



ASHESI

ASHESI UNIVERSITY

**LOW-COST SMART TRAFFIC MANAGEMENT
SYSTEM**

CAPSTONE PROJECT
B.Sc. Computer Engineering

Jude Asare Donkor
2020

ASHESI UNIVERSITY

**LOW-COST SMART TRAFFIC MANAGEMENT
SYSTEM**

CAPSTONE PROJECT

Capstone Project submitted to the Department of Engineering, Ashesi University in
partial fulfilment of the requirements for the award of Bachelor of Science degree
in Computer Engineering.


Jude Asare Donkor

2020

Declaration

I hereby declare that this capstone is the result of my own original work and that no part of it has been presented for another degree in this university or elsewhere.

Candidate's Signature:


.....

Candidate's Name:

Jude Asare Donkor
.....

Date: **29/05/2020**
.....

I hereby declare that preparation and presentation of this capstone were supervised in accordance with the guidelines on supervision of capstone laid down by Ashesi University.

Supervisor's Signature:

.....

Supervisor's Name:

.....

Date:

Acknowledgement

I am grateful to the Almighty God for giving me strength, astuteness and grace throughout this project. My big thanks to my family for encouraging me always to keep moving and their prayers.

I would like to express my gratitude to my capstone supervisor, Kofi Adu Labi, for his encouragement, as well as professional and academic advice which helped me successfully undertake this project while gaining a lot of new knowledge. I am deeply grateful.

Special thanks to Nana Akua Sereboo, for her constant encouragement, help and guidance.

My sincere gratitude goes to the engineering faculty and staff members who have been ready to assist throughout the project period.

Finally, I would also like to acknowledge friends, who have continuously encouraged me to be a better version of myself throughout my four years of study, and especially as I undertook this project

.

Abstract

In Ghana and other parts of Africa, most traffic management systems are implemented by the use of a timer at each phase. This method of traffic management is inefficient because equal length of green light is assigned to each lane at the intersection; resulting in long wait times for vehicles behind a traffic light with the red signal on, especially when other lanes are vacant. This project addresses problems such as the one stated in the previous sentence through designs and implementation of a low-cost smart traffic management system. The project is carried out by the use of electrical components that are affordable, easy to maintain and reliable. An inductive loop is used as the vehicle detection device for each lane. Traffic density acquisition is then done using an algorithm running on a single board computer, Raspberry Pi, to obtain the number of vehicles on each lane. After the acquisition of traffic density, the data is processed to obtain the green light length for a lane. After green light length determination, the traffic density acquired earlier is sent to a database on a server via Wi-Fi to enable another microcontroller to use the data to execute traffic coordination between a lane and its preceding lane. In testing the coordinated control system, the statistical analysis showed that the coordinated traffic control between a lane and its preceding lane can be improved to gain the optimal performance.

Table of Contents

<u>DECLARATION.....</u>	<u>I</u>
<u>ACKNOWLEDGEMENT.....</u>	<u>II</u>
<u>ABSTRACT</u>	<u>III</u>
<u>TABLE OF CONTENTS.....</u>	<u>IV</u>
<u>LIST OF TABLES</u>	<u>VII</u>
<u>LIST OF FIGURES</u>	<u>VIII</u>
<u>CHAPTER 1: INTRODUCTION.....</u>	<u>1</u>
<u>1.1 BACKGROUND:</u>	<u>1</u>
<u>1.2 PROBLEM STATEMENT:</u>	<u>1</u>
<u>1.3 MOTIVATION:</u>	<u>2</u>
<u>1.4 ATTEMPTS TO SOLVE THIS PROBLEM:</u>	<u>2</u>
<u>1.5 SOLUTION:</u>	<u>3</u>
<u>CHAPTER 2: LITERATURE REVIEW.....</u>	<u>4</u>
<u>2.1 IMAGE PROCESSING:.....</u>	<u>4</u>
<u>2.2 VEHICLE DETECTION (SENSORS):</u>	<u>5</u>
<u>CHAPTER 3: DESIGN.....</u>	<u>6</u>
<u>3.1 DESIGN DECISIONS AND PUGH MATRICES.....</u>	<u>6</u>
<u>3.1.1VEHICLE DETECTION COMPONENT</u>	<u>6</u>
<u>3.1.2 PUGH CHART FOR PROCESSING UNIT</u>	<u>8</u>

<u>3.2 DESIGN BLOCK DIAGRAM AND CIRCUITRY:.....</u>	<u>8</u>
<u>3.3 DESIGN ITERATION FOR TRAFFIC MANAGEMENT SYSTEM:</u>	<u>11</u>
<u>3.4 DESIGN DECISION FOR TRACKING SYSTEM:.....</u>	<u>13</u>
<u>3.5 SYSTEM ARCHITECTURE:</u>	<u>14</u>
<u>3.6 DESCRIPTION OF COMPONENTS:.....</u>	<u>17</u>
<u>CHAPTER 4: METHODOLOGY</u>	<u>19</u>
<u>4.1 HARDWARE IMPLEMENTATION</u>	<u>19</u>
<u>4.1.1 VEHICLE DETECTION</u>	<u>19</u>
<u>4.1.2 ACQUISITION OF TRAFFIC DENSITY</u>	<u>20</u>
<u>4.1.3 GREEN LIGHT SEQUENCING AND LENGTH DETERMINATION.....</u>	<u>20</u>
<u>4.1.4 PEDESTRIANS INTERRUPT</u>	<u>20</u>
<u>4.2 SOFTWARE IMPLEMENTATION.....</u>	<u>21</u>
<u>CHAPTER 5: TEST AND RESULTS.....</u>	<u>24</u>
<u>5.1 VEHICLE DETECTION</u>	<u>24</u>
<u>5.2 ACQUISITION OF TRAFFIC DENSITY, GREEN LIGHT SEQUENCING AND LENGTH DETERMINATION.....</u>	<u>25</u>
<u>5.3 WEBAPP.....</u>	<u>29</u>
<u>5.4 STATISTICAL TEST OF RESPONSE TIME OF COORDINATED TRAFFIC CONTROL BETWEEN A LANE AND ITS PRECEDING LANE.</u>	<u>32</u>
<u>5.5 TEST AND ANALYSIS OF WAIT TIME OF THE SMART SYSTEM AND OF A TRADITIONAL TRAFFIC LIGHT</u>	<u>33</u>
<u>CHAPTER 6: CONCLUSION, LIMITATIONS AND FUTURE WORK</u>	<u>36</u>
<u>6.1 CONCLUSION.....</u>	<u>36</u>

<u>6.2 LIMITATIONS</u>	<u>37</u>
<u>6.3 FUTURE WORK</u>	<u>37</u>
<u>REFERENCES</u>	<u>38</u>
<u>APPENDIX</u>	<u>40</u>
<u>APPENDIX A: RECORDED WAIT TIMES</u>	<u>40</u>
<u>APPENDIX B: ALGORITHM FOR ENTIRE SYSTEM</u>	<u>41</u>
<u>APPENDIX C: PRIORITY VEHICLE CODE NODE MCU</u>	<u>48</u>

List of Tables

Table 3. 1: Pugh matrix for best vehicle detection technique	7
Table 3. 2: Pugh matrix to choose the best processing unit	8
Table 3. 3: Component Description	17
Table 5. 1: Results from test	26
Table 5. 2: Results from Algorithm	27
Table 5. 3: Results from the Green Light Length Determination Algorithm.....	28

List of Figures

Figure 3. 1: Block diagram for inductive loop system.....	9
Figure 3. 2: Circuit for inductive loop system	10
Figure 3. 3: Ultrasonic system block diagram	11
Figure 3. 4: Circuit for the ultrasonic system.....	12
Figure 3. 5: Tracking System Block Diagram	13
Figure 3. 6: Tracking System Circuit.....	14
Figure 3. 7: System Architecture	16
Figure 3. 8: System Architecture of Tracking System.....	17
Figure 4. 1: Positioning of the sensor	19
Figure 4. 2: Pedestrian button activated.....	21
Figure 4. 3: Web application interface.....	22
Figure 4. 4: Database Relationship Diagram	23
Figure 5. 1: Diagram of lanes being used	24
Figure 5. 2: Test Setup	24
Figure 5. 3: Vehicle Detection Test	25
Figure 5. 4: Traffic Density Acquisition.....	26
Figure 5. 5: Diagram of Lanes being used	28
Figure 5. 6: Login Page Interface.....	30
Figure 5. 7: Dashboard Interface.....	31
Figure 5. 8: T test results.....	32
Figure 5. 9: case study area (Okponglo, Legon intersection)	33
Figure 5. 10: Diagram of lane (Okponglo Legon intersection) being used.	34

Chapter 1: Introduction

1.1 Background:

The growing number of vehicles, extensive number of traffic on roads and improper methods of traffic control has created and accounted for urban traffic congestion. Bad traffic management and traffic jams interfere with drivers, lead to wastage of time, cost money per year and also lead to an increase in the consumption of fuel and increase in the emission of harmful gases to the environment. The existing method used for traffic control in Ghana is the use of timers for each phase. Although this primitive method has been used many times to control traffic, it fails to reduce traffic congestion as the number of cars increase. As this problem spreads, there is a need for an advanced traffic management system to improve the existing traffic control methods. Therefore, for better traffic management, this work proposes a system for controlling the traffic light sequence and timing by the use of an inductive loop vehicle detection system. The use of sensors helps in proper traffic management by detecting vehicles from interactions with the physical environment.

1.2 Problem statement:

Traffic congestion is a growing problem which has led to the waste of resources such as time and money, and hassle of transportation. A major contribution to this growing problem occurs at intersections. Using the Accra airport intersection (liberation road) as an instance, all the traffic lights located on the intersection lanes have fixed timings where the duration of the green light is constant through all the traffic lights. This method is inept when there are multiple cars on several lanes of the intersection; it causes long waiting times when some lanes have few vehicles or no vehicle moving through them. In such situations, time is wasted since there will be a long waiting time for a vehicle populated lane's traffic light to turn green. Additionally, the emergency response system in Ghana does not have a rapid response as all emergency

systems do in the world. In Ghana, patients who are being transported from homes or from accident scenes to the hospital spend hours in an ambulance on end up maneuvering their way through heavily congested traffic on the route to get to the hospital in time for treatment.

1.3 Motivation:

I was motivated to do this project to help travelers and drivers save time by reducing the time they spend on roads because of traffic congestion and bad traffic management. After being victimized by this problem, my empathy for other victims of this problem made me realize the need for an effective traffic control system. An effective traffic control system will increase productivity and decrease the time and money spent on transportation since their estimated time of arrival will be the same as the travelers or drivers anticipated after a sprout of traffic. Also, cars will not have to wait for an unoccupied lane's traffic light to turn red before it can move through its lane. In addition, research was conducted to determine the average response time of an ambulance to the patient and the transportation time of the patient to the hospital. Per the research, the average time the ambulance spent to travel to the patient was within the range 16.2-17.6 minutes, whereas the transportation time of the patient was 82 minutes. With this information, I concluded that bad traffic management could be a cause of the vast difference between the average response time and the average transportation time. Also, the increase in transportation time puts the patient at a disadvantage since there is minimal time available for the doctors to treat or save the person's life [1].

1.4 Attempts to solve this problem:

In Ghana, Motor transport and traffic directorate (MTTD) assigns some of their personnel to regulate the traffic in the best way possible. However, this is not efficient because, during unfavorable weather conditions, the MTTD personnel are not able to do the work efficiently.

In addition, some students around the world have made systems with cameras to detect cars. With the data acquired after vehicle detection, algorithms were implemented to coordinate traffic lights correctly. Thus, giving priority to heavily congested traffic lanes. This attempt is by far better than human intervention, but with this alternative, there is no redundancy. When a camera develops a fault, and it does not function as it must, the system will fail to perform its objective [2].

