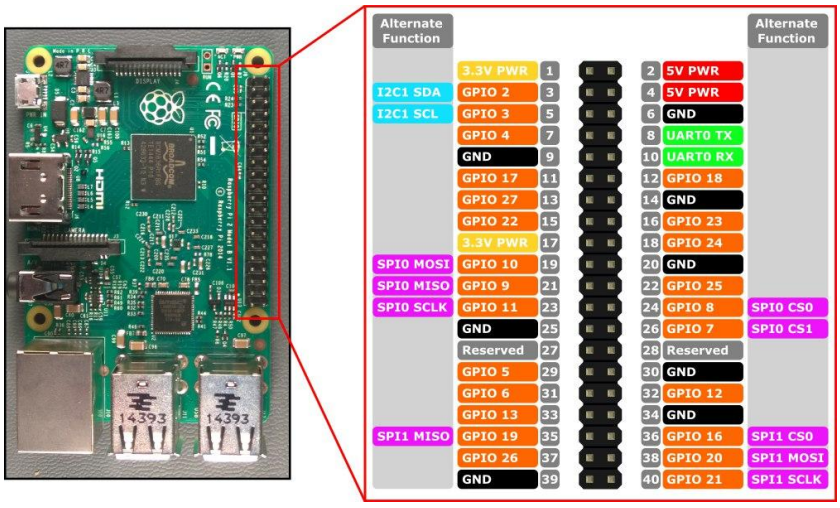

Figure 3.10: Raspberry Pi 2 pin Configuration

An SD card of 8GB or greater capacity is recommended to use for the Raspberry Pi OS. However, the lite version of Raspberry Pi OS requires lesser memory size of 4GB. Other operating systems have different requirements: for example, LibreELEC can run from a smaller card.

3.2.0.2 *Raspberry Pi Imager*

An imager provided by Raspberry Pi official website called Raspberry Pi Imager can be used to easily install the Raspberry Pi OS on the microSD card. This software is available for different platforms including windows, macOS and Linux. After downloading the software compatible for the platform at hand, we can run it and the window on figure 3.11 will appear.

Figure 3.11: Raspberry Pi OS Imager

It is possible to select from the different OS variations available on the "CHOOSE OS" dropdown. It is also possible to download the file directly from the website and select it here from our local machine. After connecting the microSD card on our personal computer, we can select it on the "CHOOSE STORAGE" dropdown and press the "WRITE" button, which will be active after selecting the appropriate OS and storage device. Once complete, we can plug the microSD card on the microSD card reader located at the back of the Raspberry Pi board.

We can use two different ways to control the Raspberry Pi. One is by using the different ports provided by the Raspberry Pi board to connect IO devices. That is, connecting a conventional keyboard and mouse to the USB ports and a desktop monitor, TV, or any display with HDMI input to the HDMI output of the Raspberry Pi. This way we can use the Raspberry Pi as a personal computer.

Another option is to perform a Raspberry Pi headless setup and use a personal computer to control the Raspberry Pi. To do this, we need two tools:

- An SSH Client
- VNC Server

3.2.0.3 *SSH Client*

An SSH client is a program that allows establishing a secure and authenticated SSH connections to SSH Servers. The server in this case is the raspberry pi. PuTTY is an SSH and telnet client, developed originally by Simon Tatham for the Windows platform. PuTTY is open-source software that is available with source code and is developed and supported by a group of volunteers. Figure 3.12 shows this software.

After installing the OS on the microSD, we must first open the boot partition of the microSD and add an extensionless file called "ssh". The Raspberry Pi OS will automatically enable SSH on first boot-up. This will allow us to access the raspberry pi's CLI using putty. To do this, we must follow the following steps.

- First use an ethernet cable to connect the Pi to a personal computer.
- On Putty enter raspberrypi or raspberrypi.local as the host name of the connection we want to create.
- The above step will open a window prompting us to login. By default, the username is Pi and the password is raspberry, which can later be changed.

This will give us access to the Raspberry pi CLI. It is possible to stop here and use just the CLI or continue, to access the Pi's desktop.

3.2.0.4 *VNC Viewer*

VNC is a graphical desktop-sharing system that uses the Remote Frame Buffer protocol to remotely control another computer. It transmits the keyboard and mouse

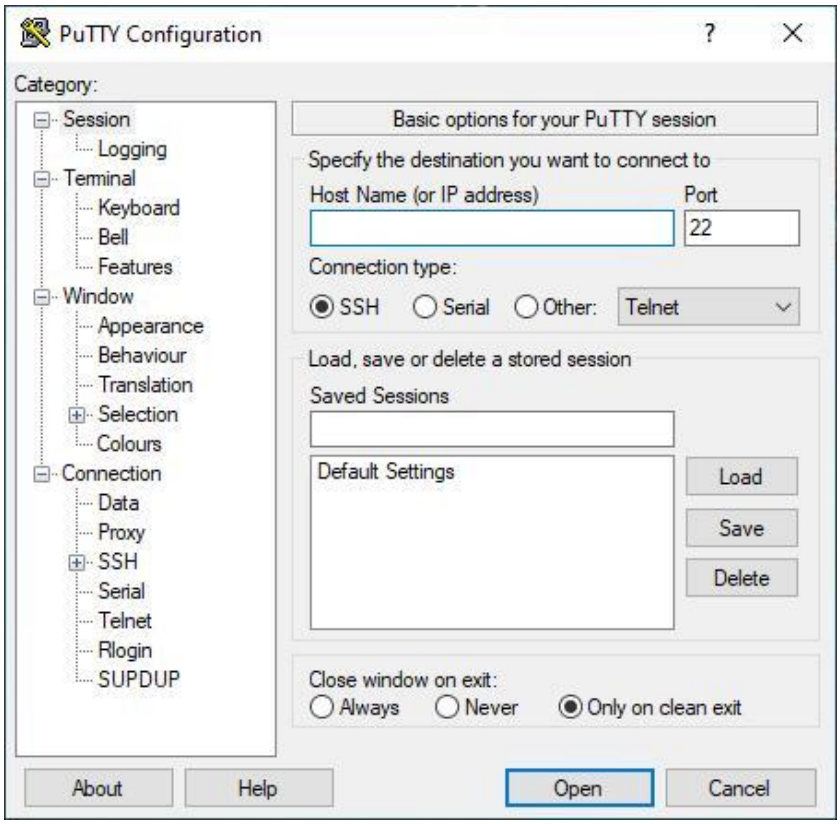

Figure 3.12: Putty

input from one computer to another. To use this tool, we first must enable VNC setting in the raspberry pi.

- On the raspberry pi CLI, enter the command “sudo raspi-config”. This opens a configuration window.
- To enable VNC, navigate to Interfacing Options > VNC select it and select Yes.

After enabling VNC setting. Run the VNC viewer and select “New connection” from the “File” menu. The window given on figure 3.13 will open.

- Enter raspberrypi.local or raspberrypi in the "VNC Server" field.
- Double click the connection icon.
- Enter Pi’s username and password in the dialog box pops up. Use either the default or the one you entered if it was changed previously.

The Raspberry pi desktop will then appear. It is given on figure 3.14.
Note that we can also use Wi-Fi and direct USB connection to do this. Even though some of steps are shared, both connections have their own setup.

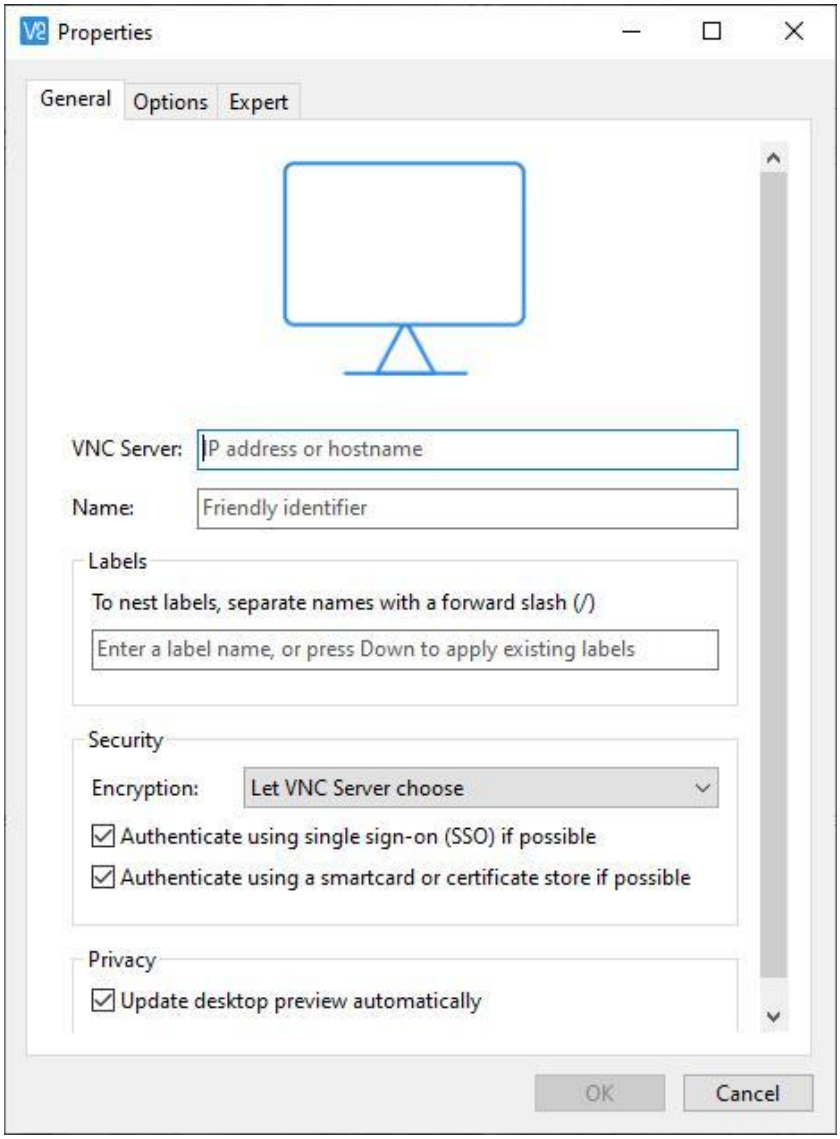

Figure 3.13: VNC Viewer New Connection

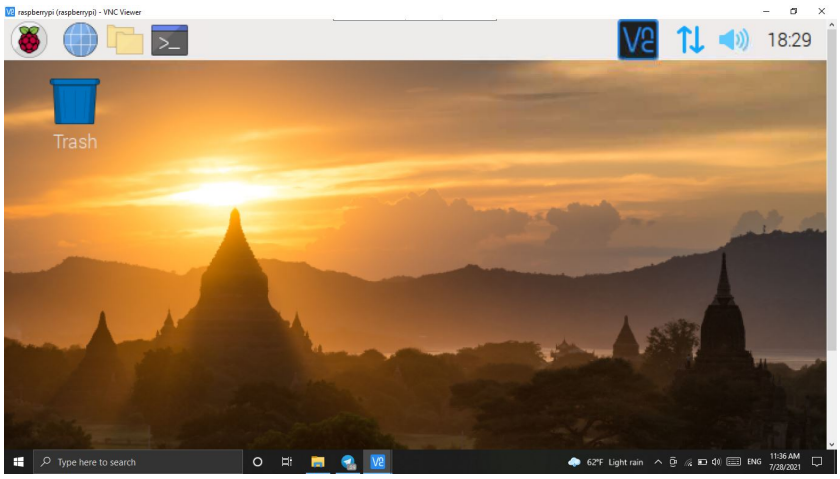

Figure 3.14: Raspberry Pi Desktop

4

DESIGN

4.1 SYSTEM DESIGN

4.1.1 *Block Diagram*

The hardware components required to build this system are given in section 3.1 with a brief description. At high-level, the system can be described by the following basic entities.

- Micro-controller
  - Object Detection Algorithm
  - Scheduling Algorithm
  - Distance Measuring Algorithm
- USB Camera
- GSM Module
- Traffic Light
- API

Let us see how these components are connected to each other to form the smart traffic control system. The block diagram below illustrates the high-level interfacing of these components.