1.5 Solution:

With the aid of an inductive loop traffic management system, traffic congestion could be monitored and controlled effectively to make transportation calm. This work proposes the use of inductive loops to detect the presence of a car. With this form of detection, graphs will be drawn to gain the peak values of the traffic congestion. With the peak values acquired, more time will be allocated to the congested traffic lane in order of decreasing peak values. Also, with the aid of real-time tracking, the ambulances' path of travel will have the traffic lights turn green during travel to reduce time taken to maneuver through cars.

To test this, a miniature road will be constructed with an intersection and mini toy cars placed on it. Some test cases will be implemented on the miniature road. For instance, there will be a test case for the cars at one traffic light and no cars at a traffic light. There will be a test case when all traffic lights have cars in front of them. There will be a test case for the responsiveness of the coordinated control system.

Chapter 2: Literature Review

Traffic management is an area researched by many engineers. Many concepts such as the use of inductive loops, image processing, radars and magnetic sensors have been proposed by several engineers as fitting techniques for managing the current growing urban traffic levels. In spite of suggestions, inductive loop is one of the most researched methods of traffic management(e.g. [3], [4], [5]). However, it has not been deployed in many countries since it is an invasive technique (road drilling is required for installation). Due to this constraint, Internet of Things and image processing has been the resort many countries have used for the deployment of their traffic management systems.

2.1 Image Processing:

With regards to intelligent traffic lights, Agrawal Nidhir [6] reported on the significance of the use of image processing to manage traffic congestion. Per the research in this paper, image processing was adopted because of the low cost associated with incorporating it to a traffic system. Image processing is a vehicle detection technique which implores the use of cameras to acquire images and process the image through ten stages. The paper suggests image acquisition, image enhancement, image restoration, color detection, erosion and dilation, segmentation and description as the ten stages of image processing.

When an image is captured, the image is converted into grayscale and later converted to a binary image. The binary image is enhanced after some effects are applied; the image goes through a restoration algorithm to undo the defects in the image. The restored image is passed through a color detector to obtain several color wavelengths and later eroded and dilated to remove abnormalities and fill broken areas. Next, objects in the output image are assigned labels such as humans, obstacles and vehicles. Objects labelled as vehicles indicate vehicle presence.

However, this technique of vehicle detection is not reliable in unfavorable weather conditions; images taken during heavy rain, fog or snow could wrongly influence the decisions of the processing unit. Correspondingly, the camera used for capturing images must be maintained properly. Proper maintenance of the camera involves cleaning of the camera lens. The camera lens is blurred by dust and other environmental particles, and when blurred, the image quality would be reduced. Thus, maintenance is not simple. Correspondingly, the cost of procurement and implementation of this technique is high.

2.2 Vehicle Detection (Sensors):

Varshan [7] designed a traffic management system with the application of Internet of Things. The aim of this paper was to detect traffic levels at lanes using ultrasonic sensors. In designing the system, he considered the use of devices such as LCD, LED, ultrasonic sensors and an ARM7 controller. The system was designed to detect traffic level status at intersections with the aid of ultrasonic sensors. An array of ultrasonic sensors was mounted at roadsides and connected to an ARM 7 controller. The controller is connected to WI-FI to transmit data from sensors to a web server unit. The ultrasonic sensors record vehicle present by comparing the change in distance to the original distance. After vehicle detection, the data acquired is then used to interpret traffic levels, stored and displayed on an LCD to enable a user to see traffic levels. Additionally, Jun et al[8] describe how Japan has adopted the use of ultrasonic sensors for vehicle detection in traffic management. In Japan, doppler ultrasonic sensors have been mounted above the road to count and obtain vehicle presence. With this knowledge, we can deduce that ultrasonic sensors can help improve traffic management systems. They are also cheap to acquire and implement. However, the ultrasonic sensor reading is affected by temperature change and extreme turbulence. Also, the pulses sent may not detect fast-moving vehicles.

Chapter 3: Design

In this chapter, the design of the system is discussed. The design decisions were deduced by using Pugh matrices to compare and choose the best technology for the project. Also, the system architecture is elaborated, and several resources which would be used are defined. With regards to the design of the system, affordability is one paramount factor when making design decisions.

The system comprises:

- Vehicle detection component
- Traffic lights signal sequence algorithm
- Pedestrian enabled system
- Tracking System
- App for monitoring

3.1 Design Decisions and Pugh Matrices

For the implementation of the proposed solution, some technologies were decided for different constituents of the system.

3.1.1 Vehicle detection component

For the aspect of vehicle detection, there are many approaches that could be used. These approaches are namely; inductive loops, ultrasonic sensors, cameras and microwave radars. Table 3.1 shows how the best detection decision was made. The prominent metrics for choosing a vehicle detection method are cost, accuracy, ease of maintenance, ability to detect both moving and stationary vehicles and speed.

Table 3. 1: Pugh matrix for best vehicle detection technique

#	Criteria	Surveillance Camera (Baseline)	Weight	Inductive Loops	Microwave Radar	Ultrasonic
1	Cost	0	4	1	1	1
2	Accuracy	0	4	1	-1	-1
3	Ease of Maintenance	0	2	1	-1	1
4	Ability to detect both moving and stationary vehicles	0	3	-1	-1	-1
5	Speed	0	3	1	0	1
	total			10	-5	2

In a Pugh matrix, 0 depicts the device is the same as the baseline, +1 depicts the device is better than the baseline, and -1 depicts the device is worse than the baseline.

From the Pugh matrix, the best method of car detection is the use of inductive loops due to its low cost, ease of maintenance, fast vehicle detection, and accuracy.

The inductive loop used should have the following features:

- Resistant to mist or dirt
- Performance during bad weather conditions (wind, rain) should be high

With concerns to the best method of car detection evaluated by the Pugh chart, due to the unavailability of components for the demo aspect, we will resort to the second-best method of detection. The second best is the use of ultrasonic sensors. This alternative sensor and inductive loop sensor are analog sensors. One analog sensor can be represented by another analog sensor since they generate analog outputs. Therefore, the ultrasonic sensor can be used to depict the inductive loop sensor.

The ultrasonic to be used should have the following features:

- Detect transparent objects
- Resistant to mist or dirt

3.1.2 Pugh chart for processing unit

For the processing unit of vehicle detection, there are many options such as Arduino, Raspberry Pi and Atmega. Table 3.2 shows how the best processing unit was chosen. The most important factors for choosing a processing unit are cost, size, ease of use, performance and speed.

Table 3. 2: Pugh matrix to choose the best processing unit

#	Criteria	Arduino (Baseline)	Weight	Raspberry- Pi	ATMEGA
1	Cost	0	2	-1	1
2	Size	0	2	1	1
3	Ease of Use	0	3	-1	-1
4	Performance	0	4	1	-1
5	Speed	0	3	1	-1
	total			4	-6

From the Pugh matrix, the best processing unit is the Raspberry Pi, mainly due to its ease of use, performance and speed.

3.2 Design Block Diagram and Circuitry:

Per these two Pugh charts, the design of the actual system will be as depicted in Figure

3.1.

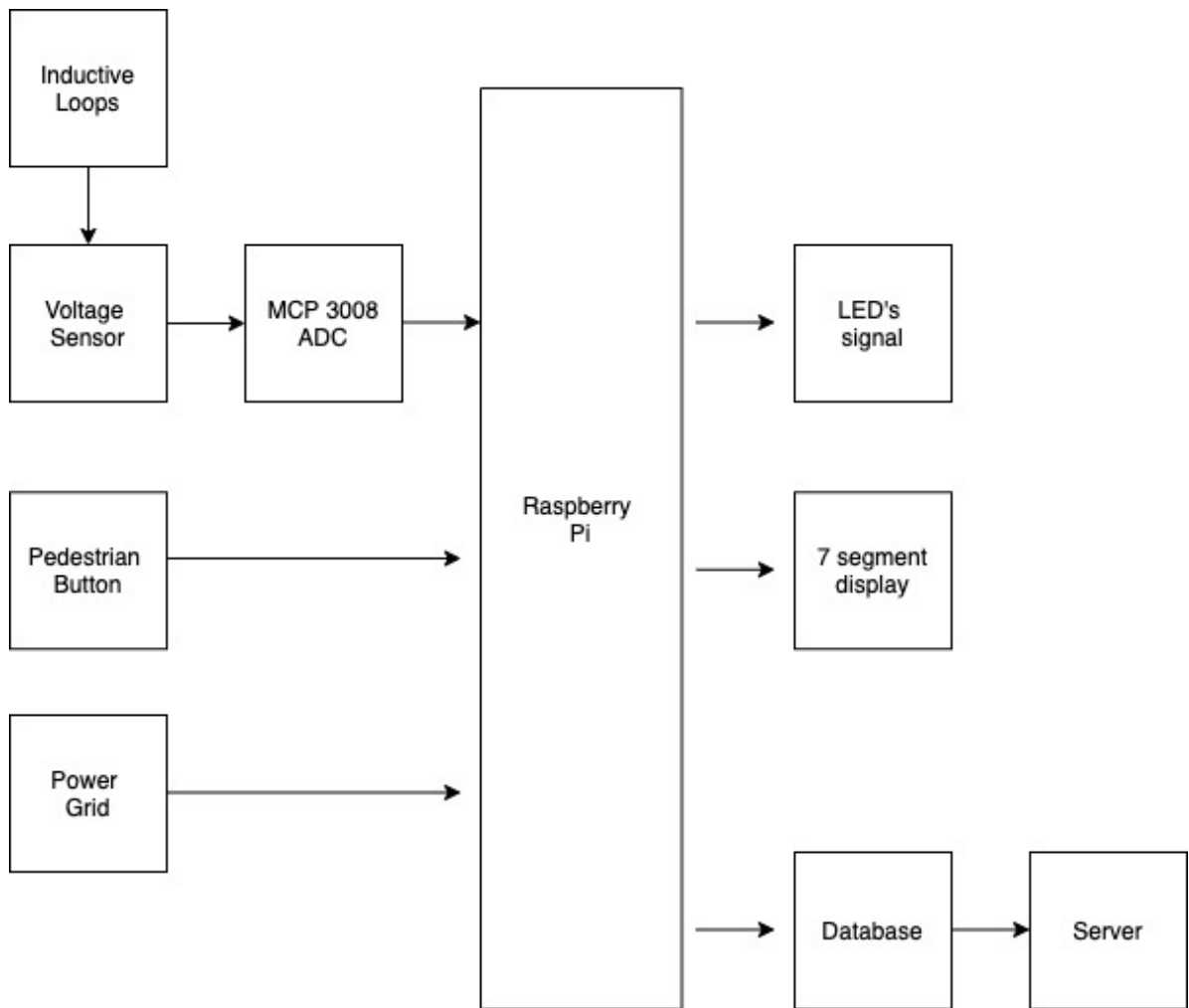


Figure 3. 1: Block diagram for inductive loop system

With regards to circuitry implementation of inductive loops with the Raspberry Pi, the circuit will be as depicted in Figure 3.2.

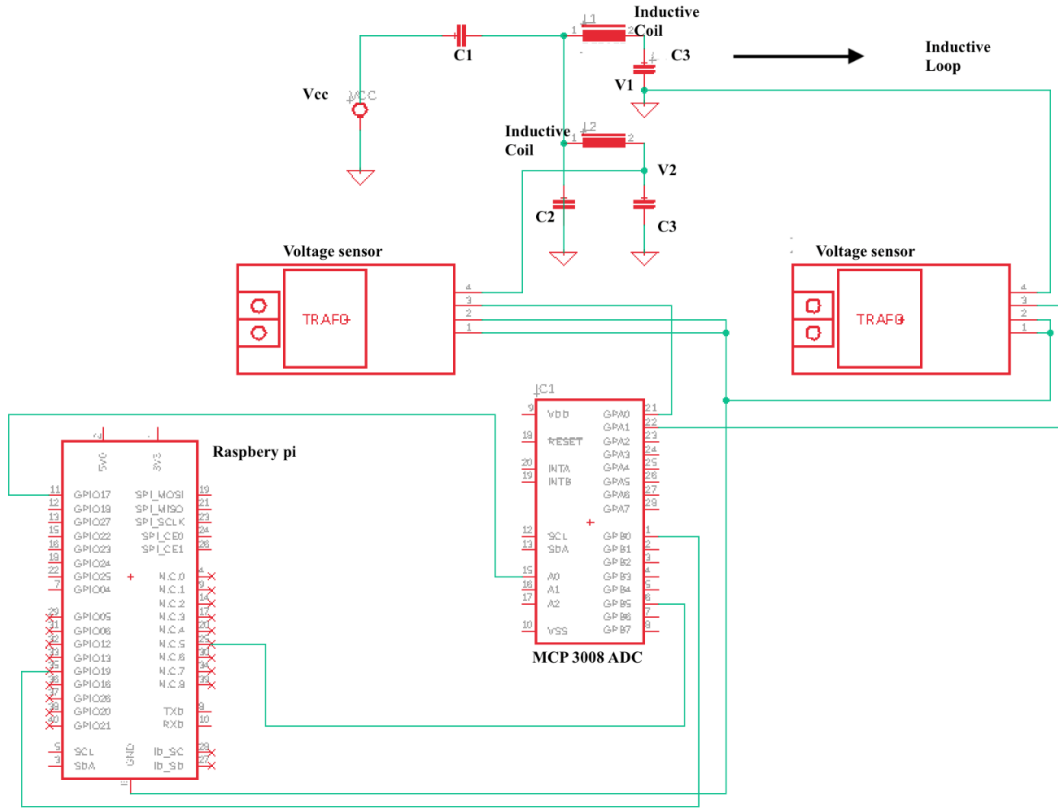


Figure 3. 2: Circuit for inductive loop system

For exposition on Figure 3.2, the inductive loop will give an output which will be read by the voltage sensor and the voltage sensor will use the analog to digital converter to send the data to the Raspberry Pi. Also, 4 capacitors, 2 inductive coils, Vcc (power grid) and ground are used to fashion the inductive loop and used to generate the output of the loop. When a car passes over the coils (loops), the inductance of both loops change. Correspondingly, a change in inductance of the loops causes a change in the voltage at V2 (across C3) and V1 (across C3). This change in voltage indicates vehicle presence.

3.3 Design Iteration for Traffic Management System:

For the prototype design, it will be implemented with the second-best option from the vehicle detection Pugh chart and best option for the microcontroller Pugh chart. The design of the prototype system will be as shown in Figure 3.3

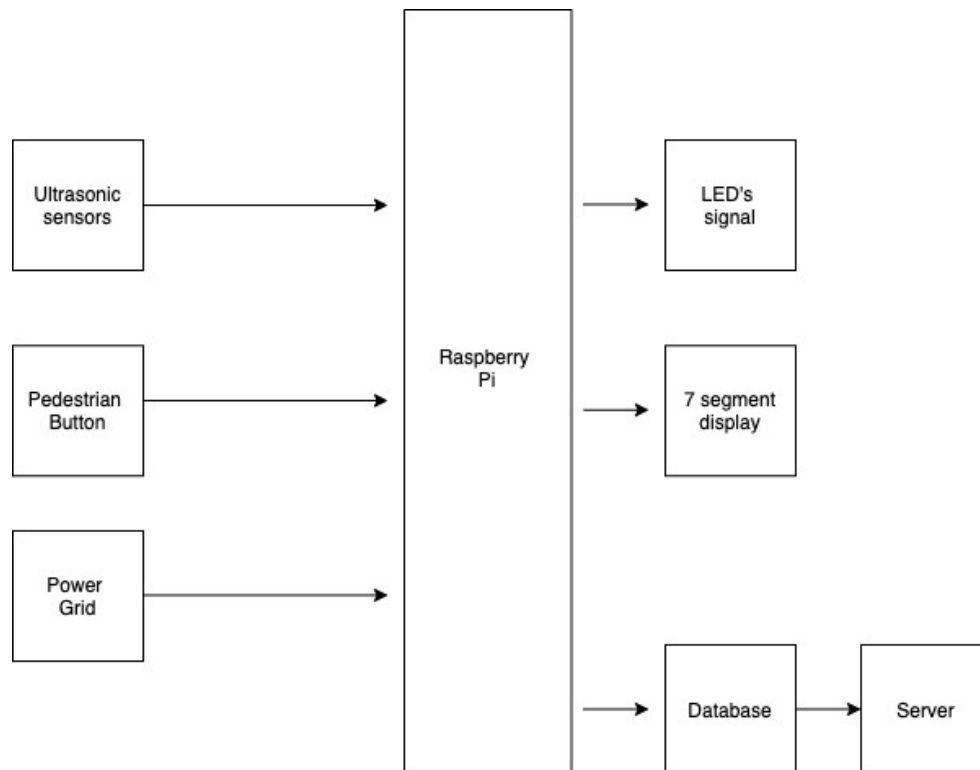


Figure 3. 3: Ultrasonic system block diagram

For circuitry implementation of the prototype block diagram, a traffic light will have its system as shown in Figure 3.4

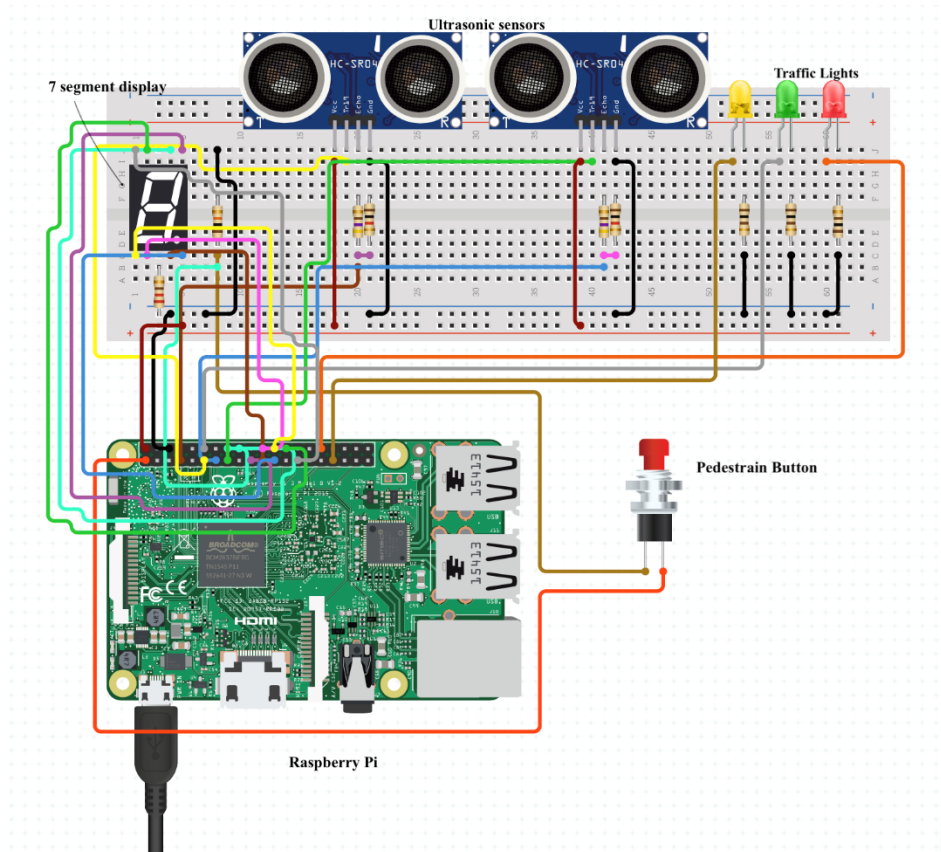


Figure 3. 4: Circuit for the ultrasonic system

With regards to the system requirements, the system must have these features:

1. The system should be able to coordinate traffic from a lane to its preceding lane.
2. The system should be able to detect a vehicle
3. The system should have a web application which interacts with the database to gain the traffic density on lanes of each intersection and show the traffic density growth per time.
4. The system should work independently of human interaction.

With regards to user requirements, the system must have these features:

1. The system should take user input
2. The signal lights should be clear to see
3. There should be a timer for pedestrians to know how soon the light will signal red.

4. The system should give priority to emergency vehicles.

With regard to non-functional requirements, the system must have these features:

1. The system must be secure and require administrative credentials to access it.
2. The system must be reliable.
3. The system must be fast

3.4 Design Decision for Tracking System:

For the implementation of the priority vehicles, the system will be as shown in Figure 3.5.

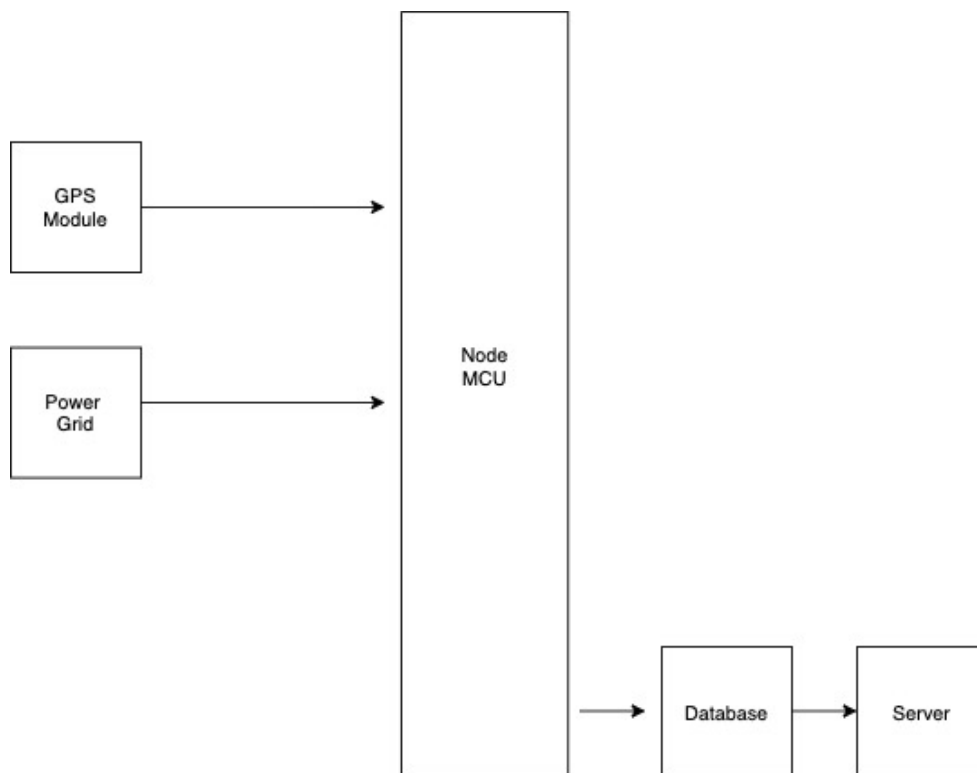


Figure 3. 5: Tracking System Block Diagram

For circuitry implementation of the block diagram, a tracking system is shown in Figure 3.6.