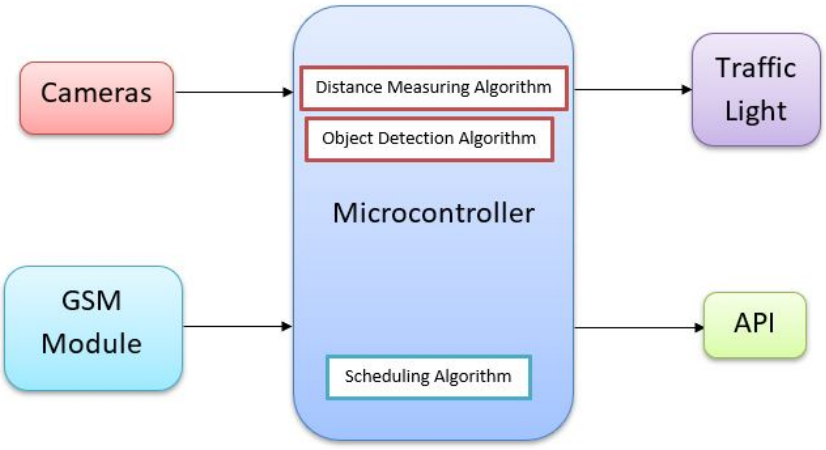

Figure 4.1: Block Diagram

4.1.2 *Description*

As illustrated on figure 4.1, this system has four basic hardware and four basic software components. The core of the system, the micro-controller, runs three of these major algorithms.

The first one is the object detection algorithm. This and the distance measuring algorithm work hand in hand with the cameras. The cameras feed the current condition of the crossroad to the micro-controller. The micro-controller can either use a real-time image or take photos within a certain time interval. The micro-controller can then use the data coming from the cameras as an input for the vehicle detection algorithm. This will give the micro-controller the number of cars on each side of the crossroad. Additionally, the micro-controller can also use data coming from the camera to determine the amount of accepting road available on each side of the crossroad.

Taking the above two parameters as an input, the scheduling algorithm will allocate some amount of time to selected lanes. Other than the parameters stated previously, this algorithm uses additional parameters to select the lanes and allocate time for them. We will see the details of these parameters and the decision-making process in sections to follow.

The normal operation of the scheduling algorithm can be interrupted by two conditions. These conditions are:

- Receiving SMS from the GSM Module
- Detection of emergency vehicles

Upon receiving an SMS, the microcontroller will decode the message and give way for lanes for specified amount of time. This feature is meant to be used by authorities or traffic police when an exceptional condition occurs. For example, if diplomats, military convoy, higher government authorities, bank money moving vehicles and the like are crossing. In such a case, priority will be given by sending an SMS message to the system. The system will halt the current allocation and give way for the priority vehicles until they have crossed, at which point the system will resume from the previous state of the crossroad.

The second exceptional case is when an emergency vehicle like ambulance or fire-fighting truck is detected. Just like the previous case, the system will give way for the side containing an emergency vehicle. After the emergency vehicle has passed, that is when it is no longer within the sight of the cameras, the system will resume with its normal routine of operation.

In all the above cases, the microcontroller will send control signal to the traffic light. The system will refrain from sending signal to the traffic lights that will cause the vehicles from crossing the crossroad.

The last feature of this system is sending of traffic record an API. The system can do this by either using a GSM module or an ethernet cable-based connection established to it. This record will contain a time stamp and the number of vehicles. This can be used as a statistical data by the Ethiopian road authority.

Based on this data, different decisions like widening the roads, using overpasses, adding traffic lights and the like can be taken. This will assist in decreasing traffic congestion.

4.1.3 *Wiring Diagram*

Interfacing of this project has two core parts. These are:

- Raspberry Pi with GSM module and related components
- Raspberry Pi with LEDs and related components

To interface the GSM SIM800L module to the raspberry pi, a capacitor and a diode is required besides the two components. The reasons for using these components are described in detail on section 3.1.6 and 3.1.7. Figure 4.2 below illustrates how these components are interfaced.

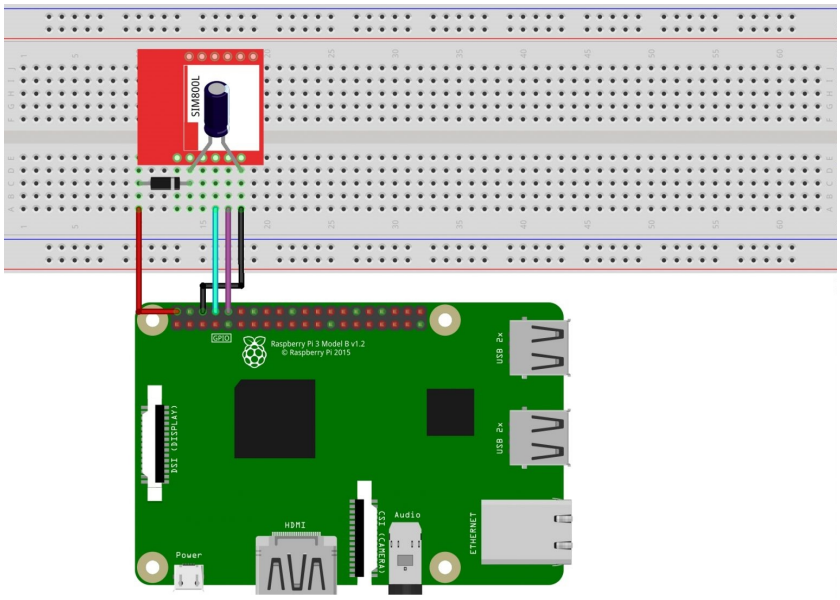

Figure 4.2: Raspberry Pi GSM Interfacing

On figure 4.2, red wire is +5V and black wire is ground. We can observe that the capacitor is connected in parallel with Vcc and ground, as described previously. The +5V source from the raspberry pi is connected to the GSM module Vcc through a diode. The pin right below the ground pin on the raspberry pi or the one connected to a cyan colored jumper color, Tx, is connected to RXD of the GSM module. The pin next to Tx connected to a purple wire, Rx, is connected to TXD

of the GSM module. Note that both the capacitor and the diode should be connected in the right polarity. In figure 4.2, the capacitor negative side is indicated by a white strip at the top right edge. Likewise, the diode's negative side or the cathode is indicated with a white line at the right side of the diode.

To interface the LEDs to the raspberry pi, a current limiting resistor is required. Figure 4.3 below illustrates how these components are interfaced.

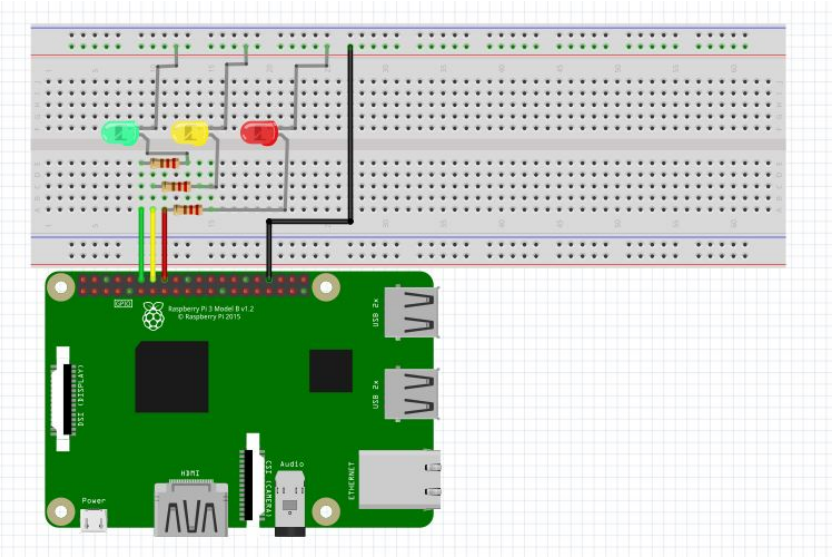

Figure 4.3: Raspberry Pi LED Interfacing

As explained in section 3.1.4, resistors are required to connect a LED to a GPIO pin. In the above picture, we can see that this is done for all three LEDs. Unlike the diode and the capacitor, the resistor is not a polarized element. So, we connect it in any direction we want.

Putting it all together, we get the wiring diagram on figure 4.4 given below. This design considers 8 traffic lights, two for each side of the crossroad. One of these traffic lights is for cars turning left while the other one is for cars going straight. This will be discussed in detail on section 4.2.2.

4.2 SOFTWARE DESIGN

4.2.1 *Coding platform and Selection*

This project involves programming on detection and counting of vehicles, deciding the amount of time to be allocated and to which lanes to allocate this time, receiving SMS message from the GSM module, sending statistical data to an API. All of these are coded using python with the help of libraries like OpenCV. The API was built using Loopback whereas the webpage was built using React.

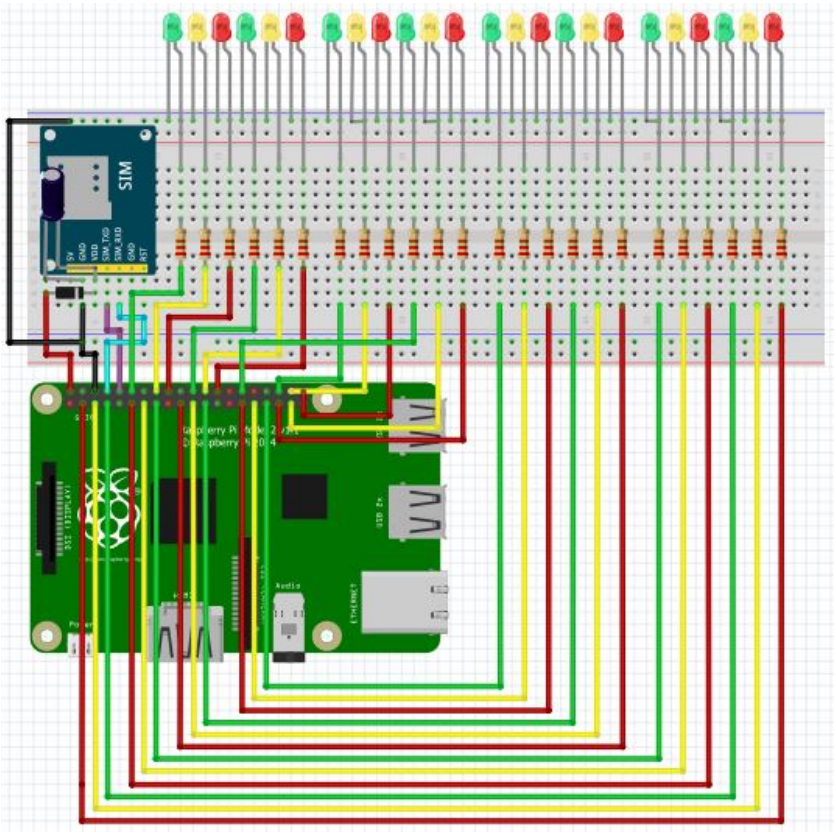

Figure 4.4: System Wiring Diagram

4.2.2 *API*

The API is intended to hold data about the number of cars coming from each side of the crossroad, a time stamp and information about the crossroad. To do this, we will follow database design given on figure 4.5. The design gives the tables, collections for NoSQL, and their attributes.

The first table, CrossRoads, is used to store information above a crossroad. Besides its ID, it has seven properties. The first four properties relate to where the crossroad is located. The fifth property is the name by which people commonly address it, it is optional parameter. The sixth parameter is a timestamp of when the particular crossroad has been registered and the last one is a time stamp of the last time it was updated. This is necessary because some attributes of this table might be changed over time.

The second table, is used to record the number of cars coming toward a crossroad at a particular time. It uses attributes “Side” and “LaneNumber” to identify a particular lane from the eight lanes. Unlike the first table, this table only needs a timestamp of when it was created because it does not change over time.

The two tables maintain a relationship using attribute “CrossRoadId” as a foreign key. The relationship maintained is a one to many, because one crossroad has many vehicles count record.

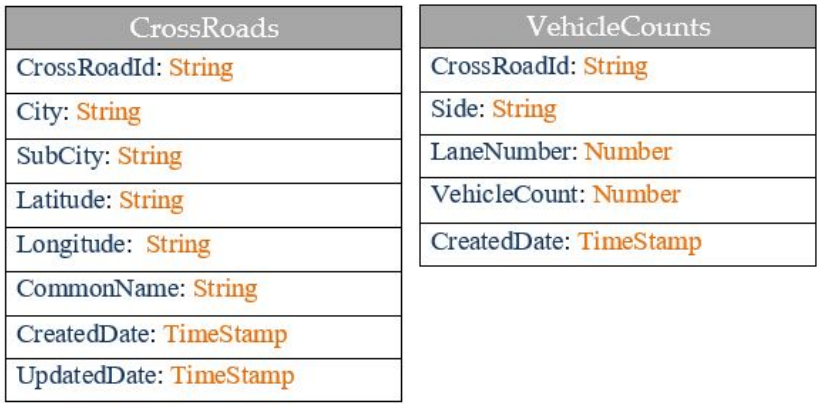

Figure 4.5: DB Collections

4.2.3 *Scheduling Algorithm*

A smart traffic light system needs to have an effective scheduling algorithm to accommodate the different and varying number of vehicles that are trying to use a shared resource. This shared resource is the crossroad. The algorithm needs to consider different parameters to be able to make a fair decision so that cars coming into the crossroad from different sides can use the shared resource effective. The following are the parameters we are going to consider in our design.