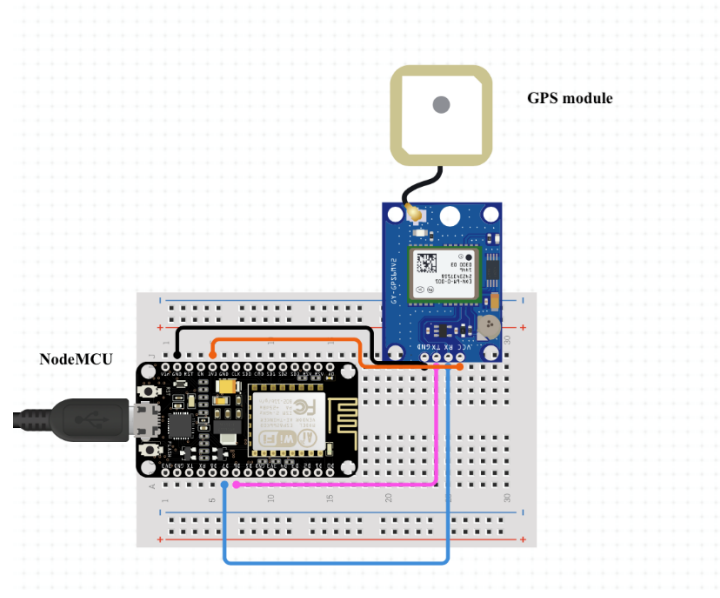


Figure 3. 6: Tracking System Circuit

3.5 System Architecture:

In this traffic management system, several ultrasonic sensors will be placed perpendicular to the road of each lane on the intersection, and each of these sensors is connected to a Raspberry Pi. The ultrasonic sensors check for the presence of a vehicle and feed data (number of cars and the presence of a car) to the Raspberry Pi. The Raspberry Pi then allocates a maximum average green time for the green signal. US highway administration[9] deduced that the maximum green time of an average vehicle is 7 seconds. From this, the Raspberry Pi multiplies the number of cars counted by 7 seconds. However, if the number of cars exceeds a threshold, the Raspberry Pi will allocate a time which depicts an average of 10-20 cars, depending on the lane size. When the time for green signal ends the Raspberry Pi then activates the ultrasonic sensor to count the number of cars and pushes the number to the database. The Raspberry Pi then activates sensors on the next lane. When there is a car present, it goes through the process stated in the previous sentences above. However, when the ultrasonic sensor does not sense any car, the Raspberry Pi will allocate a green time of zero and moves to the next

traffic light lane which follows. With the priority vehicles, the ESP module will initialize the GPS module to gain latitudes and longitudes; these coordinates will be sent to the database. The Raspberry Pi will then check if the coordinates in the database are similar to the coordinates of the traffic light location or if it indicates proximity to a traffic light. If the conditions are true, the Raspberry Pi will quickly turn all light signals red on lanes adjacent to the lane with the priority vehicle, and the lane with the priority vehicle will have its traffic signal turn green.

Furthermore, for coordinated control, the Raspberry Pi will extract the data (congestion rate) of the road ahead from the database and use the retrieved data to make decisions such as deciding a time for green light. This process is summarized in Figure 3.5.

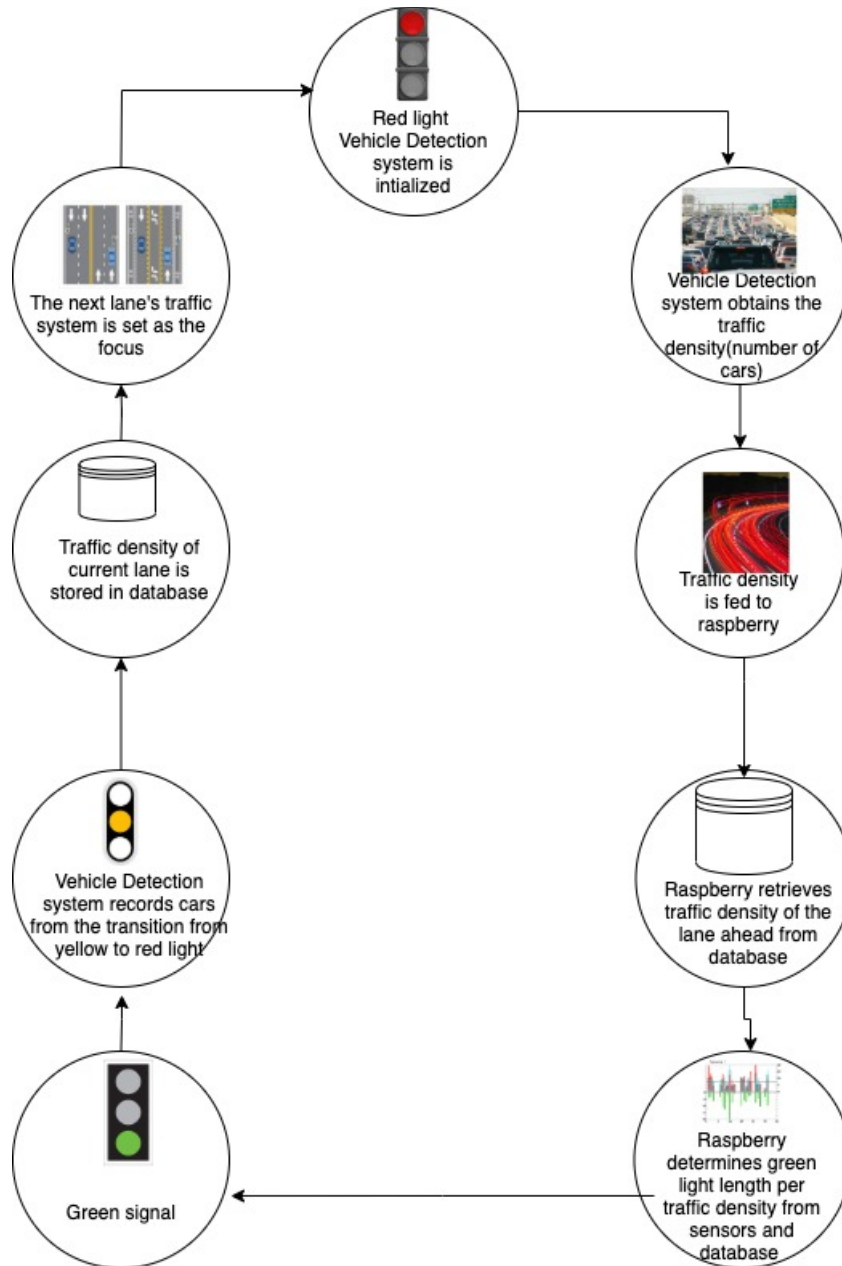


Figure 3. 7: System Architecture

Figure 3.6 shows the working of the system with regards to giving emergency vehicles priority. The description of the arrows are as follows:

- A- GPS module to transmit acquired latitudes and longitudes to node MCU located in the car for processing

	For numbering conventions, the Raspberry Pi has two types: pin numbers as known as Board and Broadcom, also known as GPIO numbers. For the implementation of this project, the Board numbering system will be used.
Push Button	Push Button is a type of switch which consists of a simple electric mechanism or air switch mechanism to turn something on or off. [11] In this project, the pushbutton is used as a pedestrian button to enable pedestrians to cross a busy road.
Ultrasonic Sensor	Ultrasonic sensors are transducers that convert ultrasound waves to electrical wave. It operates by emitting sound waves at a high frequency, then waits for the sound to be reflected back. Its transmitter and receiver are encompassed into one packaging.[12] With regards to this project, the ultrasonic sensors will be used to count the number of cars in a lane.
7 Segment Display	7 segment display is a set of seven segments shaped LED, arranged to form a squared 8. [13]In this project, the seven-segment display will be used to display the timer for pedestrians to cross a road.
Neo Gps module	This is a GPS module which functions as a GPS receiver with a built-in 25 x 25 x 4mm ceramic antenna for the strong satellite search capability. It has the ability to save data when the power is cut. This device will enable the system to give priority to an emergency vehicle upon arrival to the traffic light.
Node MCU	Node MCU is a low-cost open-source firmware developed for ESP8266 WIFI chip. It is implemented in C. It has 128Kb of RAM and 4Mb of flash storage. It integrates a 802.11b/g/n HT40 Wi-Fi transceiver to connect to a WIFI network, internet and set up a network of its own[14]. This component will be used as the controller for the tracking system.
LEDs	Led is a semiconductor which produces light. [15].This component will be used as the traffic signal lights.

Chapter 4: Methodology

The implementation of the design in the previous chapter was done at two levels: hardware implementation and software implementation.

4.1 Hardware Implementation

4.1.1 Vehicle Detection

To ensure drivers and passengers do not wait for long at traffic intersections, an algorithm was implemented based on ultrasonic sensors to detect vehicle presence in front of a traffic light. The sensors operate with pulse waveforms and provide vehicle count, presence and occupancy information. The pulse waveforms measure distances to the road surface and assign the distance to the road to a variable ($distance_{2road}$). When a vehicle interrupts the signal from the sensor to the ground, the current distance measured will be less than the distance to the road ($distance_{2road}$). Hence the sensor will interpret that measurement as the presence of a vehicle. However, when it does not detect vehicle presence, the system's focus moves to the next lane per the traffic coordination sequence. For deployment of the prototype the ultrasonic sensors were placed perpendicular to the road at a distance of 5cm. Figure 4.1 shows how it will be placed.

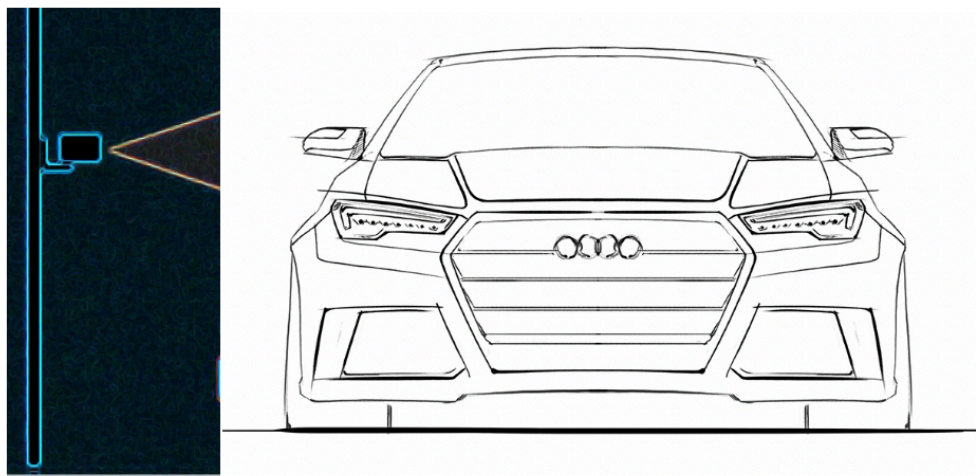


Figure 4. 1: Positioning of the sensor

4.1.2 Acquisition of traffic density

To acquire traffic density, another algorithm was implemented based on the output from ultrasonic sensors. When the ultrasonic sensors detect a car, the count of a car of the lane in focus will increase. With this in place, the Raspberry Pi will sequentially activate each sensor with a time interval of 50 microseconds. After a delay of 20 microseconds, the results are recorded, then the Raspberry Pi will process the result and increment the lane's vehicle count. The Raspberry Pi sets the vehicle count to zero when the green signal of the lane in focus comes on.

4.1.3 Green light sequencing and length determination

For green light sequencing and length determination, the Raspberry Pi uses the obtained traffic density to give the time length of the green signal. For traffic coordination between a lane and its preceding lane, the traffic density of the lane ahead is obtained and factored in the determination of the time length of the green signal alongside traffic density of that particular lane. As stated initially, the maximum time it takes a vehicle to move when the green signal is on is 7 seconds. With this insight, 7 seconds was set as the standard green time for one vehicle.