- Density of cars
- Waiting time
- Number of cars crossing intersection
- Acceptance capacity of roads after intersection
- Priority vehicles
  - Emergency vehicles
  - Priority vehicles

In scheduling algorithm such as in CPU scheduling, there are a certain property that are considered by the algorithms. Some of the basic properties are:

- *Resource Utilization*: - this refers to the usage of the resource by the entities involved in the process. For example, if we consider CPU scheduling this property corresponds to the computer's usage of the CPU or the amount of work handled by the CPU.
- *Throughput*: - refers to the amount of a product or service that a system can produce and deliver within a specified period. In our previous example, this is the same as the number of processes that are completed per unit time.
- *Waiting time*: - this refers to the total amount of time an entity spends waiting to get a resource. The resource would be the CPU and the entity would be computer processes waiting in the ready queue.

- *Turnaround time*: - refers to the time interval from the submission of an entity to be processed to the time of completion. For a CPU scheduling this is the sum of waiting time, executing on the CPU and doing I/O.
- *Response time*: - refers to the time from the submission of a request to the start of the first response.
- *Fairness*: - refers to the fair share of the resource, without having any entity suffering from indefinite postponement to utilize the resource.

Ideally, the goal of any scheduling algorithm is to optimize all the properties. That is:

- Maximize resource utilization, through put, fairness and efficiency and
- Minimize waiting time, turnaround time and response time.

Unfortunately, this is not possible. What needs to be done is make the right amount of trade off to maximize the algorithms efficiency.

Besides the properties listed above there are a few additional things that needs to be considered. One is the two types of scheduling(interrupt) and the other is deadlock.

1. *Interrupt*: - is a condition where the current use of a resource is stopped by an external entity. This is before the current resource has completed the current allocated time. Based on whether interrupt is allowed in the algorithm, there are two types of scheduling algorithms.
   - *Preemptive scheduling*:- Interrupt is accepted.
   - *Non-Preemptive scheduling*:- Interrupt is not accepted.
2. *Deadlock*: - is a condition that happens with entities sharing a resource where each entity is looking for a resource held by another entity to be released. That is if we have entities ‘a’, ‘b’, ‘c’ and ‘d’. ‘a’ is waiting for ‘b’, ‘b’ is waiting for ‘c’, ‘c’ is waiting for ‘d’, ‘d’ is waiting for ‘a’. The image below illustrates this condition in a crossroad.
3. *Ready Queue*: - a queue is a data structure that allows the retrieval of data in the order of insertion. This method of organizing and manipulating data is known as First in First out or First Come First Serve. Entities in the ready queue are ready and waiting to use the shared resource.

In figure 4.6(b), we can see that car ‘1’ is waiting for car ‘2’, car ‘2’ is waiting for car ‘3’, car ‘3’ is waiting for car ‘4’ and car ‘4’ is waiting for car ‘1’ to move. If we consider a realistic case where many other cars would be lined up behind each of these cars, this situation is hard to resolve. So, it is better to avoid it. Avoiding this situation is known as deadlock avoidance.

The parameters listed above relates to these scientific parameters as explained below.

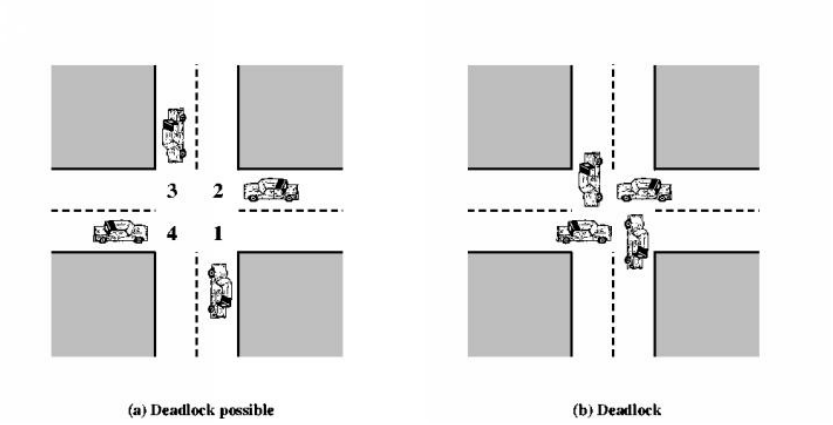

Figure 4.6: Crossroad Deadlock

- *Density of cars*: - this is the number of cars ready and waiting to use the crossroad. That is, we have four ready queues if we consider a road or eight if we consider lanes. Lanes with a greater number of cars are given priority, this does not mean that they get to cross always. Rather we can consider them as lanes with priority.
- *Waiting time*: - this parameter relates to fairness as it is used to avoid indefinite postponement of lanes or sides with less amount of car. To ensure fairness, we assign a maximum waiting time limit.
- *Number of cars crossing the intersection*: - this parameter relates to resource utilization and efficiency. Both need to be maximized. To maximize the number of cars crossing the intersection, we must give permission for two lanes to cross simultaneously. This is possible for a limited combination of lanes. From the labeling given on figure 4.7, the following are the possible lane combinations.
  - $A_2 – C_2$
    $B_2 – D_2$
    $A_1 – C_1$
    $B_1 – D_1$
  - $A_1 – A_2$
    $B_1 – B_2$
    $C_1 – C_2$
    $D_1 – D_2$
  - $A_1 – D_2$
    $B_1 – A_2$
    $C_1 – B_2$
    $D_1 – C_2$

  These lane combinations can be related mathematically as follows:

  - *For x<y*: y = x+4 or
    y = x+1 or
    y = (x+7) mod8
  - *For x<y*: y = (x+4)mod8 or
    y = x-1 or
    y = (x-7) mod8

  By using these mathematical relationships, we can check if two lanes can cross the road simultaneously.

- *Acceptance capacity of roads after intersection*: - if the accepting road of a lane is full and we give green light to it. It will eventually create the condition illustrated by figure 19. This will significantly lower the efficiency of the crossroad usage because no cars will be crossing. Therefore, this parameter is related to deadlock avoidance and efficiency. To avoid this, we must postpone passage for that lane for some amount of time.

- *Priority vehicles*: - the normal operation of the system could be interrupted. This could either one of the following two cases:

  - *Emergency vehicles*: - if the system detects an emergency vehicle on one of the sides, it interrupts the current state and give way for the emergency vehicle. This will be done by giving a green-light for the left most lane of that side, until this vehicle has crossed.
  - *Priority vehicles*: - another way the system could be interrupted is if the authorities forward a command. This command could be given to allow higher officials, convoys and the like to cross.

  The system must resume with the state it was in before it was interrupted once these vehicles have crossed. Since the system supports interrupts, it is preemptive.

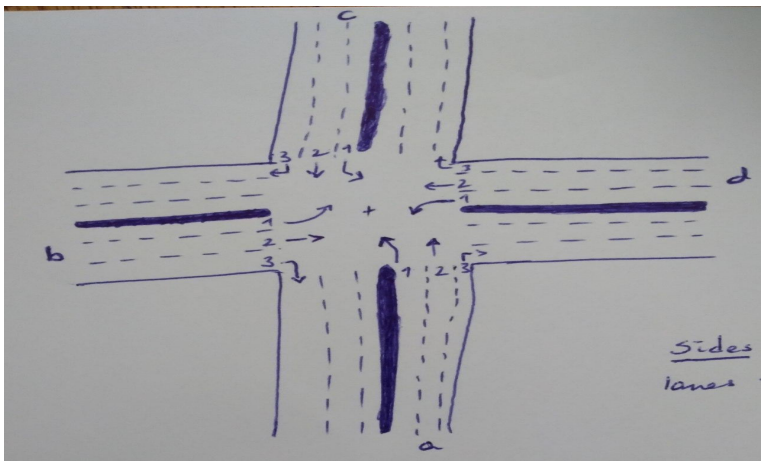

Figure 4.7: Labeled Crossroad

5 IMPLEMENTATION

5.1 SOFTWARE IMPLEMENTATION

5.1.1 *Cascade Classifier*

In our project we needed detect cars and to do that we used Haar feature-based cascade classifier, an object detection method. The algorithm is trained from a lot of positive and negative images. Positive images are images which contain the object to be detected and negative images are those which don't contain the object of interest. They merely contain the background stuff the object of interest can be in.

[13] In order to train the cascade classifier, we used a software tool called Cascade Trainer GUI. It was the simpler alternative. To start the training, you create a folder for your classifier and inside you create another two folders named "n" and "p". The folder labeled n is where you put the negative images and the folder labeled p is where you the positive images. One important point is to never include any positive images in your collection of negative images inside folder "n".

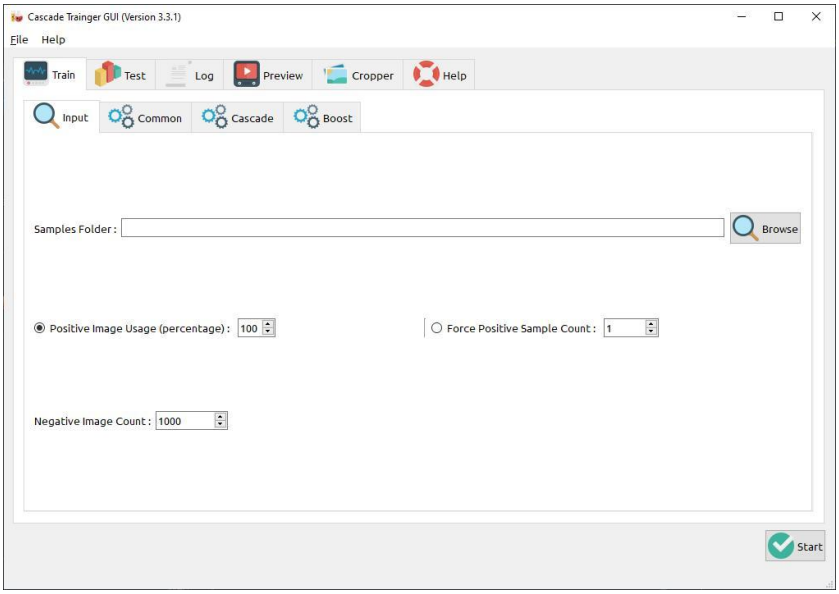

Figure 5.1: Cascade Trainer Input Tab

Common, Cascade and Boost tabs can be used for setting numerous parameters for customizing the classifier training. Cascade Trainer GUI sets the most optimized and recommended settings for these parameters by default, still some parameters need to be modified for each training. Pre-calculation buffer size can be set to help with the speed of training process. For example, if one has an 8 GB

of RAM on their computer then they can safely set both of the buffer sizes below to 2048.

Next sample width and height must be set. They shouldn't be a very big size as it will make detection slow. It is quite safe to always set a small value for this. Recommended settings for sample width and height is that to keep one aspect on 24 and set the other accordingly.

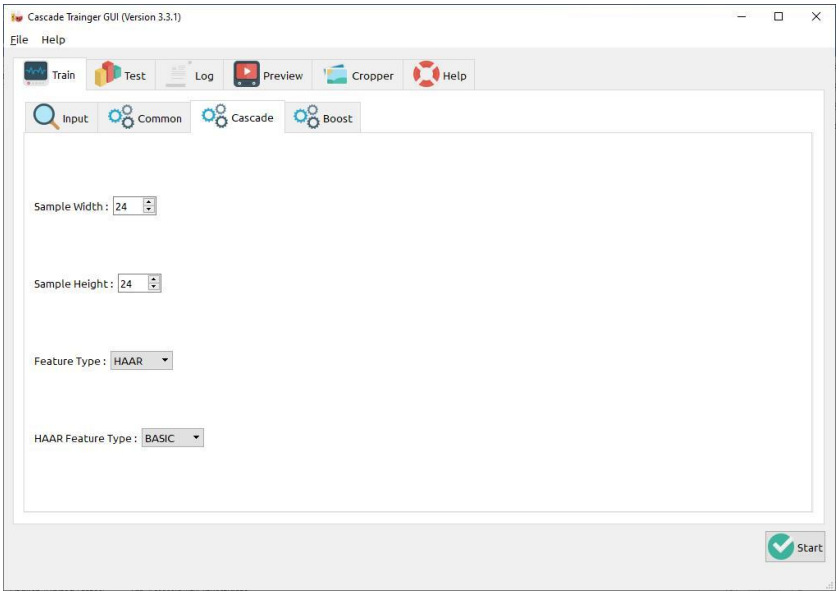

Figure 5.2: Cascade Trainer Cascade Tab

For example, if one has sample images that are 320×240 then first calculate the aspect ratio which in this case is 1.33:1 then multiply the bigger number with 24. It would be 32×24 for sample width and height.