4.1.4 Pedestrians interrupt

For pedestrian interrupt, this is the implementation of how the system enables the pedestrians to cross the road. This was implemented by the use of a push-button to take inputs as requests to be made to the Raspberry Pi. After the button was pressed, the Raspberry Pi halts traffic coordination sequencing by setting all traffic lights to red and initializes the timer to make pedestrians know the time duration for crossing a road. Figure 4.2 depicts the mechanism.

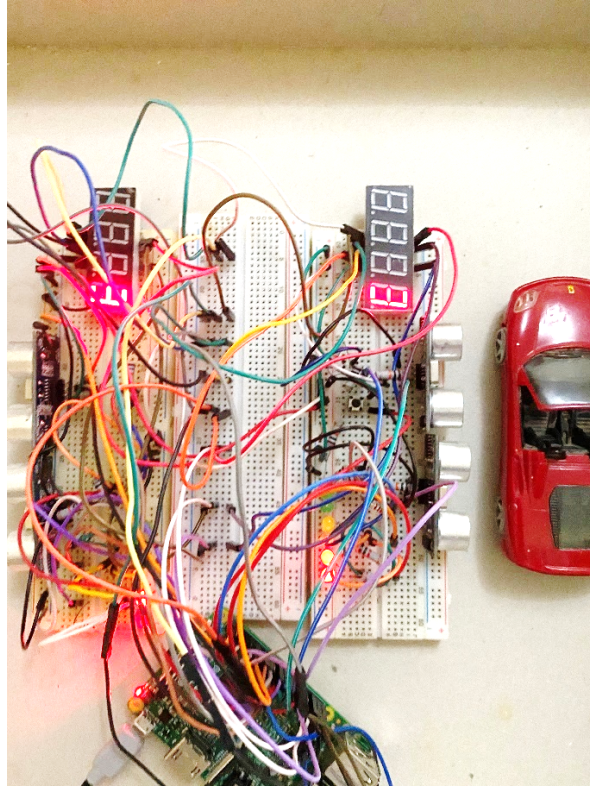


Figure 4. 2: Pedestrian button activated

4.2 Software Implementation.

For a software implementation, an application was created with angular framework in Ionic 4. This app (Trafmap) was created to pull data from the database to enable traffic regulatory bodies such as MTTD to know the current and past traffic congestion levels of a particular area. The application depicts the congestion level in the form of a line chart. Also, the app will display the number of cars which used each lane and the total number of cars which moved through the location where the traffic lights are situated. The app is shown in Figure 4.3

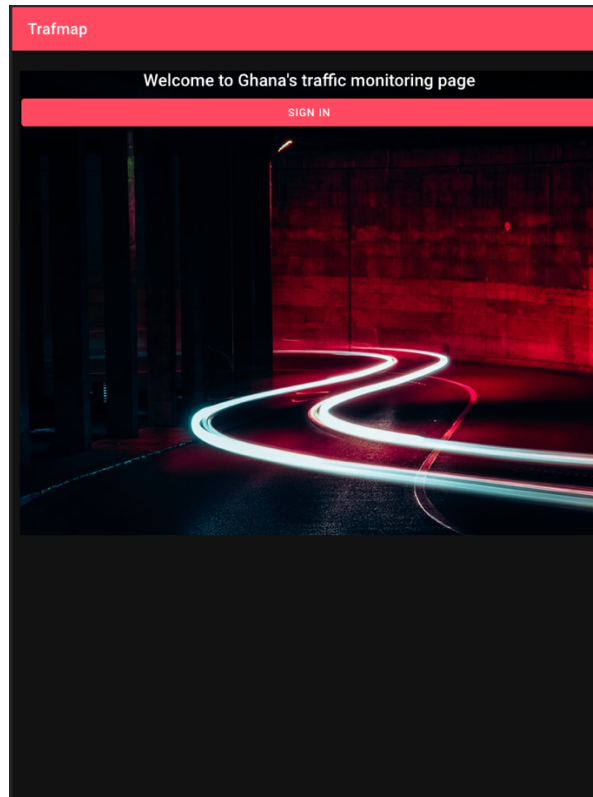


Figure 4. 3: Web application interface

In addition, a database was created to hold GPS coordinates of priority vehicles and hold the current number of vehicles on each lane, the total number of vehicles which used each lane, and the time each data was captured. This database was created on a server which can be accessed by the Raspberry Pi. The database relationship diagram is shown in Figure 4.4.

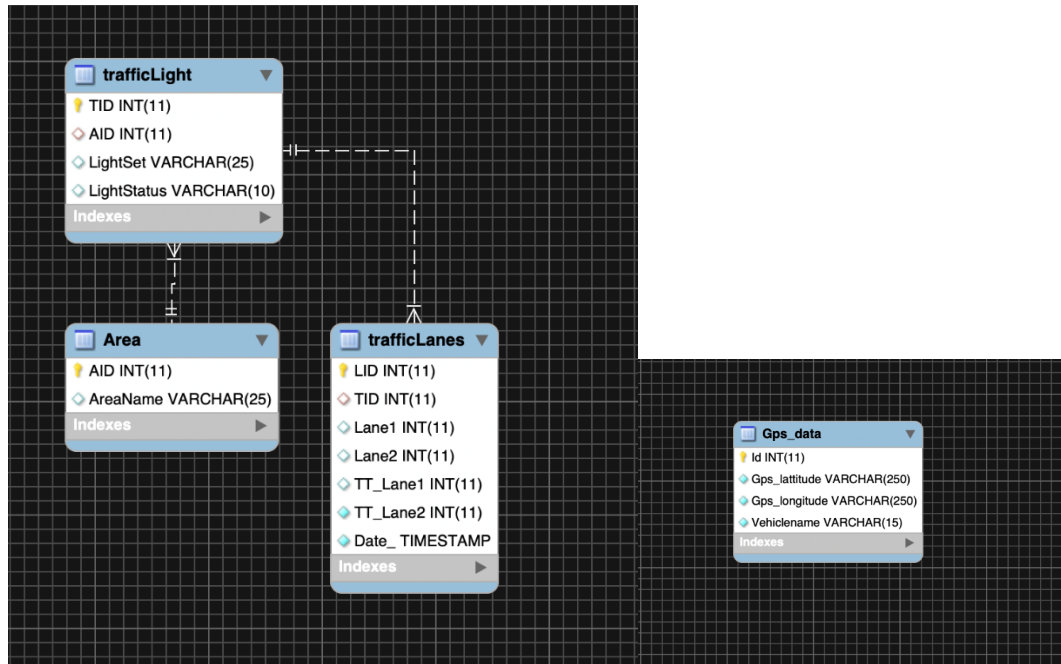


Figure 4. 4: Database Relationship Diagram

From the relationship diagram, one traffic light maps to many traffic lanes and one area maps to many traffic lights. To relate this with a real-life scenario, East Legon has many traffic lights, so this represents one area to many traffic lights relationship. Also, each set of traffic lights has a certain number of traffic lanes; this represents one set of traffic lights to many lanes. For the relationship diagram of the GPS, the Gps_latitude, Gps_longitude and Vehiclename were created in the Gps_data table.

Chapter 5: Test and results

In this chapter, the testing of the various constituents of the system is done, and the results of the system response for coordinated traffic control between a lane and its preceding lane is analyzed.

5.1 Vehicle Detection

The vehicle detection was tested using only two lanes. This was because the ultrasonic sensors needed for the system were inadequate. Likewise, the number of GPIO pins on the Raspberry Pi was limited in number. Each lane had three pairs of color LEDs (red, yellow and green) as the traffic signal light. The lanes used are shown in Figure 5.1.

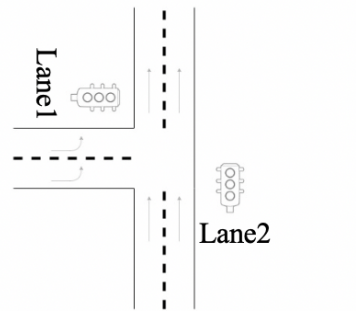


Figure 5. 1: Diagram of lanes being used

With these lanes, vehicle detection was tested. It was tested with the system, as shown in Figure 5.2.

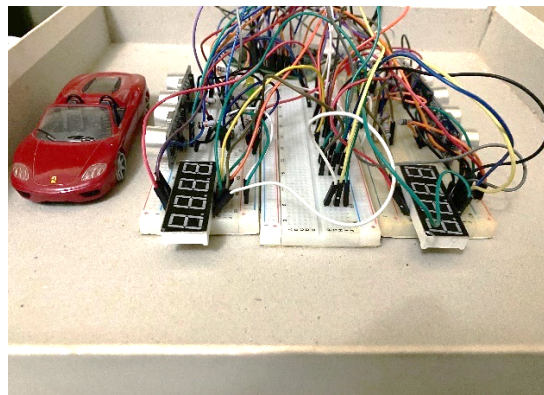


Figure 5. 2: Test Setup

For vehicle detection, two tests were implemented. The first test was implemented by placing the car on one lane and leaving the other lane vacant. When this was implemented, the vehicle was detected on the lane it occupied, and no vehicle was detected on the other lane. This result is shown in Figure 5.3.

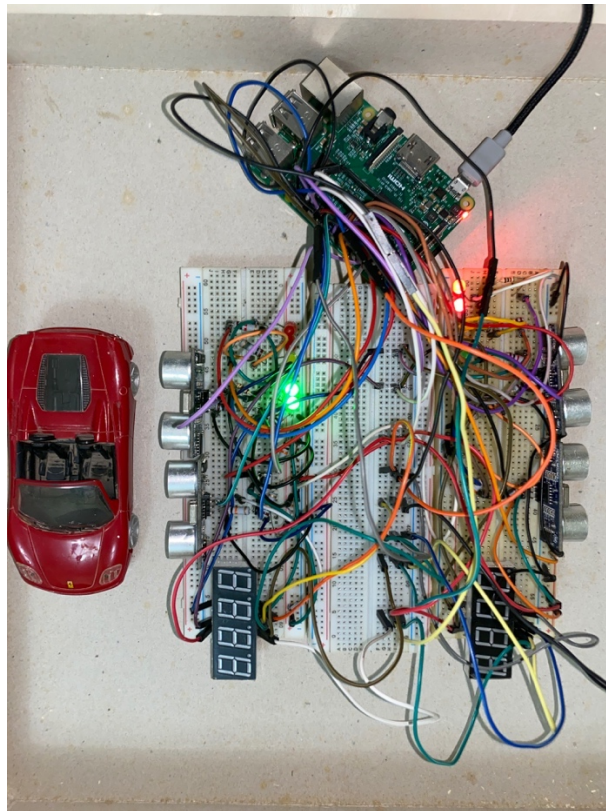


Figure 5. 3: Vehicle Detection Test

In Figure 5.3, the lane which is occupied by the vehicle has its green signal on and the unoccupied lane has its red signal on. The results show that the vehicle detection system works effectively.

5.2 Acquisition of traffic density, Green Light Sequencing and Length Determination.

The acquisition of traffic density was tested by placing several cars in the two lanes shown in Figure 5.1. The setup diagram used to test this is shown in Figure 5.4. In Figure 5.4 the folded papers are used to represent the cars.

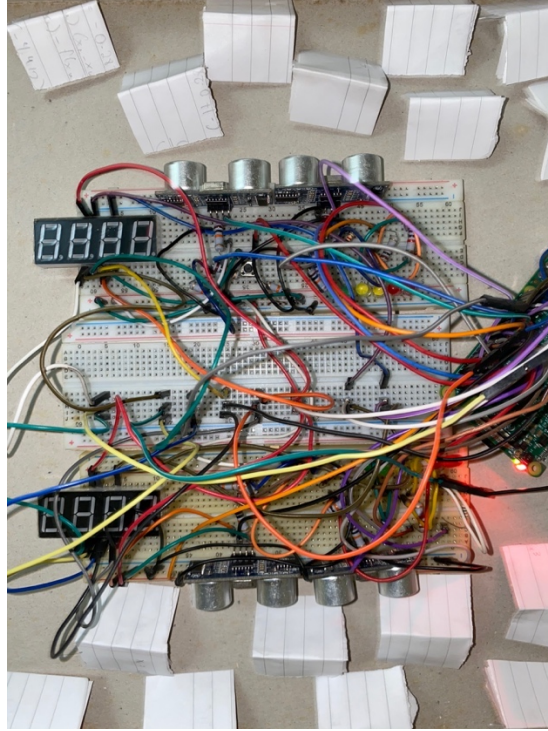


Figure 5. 4: Traffic Density Acquisition

The number of cars used, their corresponding lanes and the car count determined by the system are shown in Table 5.1

Table 5. 1: Results from test

Number of cars	Lanes	System car count
10 cars	1	10
5 cars	2	5

As shown in Table 5.1, the number of cars on a lane is the precise number of cars the system detected and counted.

For green light sequence and length determination, the algorithm was also tested on the two lanes, as shown in Figure 5.1. For this test, values of traffic density for each lane was generated using a for-loop in python. These values were fed into the algorithm to determine the green time and the sequence of the next green light. The algorithm was structured such that if

no car is detected on a particular lane, the traffic light will stay on the red signal. However, if there is a car detected, the number of cars will be accumulated and processed to gain the time length of the green signal in a time of 0.02 seconds, then the traffic light will turn green. The results of this test are shown in table 5.2

Table 5. 2: Results from Algorithm

Number of cars	Lanes	Red light status	Green time allocated
0	1	On	0
10	1	Off	67 seconds
20	2	Off	117 seconds
0	2	On	0
5	2	Off	32 seconds

The test result shown in Table 5.2 were generated by the green light and sequence algorithm. As stated in the system architecture, the maximum time allocated for an average vehicle to move when the green signal is turned on is 7 seconds (3.5 seconds of reaction time and 3.5 seconds of acceleration). With this, the algorithm multiplies the number of cars by 7 to get the green time, but 4 seconds is taken from the green time since the standard time for the yellow light is 3-6 seconds. From the table, the algorithm allocated 66 seconds of green time to 10 cars in lane1, but the algorithm allocated 116 seconds of green time to 20 cars. This was done because the standard green time for a lane is 120 seconds (including the amber light)[9]. Thus, subtracting the 3 seconds for the amber light from 120 seconds will result in 117 seconds of green time.

Correspondingly, the green light length of coordinated control was tested. This was tested with two main lanes which are shown in Figure 5.5.

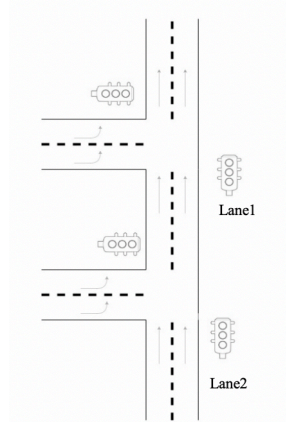


Figure 5. 5: Diagram of Lanes being used

The two lanes are main lanes of separate intersections; with two separate Raspberry Pi boards as the processing units. In order to determine the length of the green light for lane2, the traffic density of lane1 is factored in the algorithm which determines the green time. The algorithm pulls the traffic density of lane1 from the database and calculates the number of vacant spaces for vehicles in lane1 by subtracting the traffic density from the lane capacity defined in the algorithm. The difference obtained will be used in the green time determination. For this test, the values of traffic density for each lane was generated using a randomizer in Python.

Table 5. 3: Results from the Green Light Length Determination Algorithm

Lane1 capacity	Lane1 number of cars	Lane2 number of cars	Green time allocated
15 cars	5	6	38 seconds
8 cars	5	8	11 seconds
20 cars	10	20	66 seconds
5 cars	0	10	31 seconds

5.3 Webapp

The result of the home page interface is shown in Figure 4.3 in section 4.2. When the sign-in button is clicked, a sign in form is opened. This was necessary to ensure only authorized users can login in. The login form is shown in Figure 5.6. When a user logs in, the dashboard page opens. As described in the software implementation, the dashboard shows a line chart of traffic congestion over time. It also shows the number of cars which used each lane and the total number of cars which went through the area. Figure 5.7 shows the interface of the dashboard with some sample numbers.

Login
Email
Email is required.
Password
Password is required.
Submit
Forgot Passowrd

Figure 5. 6: Login Page Interface



Figure 5. 7: Dashboard Interface

5.4 Statistical test of response time of coordinated traffic control between a lane and its preceding lane.

A t test was conducted to see the responsiveness of the coordinated traffic control by comparing the execution time of the coordinated traffic control algorithm (coordinated traffic control of a lane and its preceding lane) with the execution time of the normal traffic control algorithm (traffic control on just one lane). Figure 5.8 shows the results of this test.

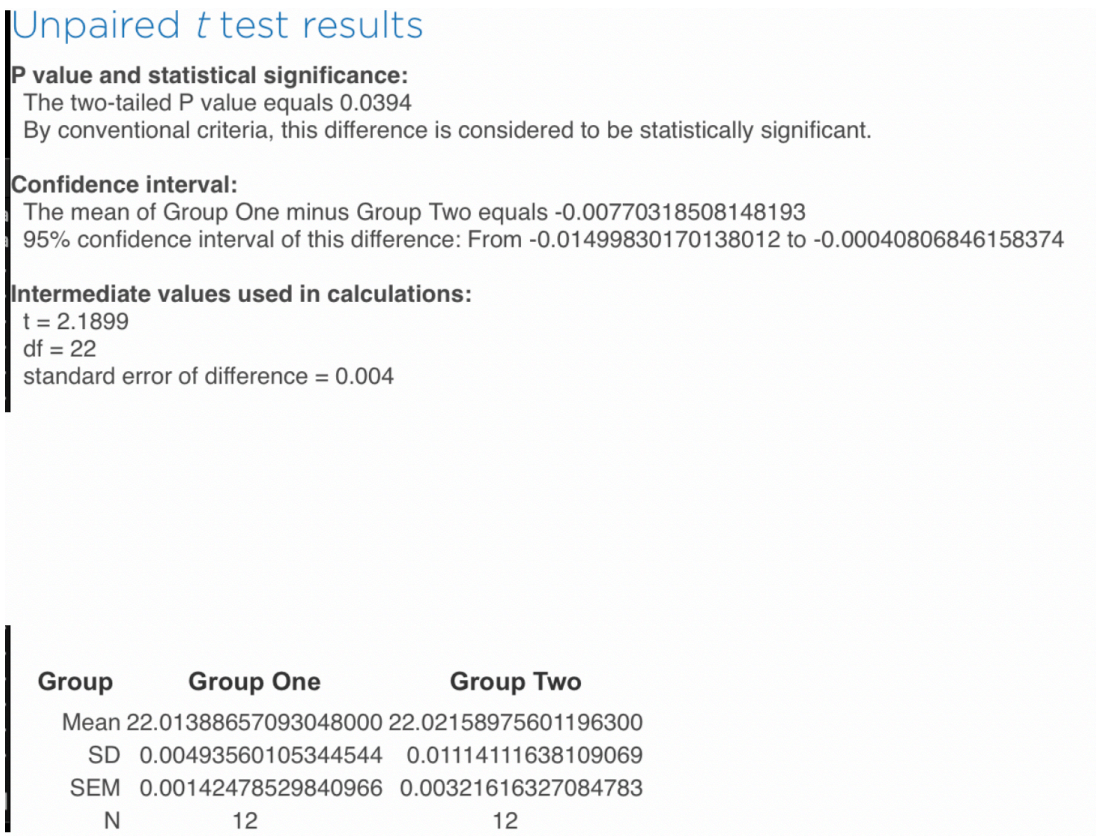


Figure 5. 8: T test results

These test results were generated with a web based t test calculator[16]. For this test the null hypothesis is that the means of the two samples are the same. But the p value acquired is 0.0394 which is less than 0.05, meaning the null hypothesis will be rejected. This result was obtained because the normal traffic control algorithm goes through fewer processes than the coordinated traffic control algorithm. Normal traffic control algorithm checks for traffic density

of a lane and uses the traffic density to determine the green time. But, the coordinated traffic control checks for traffic density and also accesses traffic density of the preceding lane from the database to determine the green time.

5.5 Test and analysis of wait time of the smart system and of a traditional traffic light

A test was conducted with a stopwatch to acquire the wait time (time for the signal to switch from red to green) for one car when the light traffic signal is red. The test was conducted on a traditional traffic light at the Okponglo, Legon intersection (Figure 5.9)



Figure 5. 9: case study area (Okponglo, Legon intersection)

This test has a sample size (one sample is the wait time of the first car in a queue on the fourth lane in front of a traffic light) of 15 and was used to generate graphs in Excel to analyze how optimal the system is in solving traffic congestion. The lane used is shown in Figure 5.10.

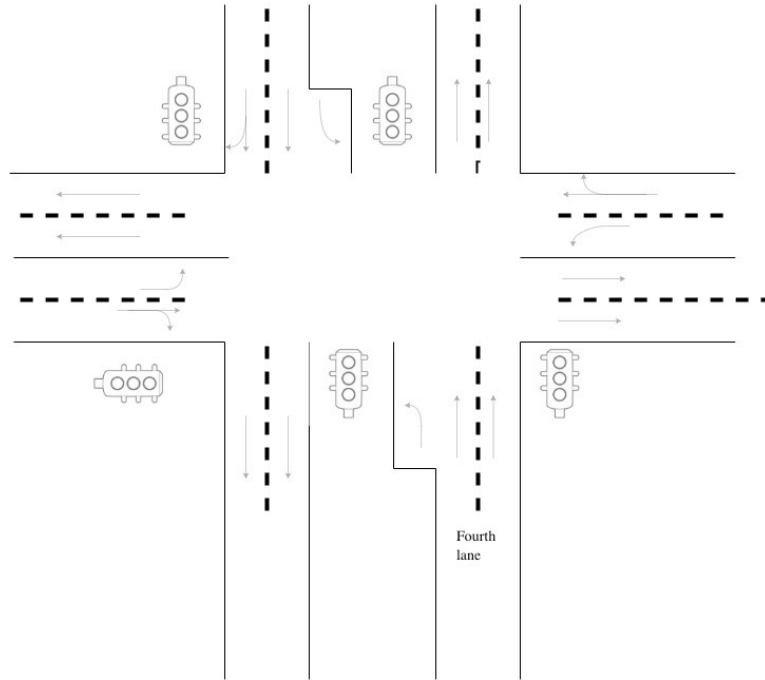


Figure 5. 10: Diagram of lane (Okponglo Legon intersection) being used.

After recording the wait times on the fourth lane of the traditional traffic light, the wait-times of the smart system was also recorded. To record the wait-times on the fourth lane of the smart system, the intersection in Figure 5.10 of the Okponglo, Legon junction was adapted. The wait times are shown in Appendix A. The wait times accumulated were used to obtain average wait times of each traffic light system. These averages were used to generate a bar graph. Figure 5.11 shows the results of this test.