The feature type can be set to HAAR or LBP. HAAR classifiers are very accurate but require a lot more time to train so it is much wiser to use LBP if you can provide your classifiers with many sample images. LBP classifiers on the other hand are less accurate but train much quicker and detect almost 3 times faster.

After all the parameters are set, "Start" button at the bottom is pressed to start training the cascade classifier. The log screen will be shown while training. Wait for the training to complete.

New files and folders will have been created in the classifier folder. The classifier folder will contain XML files that are created during different stages of training. Inside classifier folder, you'll notice something like the following.

- "stage#.xml" files are temporary files that won't be needed anymore.
- "params.xml" contains the parameters you have used for the training. (Just a reminder file)

- "cascade.xml" is the actual cascade classifier and if the training completed successfully then you should have this file inside classifier folder.
- "neg.lst", "pos.lst" and "pos_samples.vec" are temporary files created for training the classifier and they can also be removed without having any effect.

5.1.2 *API*

To implement the API, we used Loopback along with a MongoDB database. Figure 5.3 shows the project structure of the implemented API.

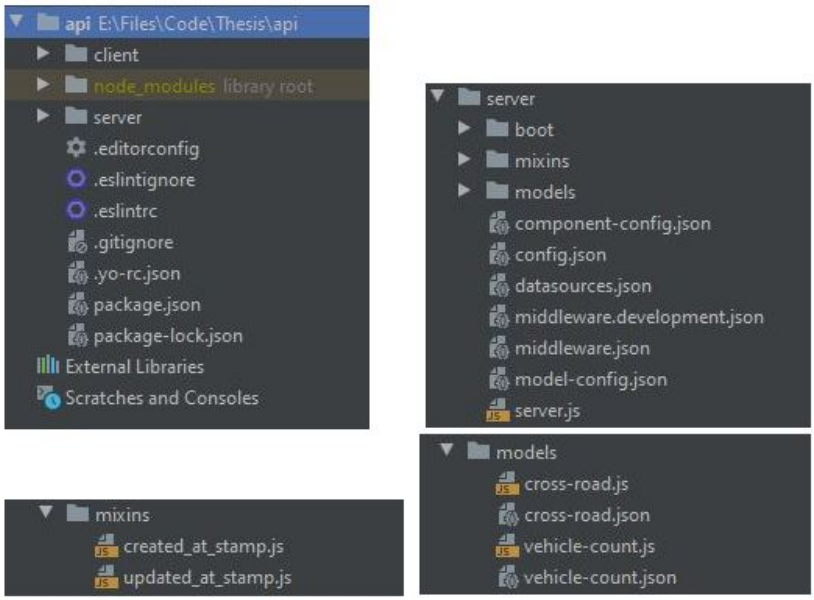

Figure 5.3: API Project Structure

Although some modifications might be done during implementation, the files on the top two pictures are generated by the framework and are used to handle top level configurations like external packages used, data sources, entry point of the program.

The files in the models folder are used to define the collections of the API. The name of the collection, its attributes, its relation with other collections, mixins and other basic definitions of the collection is defined in the ". json" files.

In the ". js" files, remote methods are defined. By default, the framework defines how to handle basic HTTP verbs. But in some cases, applications need additional methods to handle requests. Authentication, authorization, requests that need to access more than one collection are handled by such methods. These methods are called remote methods.

For this project, we have defined one remote method called "getVehicleCountBy-RoadId" in the vehicle-counts.js file. This method is used to retrieve vehicle count along with the crossroad details. The method is given below.

```
Vehiclecount.getVehicleCountByRoadId = async function(id) {
  const crossRoad = await Vehiclecount.app.models.cross_road.findOne( filter: {
    cross_road_id: id,
  });
  const vehicleCountData = await Vehiclecount.find({
    where: {
      cross_road_id: id,
    },
  });
  return {
    type: {
      crossRoadDetails: crossRoad,
      vehicleCountDetails: vehicleCountData,
    }, root: true,
  };
};
```

Figure 5.4: API Remote Method

The two ".js" files in the "mixins" folder are used to add time stamp to data inserted to the collections. As their name indicate, created_at_stamp.js is invoked when a document is created while update_at_stamp.js is used when a document is updated. The collection "vehicle-count" uses only the first mixin while "cross-road" uses both. The reason for this is given on section 4.2.2.

5.1.3 *Controller*

As stated on section 4.2.1., we used python to implement the scheduling algorithm, program that detects cars using the classifier, HTTP client and hardware controller all have been implemented using python. The module structure is shown on figure 5.5 below.

It has four basic directories and one configuration module. The config module has a config file. In it, a dictionary named configs that contains two attributes. These attributes are the API base URL and API access token. Any other configuration of the controller will be added in this dictionary or this config.py file. The remaining four modules are basic to this project. Let us look at each of these directories.

5.1.3.1 *Scheduler*

This module contains the scheduling algorithm and other model classes that assist it. The files within this module are:

- assigned_state.py
- density_state.py
- scheduler.py

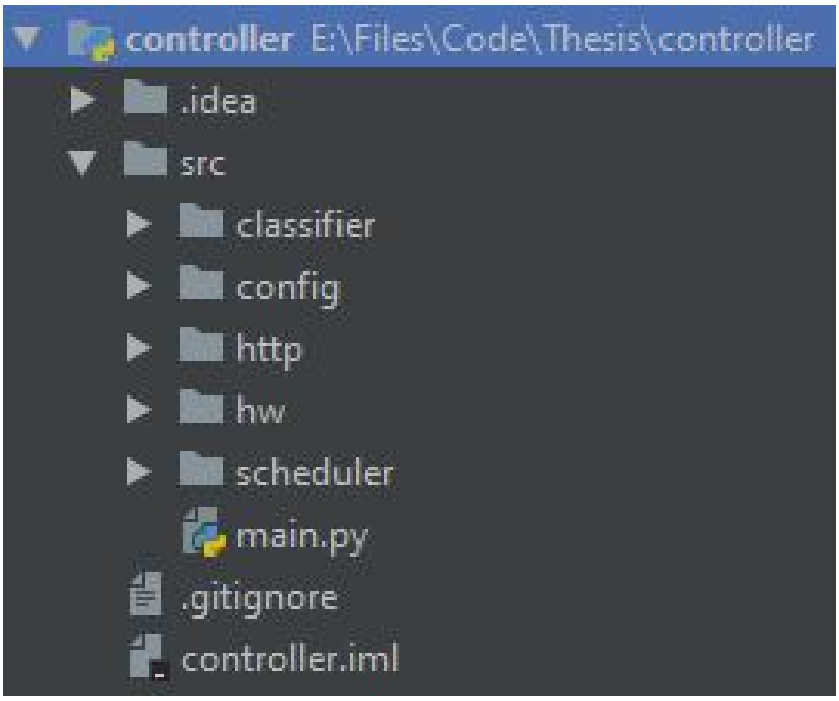

Figure 5.5: Controller Structure

The first two are used to make data manipulation easier by providing an interface for the system's state and last one is where the main scheduling algorithm resides. These python scripts are discussed in detail as follow.

i *assigned_state.py*
This python script mainly contains a class that hold data that is acquired after calculation is performed by the scheduler, hence its name. Its class diagram is given on figure 25 below.

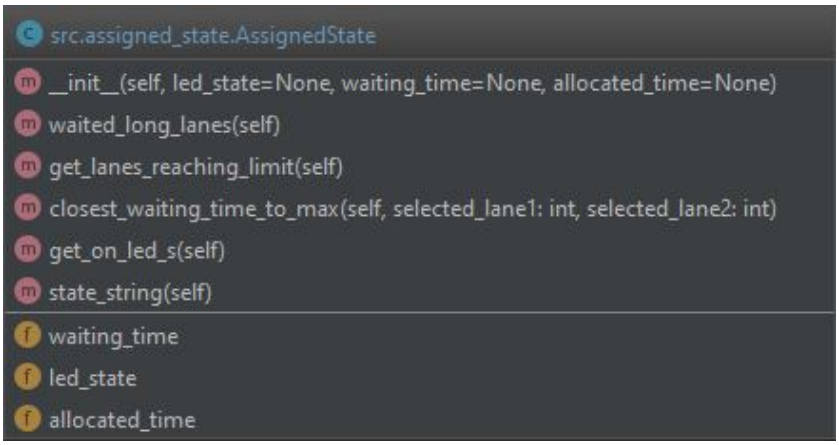

Figure 5.6: AssignedState Class Diagram

From figure 5.6, we can see that AssignedState has three fields. The first one, waiting_time is a list of eight elements. They represent each of the lanes controlled by the traffic light. All the elements are integers that represent the

amount of time a lane has waited to get a green traffic light. That is the time span from the last time the lane got a green light up to now. These values are reset to zero every time a lane gets a green light or if a lane has no car during calculation by the scheduler.

The second field, led_state, is also a list of eight elements. Its elements have Boolean values that represent the current state, "on" or "off", of each traffic light. At a time, only two of the eight values can have a value of True since only two lanes can cross the crossroad at the same time.

The third field is allocated_time. It is an integer that represents the amount of time the lanes that got the "on" state in led_state field stay one. All the three fields are listed above are the result of calculation by the scheduler and that is why they are placed in a single class.

This class also has functions that retrieve information based on the data held by the above three fields. Including the constructor, these functions are 6 in number. The purpose of these functions is as follows.

- *Constructor* : takes in three optional parameters for the three fields. It assigns a default value of "[False] *8", "[0] *8" and "0" if no value is passed.
- *waited_long_lanes* : retrieves list of indices of lanes that have waited long. That its lanes that waited longer than the maximum waiting time minus 15 seconds.
- *get_lanes_reaching_limit* : retrieves list of lane indices that will reach the maximum waiting time if the maximum assignable time is allocated now.
- *closest_waiting_time_to_max* : retrieves the closest waiting time to the maximum waiting time without considering the lanes whose index have been passed as an argument.
- *get_on_led_s* : retrieves list of lane indices that are green.
- *state_string* : returns the state in a formatted string. This function is used for testing purpose.

In addition to the AssignedState class, this file also contains two variables. These are maximum_waiting_time and maximum_assignable_time. In this implementation, they are assigned values of 120 and 40 respectively.

ii *density_state.py*

Just like the assigned_state.py file, this python script also has a class and external variable. The class is meant to hold data coming from camera, that is the number of cars for this implementation. Its class diagram is given on figure 5.7 below.

This class only has one field, car_density. It is a list of integers that represent the car densities on the eight lanes. Just like the previous class, this class also

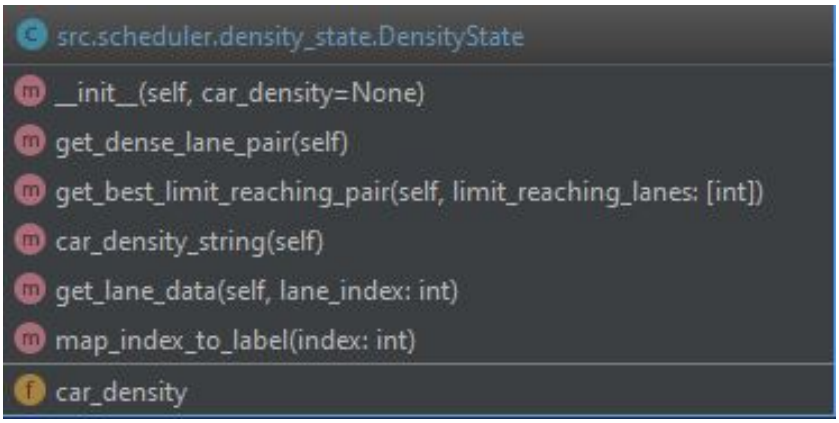

Figure 5.7: DensityState Class Diagram

has functions that retrieve information based on the data held by the field car_density. Including the constructor, these functions are 5 in number. The purpose of these functions is as follows.