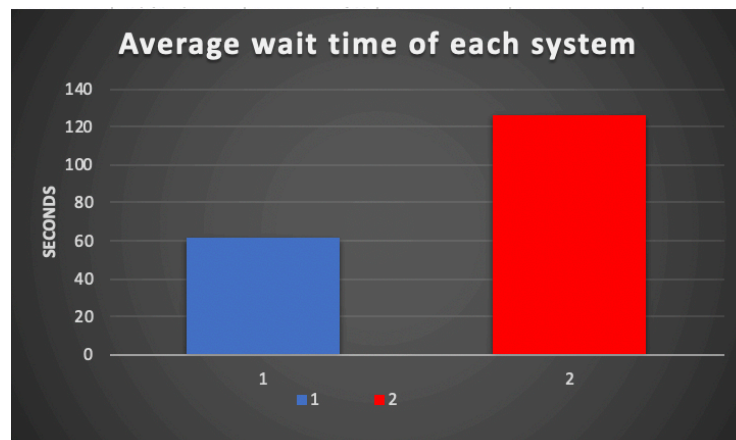


Figure 5.11: Graph of plotted averages

Key:

- 1 - Smart traffic system
- 2 – Traditional traffic system

From the graphs, the smart system had less waiting times than the traditional traffic light.

Thus, we can conclude that the smart system is optimal enough to manage traffic congestion.

Chapter 6: Conclusion, Limitations and Future Work

6.1 Conclusion

To design and successfully implement a low cost and smart traffic management system, the vehicle detection system, traffic light sequence and length determination algorithm, pedestrian enabled system and tracking system must be accurate and precise.

The vehicle detection system with ultrasonic sensors can be improved with doppler ultrasonic sensors for more accurate results and better working of the system. The doppler ultrasonic sensors are less susceptible to interference, unlike the ones used in this project. Also, the vehicle detection system with the inductive loops can be exchanged for vehicle detection system with doppler sensors because as cars keep passing over the inductive wires, the accuracy of the results will be reducing. This is because the wires will be subjected to stress from the moving cars and temperature.

In the green light sequencing and length determination, the traffic density for each lane is important. This is the data acquired from the acquisition of traffic density process. To ensure traffic congestion is fairly reduced on each lane, the standard green time for a car is introduced. Also, to ensure the gap in between two intersections is not blocked by cars entering a lane, the lane capacity is introduced. With all these, the wait time of vehicles behind traffic lights at intersections is reduced. Due to the reduction the system proposed has lesser average wait time than a traditional traffic light wait time.

For the tracking system, the Node MCU must have a strong internet connection to ensure the coordinates acquired from the GPS module is sent to the database and retrieved by the Raspberry Pi to provide a quick response.

6.2 Limitations

The system is likely to produce inaccurate results in very windy areas. This is because the sound waves that are sent and received by the trigger and echo components are not optimized to generate accurate results. If doppler ultrasonic sensors are adopted, the sound waves could be optimized to reduce any interference by wind and generate more accurate results.

For the coordination control, the response time of the system can be optimized through Pi to Pi interaction instead of the communication through the use of a database.

6.3 Future Work

In situations where sensors are faulty or destroyed, algorithms can be used to process the data accumulated over a period of time to predict how the traffic density will be, to coordinate and control traffic.

Also, when most vehicles break down in a lane, traffic congestion increases since cars will have to use road diversions to travel. This can be solved with cameras and the aid of machine learning. These tools can be used to determine how long a particular car has been stationary. When results are acquired and the situation is affirmed (for example the breakdown of a vehicle), the system can alert officials of towing enterprises to remove the vehicle.

Finally, a backup power source can be activated when there is a power cut. During power cuts, traffic congestion levels increase since the traffic management system will be deactivated. To ensure this does not occur; a backup power source will be needed to ensure the traffic system still runs when power is cut. Also, in situations where the backup power fails, the system must be able to send an alert to the MTTD to ensure one of their personnel moves to the area to regulate traffic.

References

- [1] M.-N. Mahama, E. Kenu, D. A. Bandoh, and A. N. Zakariah, "Emergency response time and pre-hospital trauma survival rate of the national ambulance service, Greater Accra (January – December 2014)," *BMC Emerg Med*, vol. 18, no. 1, p. 33, Oct. 2018, doi: 10.1186/s12873-018-0184-3.
- [2] P. Choudekar, "Implementation of image processing in real time traffic light control - IEEE Conference Publication."
<https://ieeexplore.ieee.org/document/5941662?arnumber=5941662> (accessed Oct. 13, 2019).
- [3] L. Bhaskar, A. Sahai, D. Sinha, G. Varshney, and T. Jain, "Intelligent traffic light controller using inductive loops for vehicle detection," in *2015 1st International Conference on Next Generation Computing Technologies (NGCT)*, Sep. 2015, pp. 518–522, doi: 10.1109/NGCT.2015.7375173.
- [4] S. S. Mohammed Ali, B. George, and L. Vanajakshi, "Multiple inductive loop detectors for intelligent transportation systems applications: Ramp metering, vehicle re-identification and lane change monitoring systems," in *2013 IEEE Symposium on Computers Informatics (ISCI)*, Apr. 2013, pp. 176–180, doi: 10.1109/ISCI.2013.6612398.
- [5] S. Sheik Mohammed Ali, B. George, L. Vanajakshi, and J. Venkatraman, "A Multiple Inductive Loop Vehicle Detection System for Heterogeneous and Lane-Less Traffic," *IEEE Transactions on Instrumentation and Measurement*, vol. 61, no. 5, pp. 1353–1360, May 2012, doi: 10.1109/TIM.2011.2175037.
- [6] A. Nidhi and S. Amit, "Intelligent Real Time Traffic Controller Using Image Processing – A Survey," vol. 4, no. 4, Apr. 2015.
- [7] N. Varsha and sushilku Rajbhoj, "An Intelligent Framework for Vehicle Traffic Monitoring System using IoT," 2017.
- [8] J. Liu, J. Han, H. Lv, and B. Li, "An Ultrasonic Sensor System Based on a Two-Dimensional State Method for Highway Vehicle Violation Detection Applications," *Sensors*, vol. 15, no. 4, pp. 9000–9021, Apr. 2015, doi: 10.3390/s150409000.
- [9] "Traffic Signal Timing Manual: Chapter 5 - Office of Operations."
<https://ops.fhwa.dot.gov/publications/fhwahop08024/chapter5.htm> (accessed Apr. 19, 2020).
- [10] C. Chng, "Introduction to IoT Using the Raspberry Pi."
<https://www.codemag.com/Article/1607071/Introduction-to-IoT-Using-the-Raspberry-Pi> (accessed Feb. 24, 2020).
- [11] Herga, "What is a push button switch?," Nov. 19, 2018.
<https://www.herga.com/pressrelease/detail.php?aid=100&did=What-is-a-push-button-switch?> (accessed Feb. 24, 2020).
- [12] "All About Ultrasonic Sensors & How They Work with Arduino | Arrow.com," Apr. 04, 2018. <https://www.arrow.com/en/research-and-events/articles/ultrasonic-sensors-how-they-work-and-how-to-use-them-with-arduino> (accessed Feb. 24, 2020).
- [13] M. Rouse, "What is seven-segment display? - Definition from WhatIs.com," *WhatIs.com*, Jan. 04, 2012. <https://whatis.techtarget.com/definition/seven-segment-display> (accessed Feb. 24, 2020).
- [14] "Insight Into ESP8266 NodeMCU Features & Using It With Arduino IDE (Easy Steps)," *Last Minute Engineers*, Aug. 20, 2018. <https://lastminuteengineers.com/esp8266-nodemcu-arduino-tutorial/> (accessed Apr. 19, 2020).

- [15] “What is an LED: Some Basic Information (Revised 2019) | MyLEDLightingGuide | MyLEDLightingGuide.” <https://www.myledlightingguide.com/blog-what-is-an-led> (accessed Apr. 19, 2020).
- [16] “GraphPad QuickCalcs: t test calculator.” <https://www.graphpad.com/quickcalcs/ttest1.cfm> (accessed May 10, 2020).

Appendix

Appendix A: Recorded wait times

Smart System	Traditional System
54.3	128.63
38.76	124.2
53.58	124.58
65.4	124.3
66.45	124.15
74.4	127.96
63.45	129.89
65.46	126.1
66.6	123.7
66.45	124.13
55.75	126.11
59.43	127.34
64.45	127.96
56.34	129.87
69.9	124.15

Appendix B: Algorithm for entire system

```
#Traffic Light system
#@author: Jude Asare Donkor
#Department of Computer Engineering
#Ashesi University
#Final Capstone Project

#!/usr/bin/python
import RPi.GPIO as GPIO
#from threading import Thread
import mysql.connector
import time
import signal
import sys
#import threading
import multiprocessing
GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)
#Global variables
red1 = 11
yellow1 = 12
green1 = 13
red2 = 15
yellow2 = 16
green2 = 18
# red3 = 19
# yellow3 = 21
# green3 = 22
b = 0
k = 0
lanes = 0
lanes1 = 0
intial_distance = 0
intial_distance2 = 0
final_distance = 0
final_distance2 = 0
total_cars_lane1 = 0
total_cars_lane2 = 0
total_cars_lane3 = 0
total_cars = 0
intime = 0
green_time = 0
TRIG1 = 23
ECHO1 = 24
TRIG2 = 8
ECHO2 = 10
TRIG3 = 26
ECHO3 = 19
TRIG4 = 22
ECHO4 = 21

#Setup for traffic light
GPIO.setup(red1,GPIO.OUT)
GPIO.setup(yellow1,GPIO.OUT)
GPIO.setup(green1,GPIO.OUT)
GPIO.setup(red2,GPIO.OUT)
GPIO.setup(yellow2,GPIO.OUT)
GPIO.setup(green2,GPIO.OUT)

#7 segment display
```

```

GPIO.setup(29,GPIO.IN)#ppush button1 pin
GPIO.setup(26,GPIO.IN)#push button2 pin
GPIO.setup(31,GPIO.OUT)
GPIO.setup(32,GPIO.OUT)
GPIO.setup(33,GPIO.OUT)
GPIO.setup(35,GPIO.OUT)
GPIO.setup(36,GPIO.OUT)
GPIO.setup(37,GPIO.OUT)
GPIO.setup(38,GPIO.OUT)
GPIO.setup(40,GPIO.OUT)
button_state = 0 #GPIO.input(29)

# sending to database function
def sendtodb():
    mydb = mysql.connector.connect(
        host="192.168.8.114",
        user="raspi",
        passwd="raspi",
        database="traffic"
    )
    mycursor = mydb.cursor()
    lane1 = lane1_checkcars()
    lane2 = lane2_checkcars()
    TT_lane1 = total_cars_lane1
    TT_lane2 = total_cars_lane2
    sql_query = "INSERT INTO trafficLanes (Lane1, Lane2, TT_lane1, TT_lane2)
VALUES(%s,%s,%s,%s)"
    val =(lane1,lane2,TT_lane1,TT_lane2)
    mycursor.execute(sql_query,val)
    mydb.commit()
    # print(mycursor.rowcount, "record inserted.")

# function to check pushbutton
def buttonstate():
    button_state = GPIO.input(29)
    if button_state == 1:
        button_state = 1
    return button_state

#ultrasonic sensor functions
def ultrasonic1():
    GPIO.setup(TRIG1,GPIO.OUT)
    GPIO.setup(ECHO1,GPIO.IN)
    # GPIO.output(TRIG1, False)
    # time.sleep(0.00002)
    GPIO.output(TRIG1, True)
    time.sleep(0.02)
    GPIO.output(TRIG1, False)
    while GPIO.input(ECHO1)==0:
        pulse_start1 = time.time()
    while GPIO.input(ECHO1)==1:
        pulse_end1 = time.time()
    pulse_duration1 = pulse_end1 - pulse_start1
    distance1 = pulse_duration1 * 17150
    distance1 = round(distance1, 2)
    time.sleep(0.02)
    return distance1
def ultrasonic2():
    # GPIO.output(TRIG2, False)
    # time.sleep(0.00002)
    GPIO.setup(TRIG2,GPIO.OUT)
    GPIO.setup(ECHO2,GPIO.IN)
    global pulse_end2
    GPIO.output(TRIG2, True)