- *Constructor* : takes in one optional parameter for the car_density fields. It assigns a default value of "[ ]" if no value is passed.
- *get_dense_lane_pair* : retrieves pair of lane indices as a list. This pair represent the two lanes that has the highest number of cars and can cross together.
- *- get_best_limit_reaching_pair* : retrieves pair of lane indices that are going to reach the maximum waiting time if the maximum assignable time is allocated and has the highest car density as a pair.
- *car_density_string* : returns the state in a formatted string. This function is used for testing purpose.
- *get_lane_data* : accepts lane index that ranges from 0 to 7 and returns a LaneData object. This class is in the http_client file and will be discussed in detail a section about the http module.
- *map_index_to_label* : this is a static function that returns a string based on the lane index. This string is the side later in lowercase concatenated with the lane number. For example, "b1".

This class can be used to hold other density related data like the accepting capacity of opposite roads. This is one of the features to be considered in the future. Like the first script, this script also has a variable besides the class. The variable is possible_lane_pairs and it contains a list of pair of lanes that can cross the crossroad simultaneously. This pairs are discussed in section 4.2.2.

iii *scheduler.py*

This is the core file where the scheduling algorithm reside along with different helper functions. The class diagram and description of the functions are presented below.

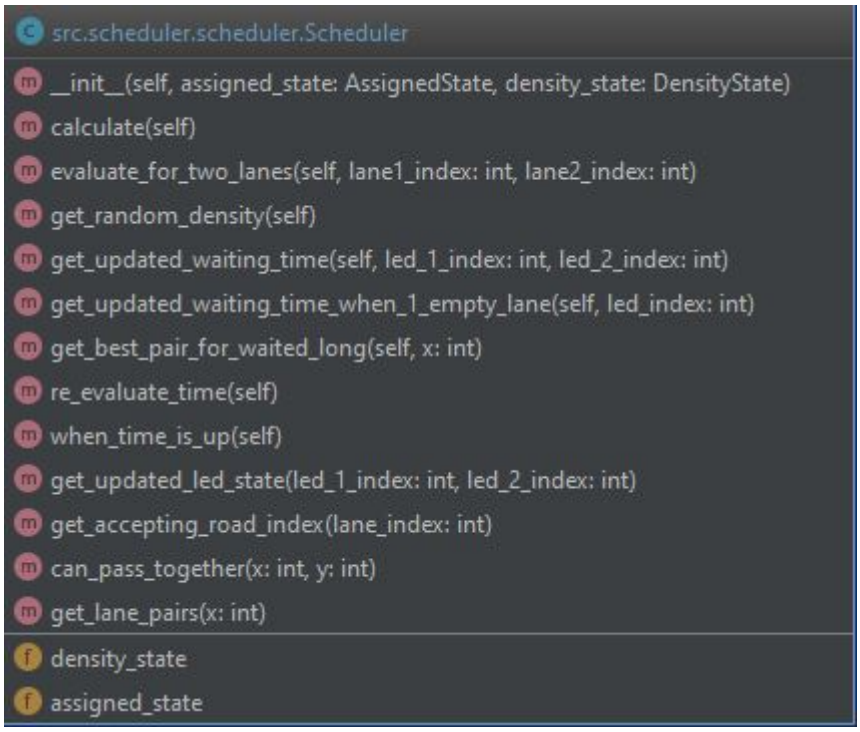

Figure 5.8: Scheduler Class Diagram

The first two functions are where the scheduling is performed. The remaining functions are helps to the two main functions, let us look into these functions.

- *get_updated_waiting_time*: this function accepts two arguments and returns a list of the lanes waiting time. It makes the waiting time of the two lanes zero. For the remaining lanes, it adds the allocated time to their previous waiting time given that there is at least one car in that lane.
- *get_updated_led_state*: this function accepts two arguments and returns a list of Boolean values. It assigns "True" for the indices passed in as an argument and "False" for the rest.
- *get_updated_waiting_time_when_1_empty_lane*: this function is like the first one. The difference is that it only accepts one lane index.
- *can_pass_together*: this function takes two integers, lane indices, as arguments and returns a Boolean. As the name indicates, it checks if the two lanes can cross together.
- *get_best_pair_for_waited_long*: this function accepts a single argument and returns an integer, lane index. It chooses from the possible pairs of the lanes, passed in as the its argument. It does so by comparing their vehicle density and waiting time. It gives priority for waiting time.
- *get_lane_pairs*: this function accepts the index of a lane and returns list of lane indices that can possible be paired with it.

- *get_random_density*: this is a function that return a randomized list of car densities based of the current AssignedState and DensityState stored on the object.

The scheduling algorithm is implemented using two functions. The second function is used to avoid redundant code.

- *calculate*: this function makes major decisions of the algorithm by accessing the AssignedState and DensityState objects stored on the scheduler instance. Once it narrows the decision down to two lanes, it invokes "evaluate_for_two_lanes" function with the selected indices. The invoked method will decide how much time to allocate to the selected lanes. Since the flow chart is too bulky to fit in a single chart, we have splitted it into two. Figure 5.14 and figure 5.15 gives the flow charts.
- *evaluate_for_two_lanes*: this function accepts two arguments, index of the two lanes, and returns an instance of AssignedState. It flows chart is given on figure 5.13.

5.1.3.2 *hw*

This module is where all file related to hardware access and manipulation resides. In our implementation it has one file named hardware_manipulator.py. As its name indicates, this file is responsible for the controlling of hardware parts. It contains variables for each LED, list of green LEDs, list of Yellow LEDs, list of Red LEDs and two functions that operate on the lists. The functions are described below.

- *turn_greens_on*: this function takes a list of lane indices. It turns the red lights of these lanes off and turns their green lights on. For lanes not included in the argument, it turns the red lights on after turning their yellow light off.
- *turn_yellows_on*: just like the previous function, this function also takes a list on lane indices. It then turns the green lights of these lanes off and turns their yellow lights on.

5.1.3.3 *http*

This is the module that has scripts that are responsible for API communication. Any script that is related to accessing or manipulating data on a remote server is to be placed in this module, but for now it has just one python script called loopback_client.py that is used to communicate with the API we made.

It has two classes:

- LoopbackClient
- LaneData
- *LoopbackClient*
  This class is used to send data about the lanes to the API. Its class diagram looks like the following:

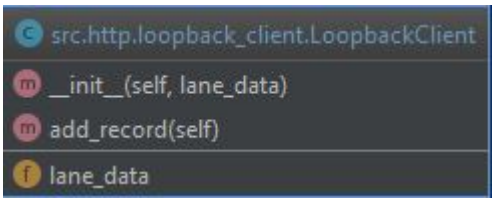

Figure 5.9: LoopbackClient Class Diagram

It has a single attribute that is of the LaneData Datatype. It takes this object and make an HTTP request to the Loopback API using the add_record() function.

- *LaneData*
  A data structure for holding data about vehicle count and the corresponding side and lane number. Its class diagram is given below.

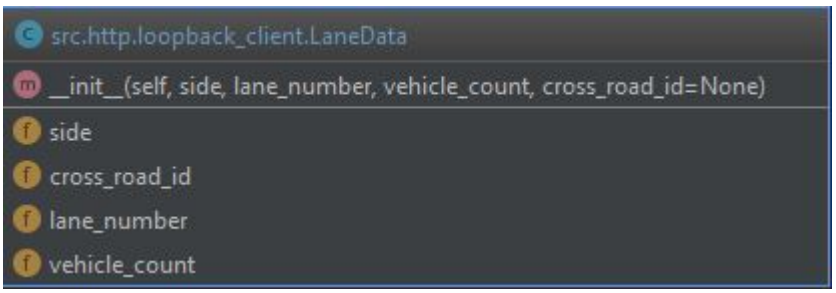

Figure 5.10: LaneData Class Diagram

In addition to the three attributes mentioned, it has an attribute "cross_road_id" that is used to identify the crossroad from which the data is coming from.

5.1.3.4 *classifier*

In this package the classifier or the ".xml" file and Classifier class are found. The program in the Classifier class also creates some temporary files while it performs vehicle detection and counting. The class diagram for the classifier class is given on figure 5.13 below.

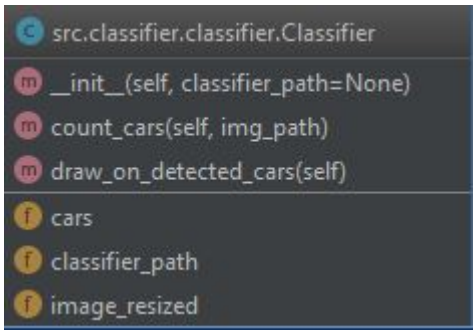

Figure 5.11: Classifier Class Diagram

As you can see from the diagram above, it has three attributes. The main attribute "classifier_path", is the route to the "ẋml" file. The two functions it has are described below.

- *count_cars*: provided with the route to the image, it runs vehicle detection on the picture and return the number of cars it counted. In the process it assigns values to the remaining two attributes "cars" and "image_resized".
- *draw_on_detected_cars*: using the two attributes saved by the above method, it draws a square box on each of the car the previous method detected. This method is merely used as a visual aid for testing and demonstration.

All these modules come together in the main.py file. This python script has a sing method called "main()" which is registered as the main function and serves as the entry point of the whole program. Figure 5.12 on section 5.1.4 illustrates how it does this.

5.1.4 *Implemented system flow charts*

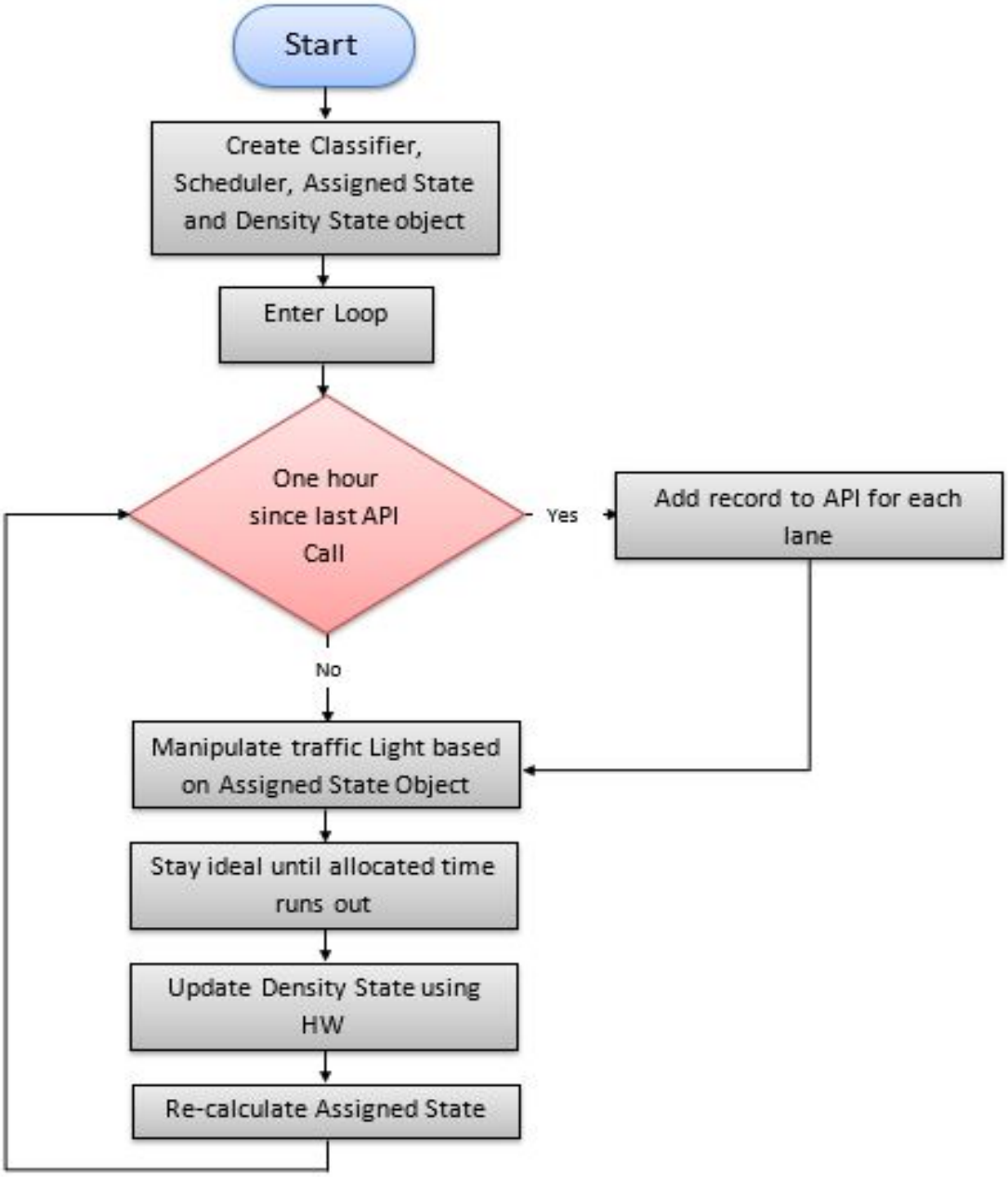

Figure 5.12: Main Function Flow Chart

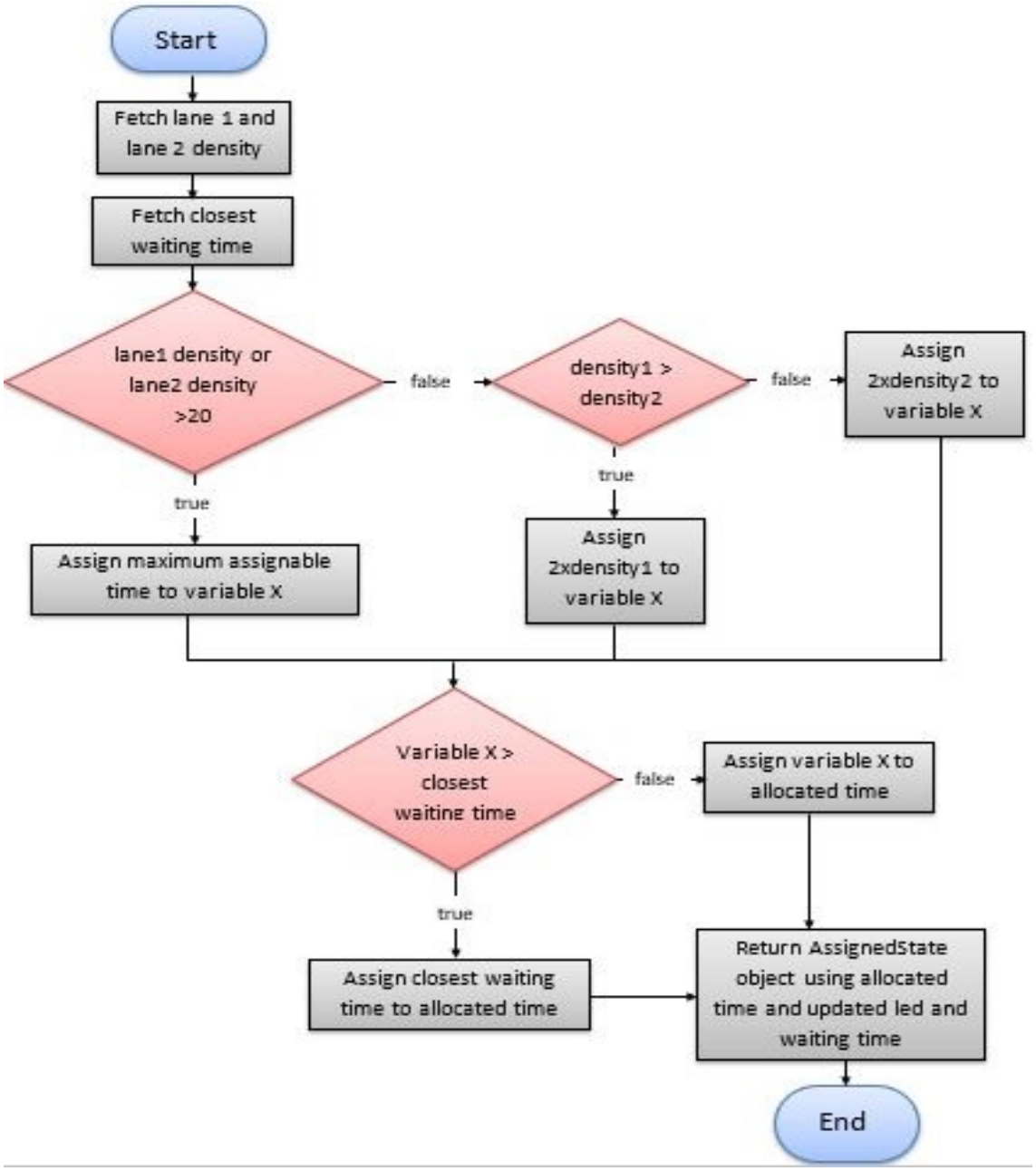

Figure 5.13: evaluate_for_two_lanes flow chart

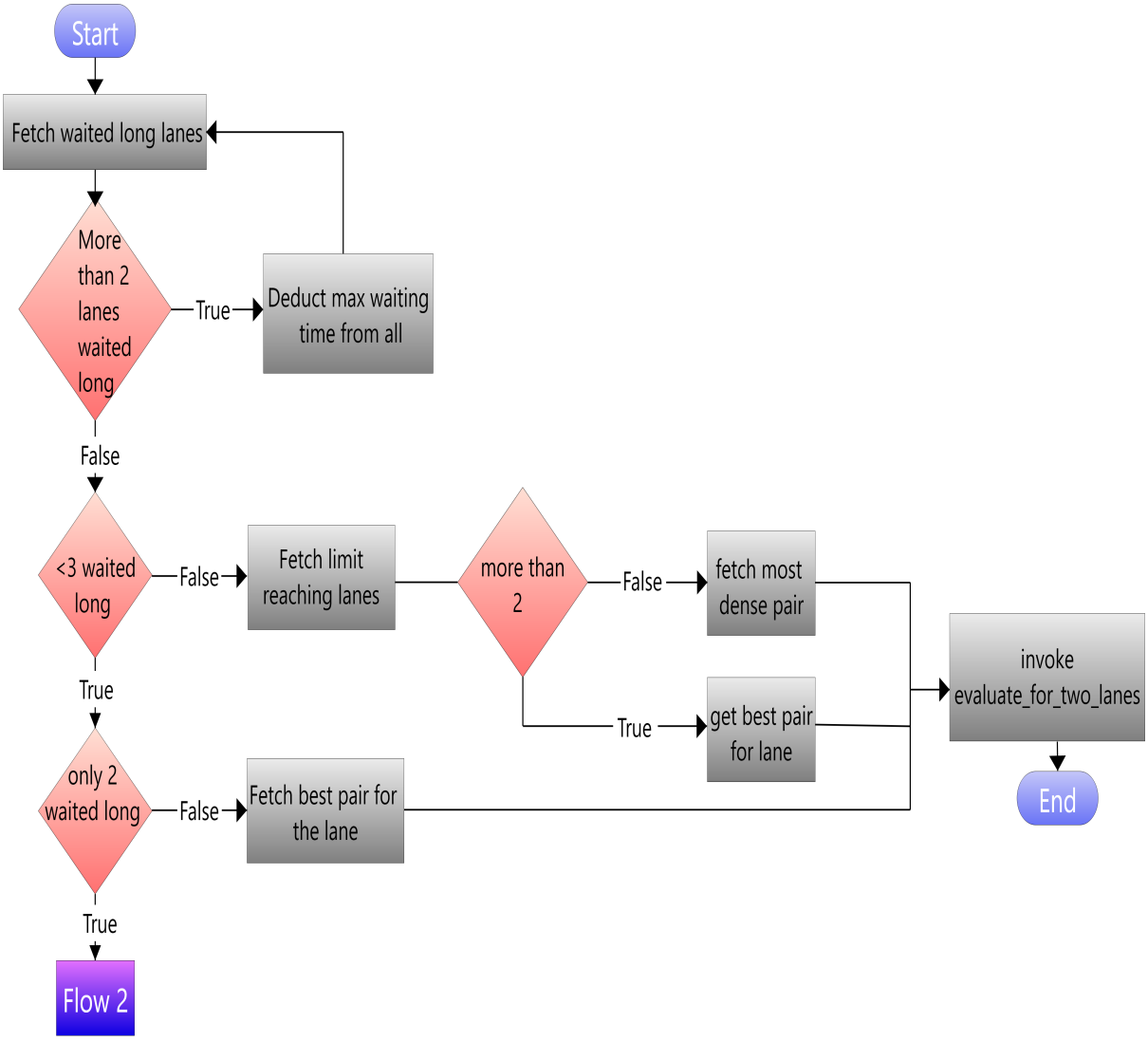

Figure 5.14: calculate flow chart 1

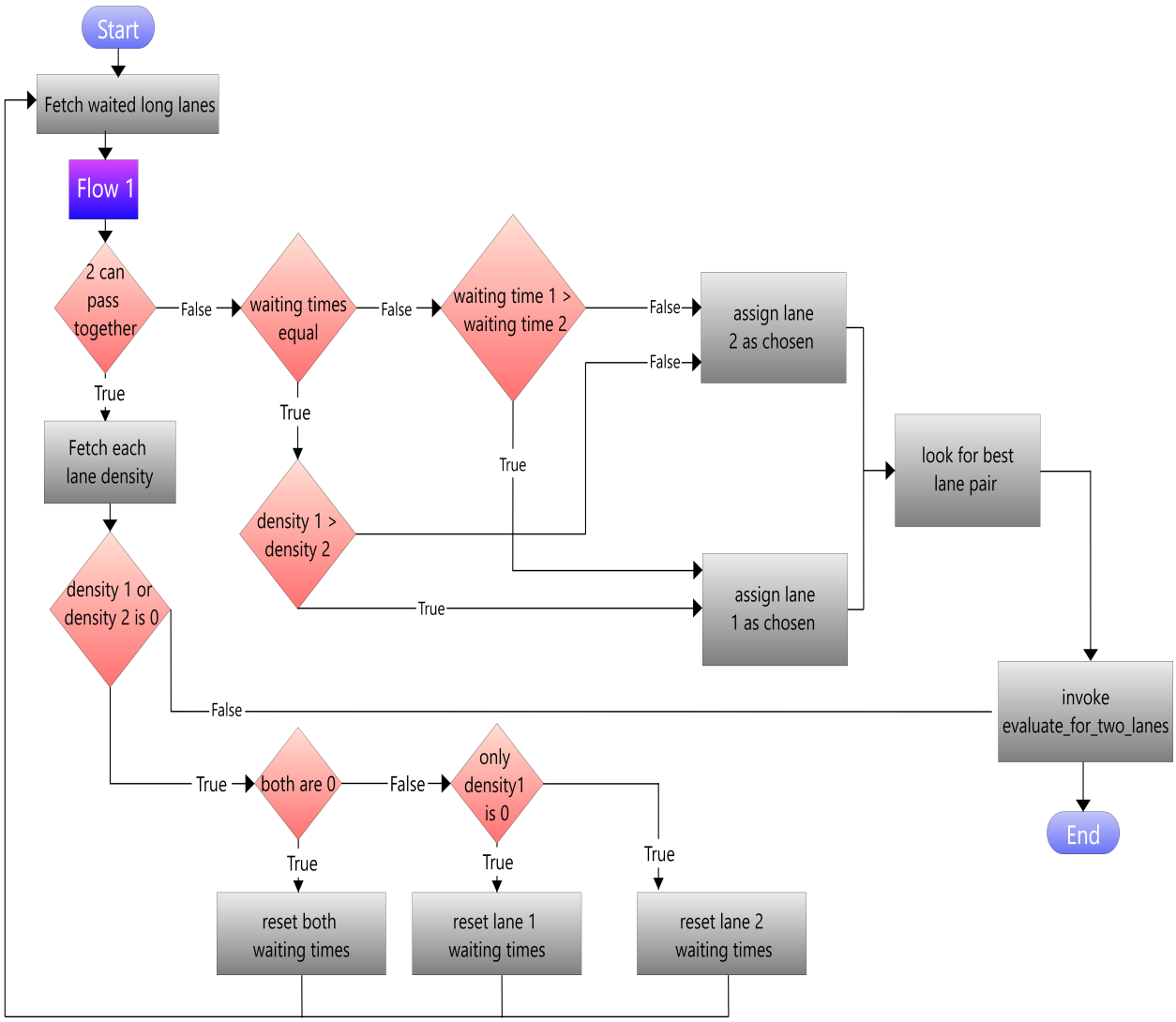

Figure 5.15: calculate flow chart 2

5.2 HARDWARE IMPLEMENTATION

Implementing the hardware takes interfacing the micro-controller with the LEDs and the GSM module. Instead of directly connecting the resistors and LEDs to a breadboard, we first built a prototype of a crossroad. To do this we first had to solder three LEDs and their respective resistors. This is given on figure 5.15 below.