```

```

time.sleep(0.02)
GPIO.output(TRIG2, False)
while GPIO.input(ECHO2)==0:
    pulse_start2 = time.time()
while GPIO.input(ECHO2)==1:
    pulse_end2 = time.time()
pulse_duration2 = pulse_end2 - pulse_start2
distance2 = pulse_duration2 * 17150
distance2 = round(distance2, 2)
time.sleep(0.02)
return distance2
def ultrasonic3():
    # GPIO.output(TRIG3, False)
    # time.sleep(0.00002)
    GPIO.setup(TRIG3,GPIO.OUT)
    GPIO.setup(ECHO3,GPIO.IN)
    GPIO.output(TRIG3, True)
    time.sleep(0.02)
    GPIO.output(TRIG3, False)
    while GPIO.input(ECHO3)==0:
        pulse_start3 = time.time()
    while GPIO.input(ECHO3)==1:
        pulse_end3 = time.time()
    pulse_duration3 = pulse_end3 - pulse_start3
    distance3 = pulse_duration3 * 17150
    distance3 = round(distance3, 2)
    time.sleep(0.02)
    return distance3
def ultrasonic4():
    # GPIO.output(TRIG4, True)
    # time.sleep(0.00002)
    GPIO.setup(TRIG4,GPIO.OUT)
    GPIO.setup(ECHO4,GPIO.IN)
    GPIO.output(TRIG4, True)
    time.sleep(0.02)
    GPIO.output(TRIG4, False)
    while GPIO.input(ECHO4)==0:
        pulse_start4 = time.time()
    while GPIO.input(ECHO4)==1:
        pulse_end4 = time.time()
    pulse_duration4 = pulse_end4 - pulse_start4
    distance4 = pulse_duration4 * 17150
    distance4 = round(distance4, 2)
    time.sleep(0.02)
    return distance4

# function to check vehicle presence and acquisition of number of vehicles
def lane1_checkcars():
    global total_cars_lane1
    lane1_cars = 0
    initial_distance = ultrasonic1()
    initial_distance2 = ultrasonic2()
    if initial_distance < 5 or initial_distance2 < 5:
        lane1_cars += 1
    total_cars_lane1 = total_cars_lane1 + lane1_cars
    return lane1_cars

def lane2_checkcars():
    global total_cars_lane2
    lane2_cars = 0
    initial_distance = ultrasonic3()
    initial_distance2 = ultrasonic4()
    if initial_distance < 5 or initial_distance2 < 5:
        lane2_cars += 1
    total_cars_lane2 += lane2_cars

```

```

        return lane2_cars

# function to allocate time for green signal
def lane1_time():
    intime = lane1_checkcars()
    if (intime <= 10):
        intime = intime * 7
        green_time = intime
    else:
        green_time = 7*10
    return green_time

def lane2_time():
    intime = lane2_checkcars()
    if (intime <= 10):
        intime = intime * 7
        green_time = intime
    else:
        green_time = 7*10
    return green_time

#function to deactivate red light
def lanecheck_time(lane):
    pw = 0
    if lane == 0:
        pw = 0
    else:
        pw = 1.5
    return pw

#function to intialise green light
def checkcar(lane1): #for green and yellow light
    y = True
    if lane1 == 0:
        y = False
    return y
def checkcarf(lane2):
    ty = False
    if lane2 == 0:
        ty = True
    return ty

#function to intiliase pedestrian timer
def pushbutton():
    GPIO.output(red1,True)
    GPIO.output(red2,True)
    timer()

def timer():
    time.sleep(1)
    for x in range(0,1):
        GPIO.output(33,1)#9
        GPIO.output(31,1)
        GPIO.output(38,1)
        GPIO.output(36,1)
        GPIO.output(37,1)
        GPIO.output(35,1)
        time.sleep(0.5)
        GPIO.output(33,0)#9
        GPIO.output(31,0)
        GPIO.output(38,0)
        GPIO.output(36,0)
        GPIO.output(37,0)
        GPIO.output(35,0)

```

```

time.sleep(0.5)
GPIO.output(33,1)#8
GPIO.output(31,1)
GPIO.output(38,1)
GPIO.output(36,1)
GPIO.output(37,1)
GPIO.output(32,1)
GPIO.output(35,1)
time.sleep(0.5)
GPIO.output(33,0)#8
GPIO.output(31,0)
GPIO.output(38,0)
GPIO.output(36,0)
GPIO.output(37,0)
GPIO.output(32,0)
GPIO.output(35,0)
time.sleep(0.5)
GPIO.output(33,1)#7
GPIO.output(31,1)
GPIO.output(38,1)
time.sleep(0.5)
GPIO.output(33,0)#7
GPIO.output(31,0)
GPIO.output(38,0)
time.sleep(0.5)
GPIO.output(33,1)#6
GPIO.output(37,1)
GPIO.output(38,1)
GPIO.output(36,1)
GPIO.output(32,1)
GPIO.output(35,1)
time.sleep(0.5)
GPIO.output(33,0)#6
GPIO.output(37,0)
GPIO.output(38,0)
GPIO.output(36,0)
GPIO.output(32,0)
GPIO.output(35,0)
time.sleep(0.5)
GPIO.output(33,1)#5
GPIO.output(37,1)
GPIO.output(38,1)
GPIO.output(36,1)
GPIO.output(35,1)
time.sleep(0.5)
GPIO.output(33,0)#5
GPIO.output(37,0)
GPIO.output(38,0)
GPIO.output(36,0)
GPIO.output(35,0)
time.sleep(0.5)
GPIO.output(31,1)#4
GPIO.output(38,1)
GPIO.output(37,1)
GPIO.output(35,1)
time.sleep(0.5)
GPIO.output(31,0)#4
GPIO.output(38,0)
GPIO.output(37,0)
GPIO.output(35,0)
time.sleep(0.5)
GPIO.output(33,1)#3
GPIO.output(31,1)
GPIO.output(38,1)
GPIO.output(36,1)

```

```

        GPIO.output(37,1)
        time.sleep(0.5)
        GPIO.output(33,0)#3
        GPIO.output(31,0)
        GPIO.output(38,0)
        GPIO.output(36,0)
        GPIO.output(37,0)
        time.sleep(0.5)
        GPIO.output(33,1)#2
        GPIO.output(31,1)
        GPIO.output(37,1)
        GPIO.output(36,1)
        GPIO.output(32,1)
        time.sleep(0.5)
        GPIO.output(33,0)#2
        GPIO.output(31,0)
        GPIO.output(37,0)
        GPIO.output(36,0)
        GPIO.output(32,0)
        time.sleep(0.5)
        GPIO.output(31,1)#1
        GPIO.output(38,1)
        time.sleep(0.5)
        GPIO.output(31,0)#1
        GPIO.output(38,0)
        time.sleep(0.5)
        GPIO.output(33,1)#0
        GPIO.output(31,1)
        GPIO.output(38,1)
        GPIO.output(36,1)
        GPIO.output(32,1)
        GPIO.output(35,1)
        time.sleep(0.5)
        GPIO.output(33,0)#0
        GPIO.output(31,0)
        GPIO.output(38,0)
        GPIO.output(36,0)
        GPIO.output(32,0)
        GPIO.output(35,0)
        time.sleep(0.5)
        GPIO.output(40,1)
        time.sleep(0.5)
        GPIO.output(40,0)

#function for traffic coordination
def trafficcoordination():
    #red1,yellow1,green1
    b = buttonstate()
    GPIO.output(red1,checkcarf(lane1_time))
    GPIO.output(green1,checkcar(lane1_time()))
    time.sleep(lane1_time())
    # buttoncheck()
    b = buttonstate()
    GPIO.output(green1,False)
    GPIO.output(yellow1,checkcar(lane1_time()))
    time.sleep(lanecheck_time(lane1_time()))
    # buttoncheck()
    b = buttonstate()
    GPIO.output(yellow1,False)
    GPIO.output(red1,True)
    sendtodb()
    GPIO.output(red2,checkcarf(lane2_time()))
    GPIO.output(green2,checkcar(lane2_time()))
    time.sleep(lane2_time())
    # buttoncheck()

```



```

b = buttonstate()
GPIO.output(green2,False)
GPIO.output(yellow2,checkcar(lane2_time()))
time.sleep(lanecheck_time(lane1_time()))
b = buttonstate()
GPIO.output(yellow2,False)
GPIO.output(red2,True)
sendtodb()
b = buttonstate()
time.sleep(1)
if b == 1:
    pushbutton()

#main
if __name__ == '__main__':
    while True:
        GPIO.output(red2,True)
        GPIO.output(red1,True)
        buttonstate()
        trafficcoordination()
        #sendtodb()

```

Appendix C: Priority vehicle code Node MCU

```
//Priority Vehicle code
//@author: Jude Asare Donkor
//Department of Computer Engineering
//Ashesi University
//Final Capstone Project
#include <ESP8266WiFi.h>
#include <ESP8266WiFiMulti.h>
#include <ESP8266HTTPClient.h>
#include <WiFiClient.h>
ESP8266WiFiMulti WiFiMulti;
#include <SoftwareSerial.h>
#include <TinyGPS++.h>
SoftwareSerial gpsSerial(4,5); //txpin- pin8 rxpin - pin9
TinyGPSPlus gps;
float latitude,longitude;

void setup() {
  //Serial.begin(115200);
  gpsSerial.begin(9600);
  Serial.begin(9600);
  Serial.println();
  Serial.println();
  Serial.println();
  for (uint8_t t = 4; t > 0; t--) {
    Serial.printf("[SETUP] WAIT %d...\n", t);
    Serial.flush();
    delay(1000);
  }
  WiFi.mode(WIFI_STA);
  WiFiMulti.addAP("Jd", "Peakab00");//wifi and wifi password
}

void loop() {
while (gpsSerial.available())
{
  int data = gpsSerial.read();
  if (gps.encode(data))
  {
    latitude = (gps.location.lat());
    longitude = (gps.location.lng());
    if ((WiFiMulti.run() == WL_CONNECTED)) {
      WiFiClient client;
      HTTPClient http;
      Serial.print("[HTTP] begin...\n");
      //url to send data
      if (http.begin(client,
"http://192.168.8.114/Gps/insert.php?insert&Gps_latitude="+String(latitude)+"&Gps_longitude="+String(longitude)+"&Vehiclename=Ambulance"" ) { // HTTP
//"http://localhost/Gps/insert.php?insert&Gps_latitude="+String(latitude)+"&Gps_longitude="+String(longitude)+"&Vehiclename=Ambulance""")
      Serial.print("[HTTP] GET...\n");
      // start connection and send HTTP header
      int httpCode = http.GET();
      // httpCode will be negative on error
      if (httpCode > 0) {
        // HTTP header has been send and Server response header has been handled
        Serial.printf("[HTTP] GET... code: %d\n", httpCode);
```

```

        // file found at server
        if (httpCode == HTTP_CODE_OK || httpCode == HTTP_CODE_MOVED_PERMANENTLY)
    {
        String payload = http.getString();
        Serial.println(payload);
    }
    } else {
        Serial.printf("[HTTP] GET... failed, error: %s\n",
http.errorToString(httpCode).c_str());
    }
    http.end();
    } else {
        Serial.printf("[HTTP] Unable to connect\n");
    }
    }

    Serial.print ("latitude: ");
    Serial.println (latitude);
    Serial.print ("longitude: ");
    Serial.println (longitude);
    }
    }
    //delay(10000);
}

```