We then placed it in a box, and it looks like the one given on figure 5.16. This box is meant to simulate a single traffic light. Since two traffic lights are required for each side of the crossroad, we made 8 of such boxes that simulate a traffic light. For half of these traffic lights, we used similar pattern of colored wires. The colors used were, green for green LED, orange for yellow LED, blue for red LED and brown for ground wire. Note that the cathodes of these LEDs have been shorted. For the remaining half, we used stripes of the colors used for the first half of traffic light boxes.

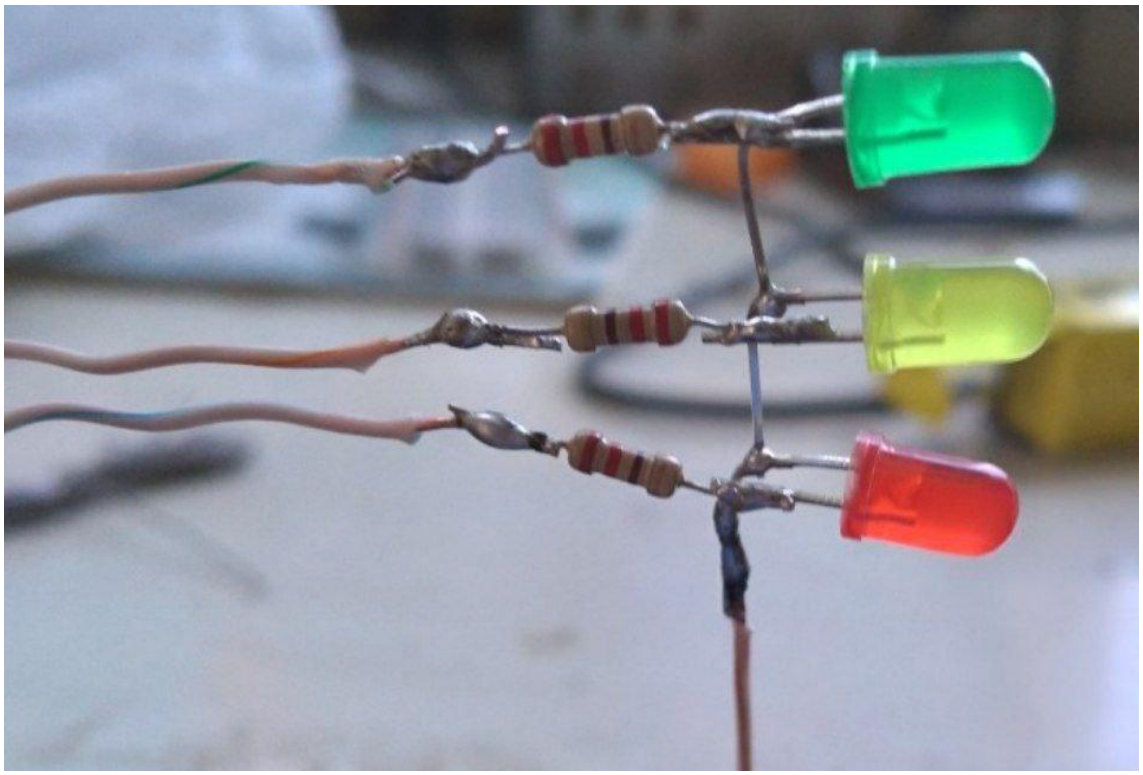

Figure 5.16: Soldered LEDs

After building these traffic lights. We proceeded to making a prototype of the crossroad. For this we had used a picture of the crossroad made using photoshop and a carton like the one we used for the traffic light prototypes. Here all it takes it sticking a paper print of the crossroad picture on the carton. The picture is given below.

We finished the prototype by fixing the eight traffic lights on the crossroad prototype we made using the above picture. This prototype is then interfaced with micro-controller. Additionally, the micro-controller is also interfaced with the GSM module.

The GSM module uses UART for communication. So, we connect Tx of the GSM module to the Rx (pin 10) of the raspberry pi and Rx of the GSM module to the Tx (pin 8) of the raspberry pi. The configuration of GSM SIM800L is given on figure 3.6 and the raspberry pi pin configuration is given on figure 3.10 above.

For this project, we are only interested in NET, Vcc, RXD, TXD and GND. NET is where the Helical Antenna is connected. All the connections on the GSM module are soldered.

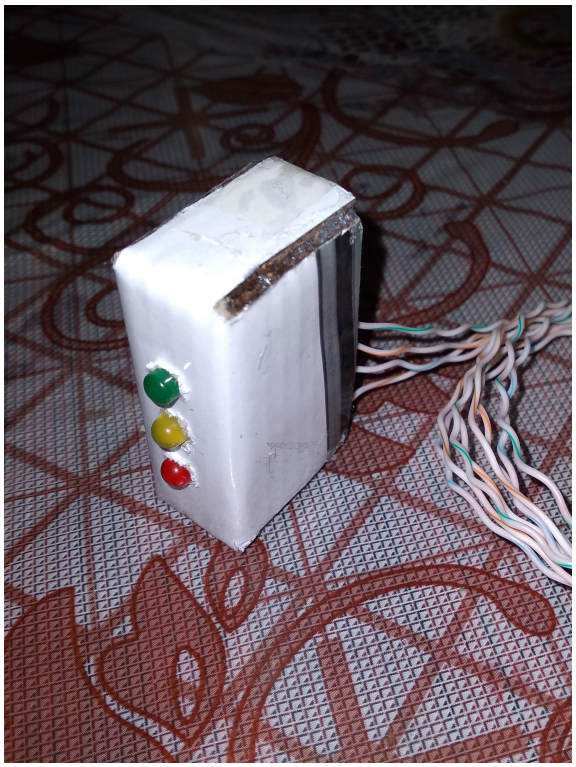

Figure 5.17: Traffic Light Prototype

Figure 5.18: Photoshopped Crossroad

6

RESULT

The cascade classifier was able to detect cars from images we took. However it needs a clear picture and for there to be a little space between the cars. The classifier also does not respond well when there is a shade or no bright light on the object to be detected.Figure 6.1 to Figure 6.5 shows the classifier detecting cars from pictures we around 4kilo roundabout.

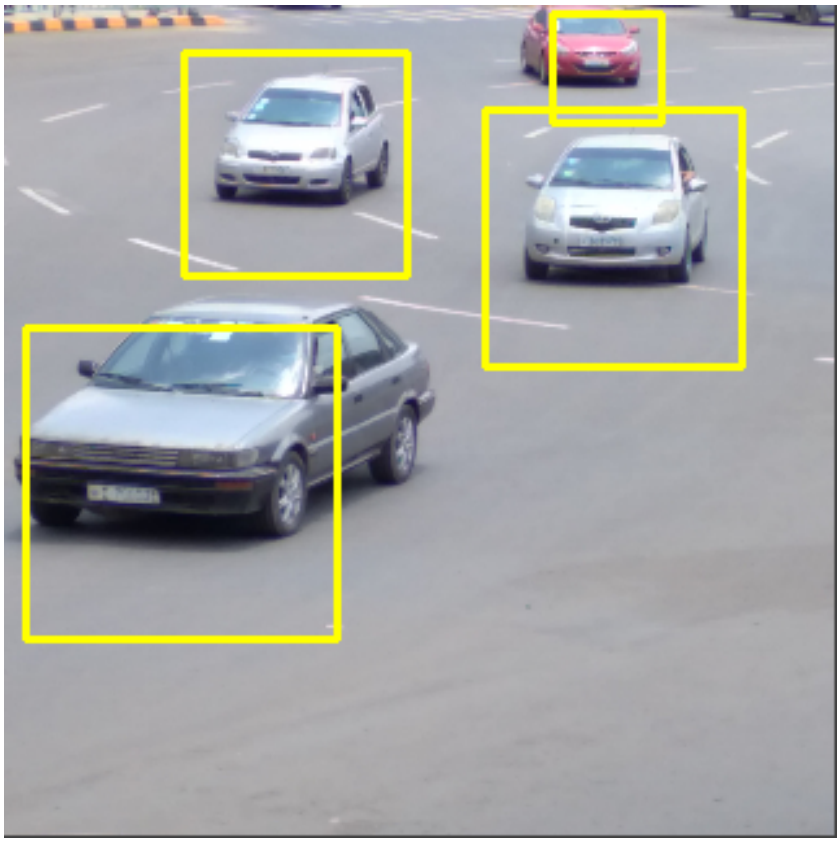

Figure 6.1: Detection 1

As we can see the pictures given on figure 6.1 through figure 6.5 result pictures given above, the classifier is able to detect vehicles in a picture effectively given that the vehicles are spaced a bit apart. Once the number of cars is acquired from the count_cars function in the Classifier, the Scheduler can decide which lanes to allow through with the amount of time to allocate for them. Based on its the schedulers calculation, the main function will control the traffic lights using the hardware_manipulator.Figure 6.6 and Figure 6.7 illustrates the prototype controlled by the program.

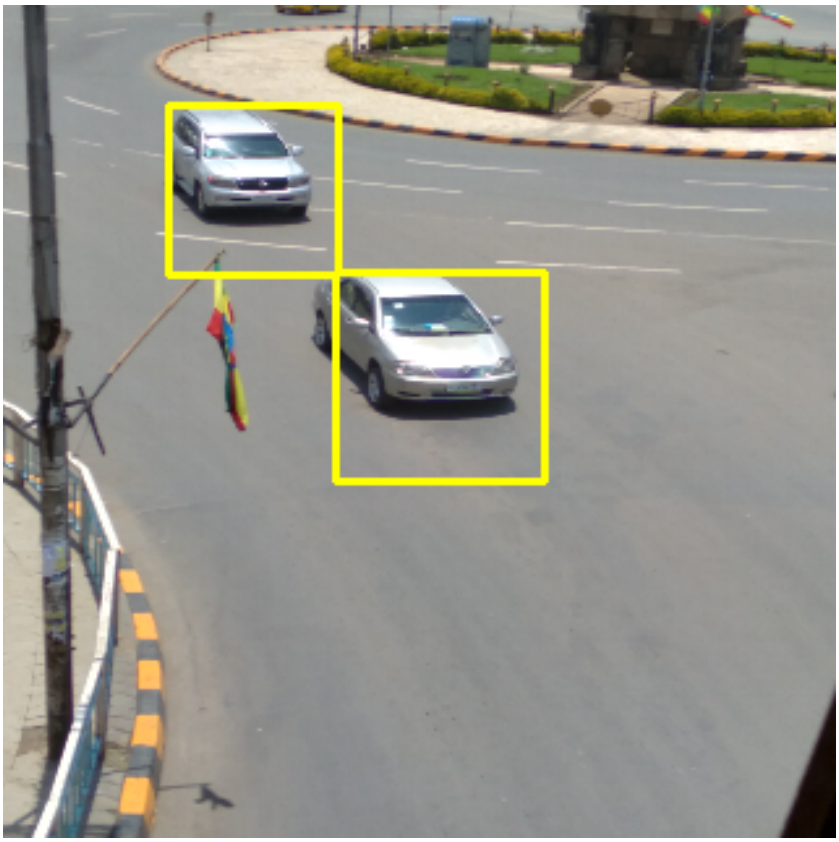

Figure 6.2: Detection 2

As indicated on figure 5.14, the first thing the main function does is to check if it has been an hour since the last API call and do so if it has been an hour. It sends the density data it gets from the classifier to the API. This data can be preview in a simple web page we implemented. Figure 6.8 illustrates the web page.

We can see from Figure 6.8 that all the necessary data that corresponds to the vehicle density is given on the table. The information at the top of the table is the details of the crossroad, which is stored in the cross_road collection, refer section 4.2.2. The table contains data recorded for the selected crossroad. The collapsible on the left is meant to allow switching between crossroads. At this point it is not functional.

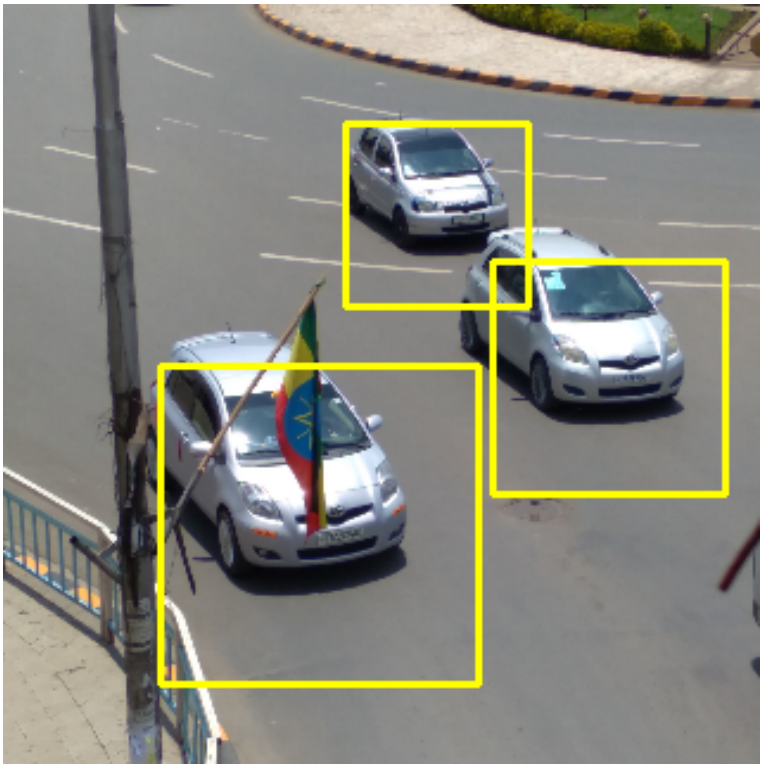

Figure 6.3: Detection 3

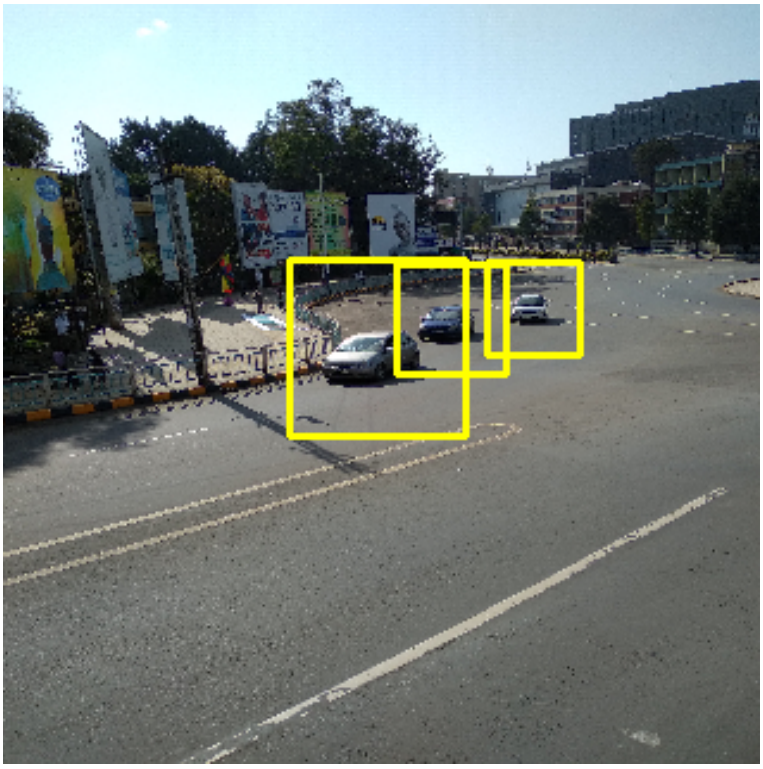

Figure 6.4: Detection 4

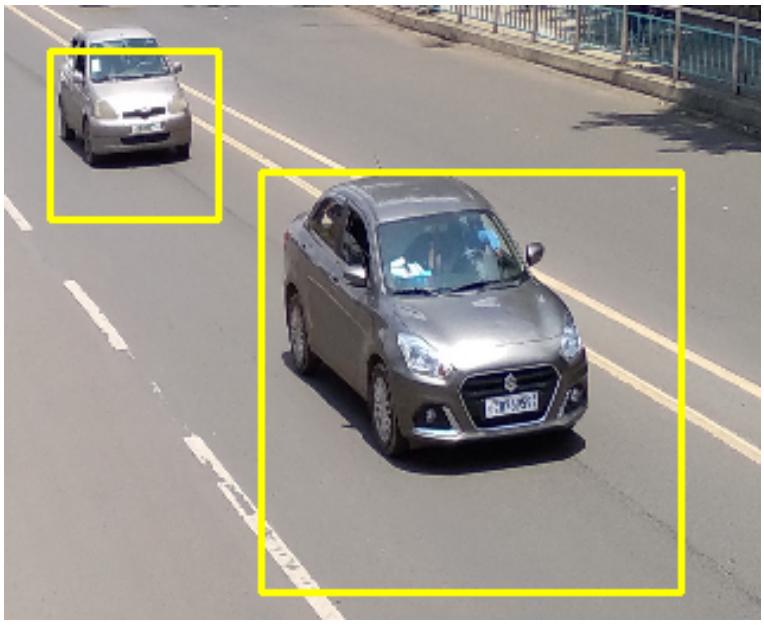

Figure 6.5: Detection 5

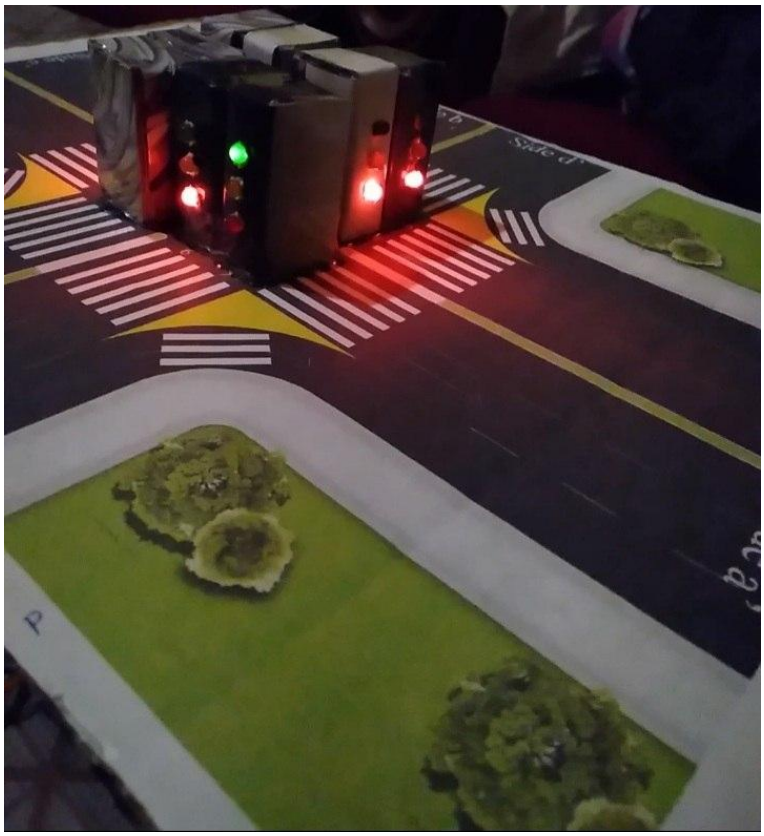

Figure 6.6: Controlling Prototype 1

Figure 6.7: Controlling Prototype 2

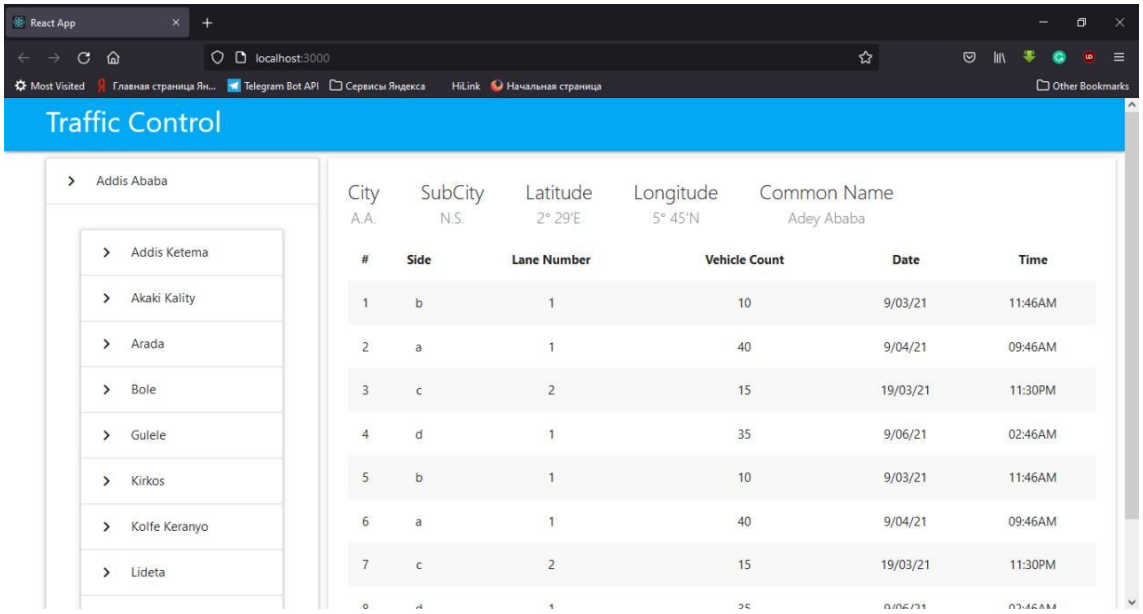

Figure 6.8: Traffic Control Website

7

CONCLUSION AND DISCUSSION

Density based traffic control was found to overcome the limitation of today's method of traffic signaling method. Today's traffic signaling method allocated equal time for all roads of the intersection. This would mean roads with low density of traffic or even no traffic at all will still get an equal green light time to roads with a high density of traffic.

The system we built solves this problem by first detecting the number of cars or the density of traffic on each road and allocating an appropriate green light time for them. Since roads with a higher density of traffic get more green light time, they will be able to relieve the traffic on their side early before more buildup of traffic. This way the different roads of an intersection will not become congested with traffic.

FUTURE WORK

- Make the car cascade classifier learn from the data coming from cameras. At present the car cascade classifier was trained only once. This can become insufficient for real world scenarios. By making the cascade classifier learn from live data coming from cameras, it will become more able at detecting cars.
- Add a way to accept interrupt. This will help in scenarios when emergency vehicles want to pass. When the interrupt is generated, the system will give priority to the emergency vehicle and turn the light green for the lane the emergency vehicle is on.
- Add a way of estimating the amount of space left across the road so the road they go to doesn't become congested. This will be some kind of distance measuring algorithm.
- Make the scheduling algorithms better by adding a way to accommodate the distance measuring algorithm discussed above.
- Adding support for multiple intersections and adding feature on the website to display the different intersections and navigate between them.
- Make the computation of detecting cars and scheduling on a remote server. This will save processing power and money that would have been spent on installation of high performance processors everywhere.

BIBLIOGRAPHY

[1] Gaikwad et al. "Image Processing Based Traffic Light Control." In: (Mar. 2014).

[2] A. A. SHAFIE MD.Hazrat ALI Syuhei KUROKAWA. "Autonomous Road Surveillance System: A Proposed Model for Vehicle Detection and Traffic Signal Control." In: (Mar. 2013).

[3] Andrew Selasi Agbemenu Benjamin Kommey Seth Djanie Kotey. "Autonomous Vehicle Density-based Traffic Control System." In: (Mar. 2014).

[4] *Raspberry Pi*. https://en.wikipedia.org/wiki/Raspberry_Pi. 2021.

[5] *What is computer vision*. https://www.ibm.com/topics/computer-vision. 2021.

[6] Jake Frankenfield. *Machine Learning*. https://www.investopedia.com/terms/m/machine-learning.asp. 2021.

[7] Sumit Saha. *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way*. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53. 2021.

[8] *About*. https://opencv.org/about/. 2021.

[9] Ben Mauss. *Haar-like Features: Seeing in Black and White*. https://levelup.gitconnected.com/haar-like-features-seeing-in-black-and-white-1a240caaf1e3. 2021.

[10] Ben Mauss. *The Integral Image*. https://levelup.gitconnected.com/the-integral-image-4df3df5dce35. 2021.

[11] *Turning on an LED with your Raspberry Pi's GPIO Pins*. https://thepihut.com/blogs/raspberry-pi-tutorials/27968772-turning-on-an-led-with-your-raspberry-pis-gpio-pins. 2021.

[12] *Raspberry Pi Documentation*. https://www.raspberrypi.org/documentation/. 2021.

[13] Amin Ahmadi. *Cascade Trainer GUI*. https://amin-ahmadi.com/cascade-trainer-gui/. 2021.