UNIVERSITY OF NAIROBI

SCHOOL OF COMPUTING AND INFORMATICS

DESIGN OF DIGITAL DYNAMIC SPEED GOVERNOR FOR PUBLIC SERVICE VEHICLES IN KENYA

BY:

MUTHOKA NICHOLUS KIMALI

P53/78981/2015

SUPERVISORS:

PROF. WILLIAM OKELO-ODONGO

PROF. OBOKO ROBERT OBWOCHA

A PROJECT PROPOSAL SUBMITTED TO THE UNIVERSITY OF NAIROBI IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF MASTER OF SCIENCE DEGREE IN DISTRIBUTED COMPUTING TECHNOLOGY.

NOVEMBER, 2022.

DECLARATION

This proposal is my original work and, to the best of my knowledge, has not been presented for a degree award at this or any other university.

Signature: ……[illegible]…… Date: 07 December 2022 …………………………..

Name: Muthoka Nicholus Kimali

This proposal has been submitted to the School of Computing and Informatics for examination with our approval as university supervisors.

Signature:… [illegible] Date: …7 Dec 2022……..

Name: Prof. Oboko Robert Obwocha

DEDICATION

This work is dedicated to my beloved wife and son. Their compassion, understanding, and perseverance greatly inspired me.

ACKNOWLEDGEMENT

I thank the Almighty God for granting me health, wisdom, grace, and compassion during all phases of this project. My deepest appreciation goes to my supervisors, Prof. William Okelo-Odongo and Prof. Oboko Robert Obwocha, for agreeing to help me through this project. I appreciate the guidance and advice you provided in this project's development. I also want to thank all of my lecturers for the great help they have given me throughout this academic journey. Finally, I am grateful for the support I got from Dalcom Kenya Limited, Saferider Systems Kenya Limited, and Gag Technologies Limited concerning the speed governor technologies employed in Kenya and the links to NTSA and KEBS.

ABSTRACT

The increase in road accidents on Kenyan roads led NTSA to introduce digital speed governors in public service vehicles. Several companies have created various designs of speed limiters to help alleviate this problem. However, statistics from NTSA show that there is an increase in the number of road accidents and fatalities. According to the NTSA road crash reports for 2019/2020, there was a 5.8% increase in fatalities as of October 31st, 2020, which is a clear indication that despite the adoption of speed limiters, there is a major increase in fatalities and casualties. The existing speed limiters only limit the vehicle's speed at 80KM/H. This means that a vehicle may still be moving at 80KM/H in an area designated for a lower speed limit. A good example would be in townships, where the speed limit should not exceed 30KM/H. Therefore, this is a significant risk and creates a high possibility of an accident happening. This research study focuses on how to use various technologies including GPS, GPRS, and MQTT to allow the speed governor to limit the speed dynamically according to location. A GPS receiver will get the location of the device, GPRS will connect to the cloud and download speed limit information, and MQTT will function as the communication protocol between the device and the server. In this study, the vehicle’s speed sensor will detect the speed of the vehicle. The sensor outputs a square wave signal at different frequencies dependent on the speed of the vehicle. Speed limiting will be done using the Electronic Throttle Control mechanism. This study will review and evaluate existing speed limiters used in Kenya. In the review, the study will outline the shortcomings of the existing speed limiters. This study will conduct an evaluation of the dynamic speed limiter that will be prototyped and the results will be documented.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AMQP	Advanced Message Queuing protocol
API	Application Programming Interface
CAN	Control Area Network
ECM	Engine Control Module
ETC	Electronic Throttle Control
GPS	Global Positioning Systems
ICT	Information and Communications Technology
IP	Internet Protocol
KM/H	Kilometres per hour
LAMP	Linux, Apache, MySQL, PHP/Perl/Python
MQTT	Message Queuing Telemetry Transport
NTSA	National Transport and Safety Authority
PWM	Pulse Width Modulation
SSH	Secure Shell
UART	Universal Asynchronous Receiver/Transmitter
UPU	Universal Policing Unit

DEFINITION OF IMPORTANT TERMS

- **Speed Governor/Speed Limiter: -** A device for measuring and regulating the speed of a vehicle.
- **Overspeed: -** Speed greater than the rated speed.
- **Speed Limit: -** Maximum speed that a vehicle is allowed to travel at a particular stretch of road.
- **Speed Limiting: -** Measuring and regulating the speed of a vehicle using a pre-defined speed limit.
- **Dynamic Speed Limiting: -** Measuring and regulating the speed of a vehicle using a speed limit, which is variable and location-based.
- **Geo-fence: -** A virtual geographical boundary.

CHAPTER 1: INTRODUCTION

1.1 BACKGROUND

Speed limiting in vehicles has been widely deployed in efforts to reduce road accidents. Digital speed governors have replaced mechanical speed governors in modern vehicles. The digital speed governor works by receiving signals from a series of sensors that detect how fast the vehicle is going and then manages nearly all the engine's functions. Once a predetermined top speed is reached, the speed governor steps in and restricts the flow of air and fuel to the engine and even the sparks that cause combustion. This means that a predetermined top speed is not exceeded. In dynamic speed limiting, the predetermined top speed is not always a fixed value. The value of the predetermined top speed varies depending on the location and human activities. In most townships in Kenya, the speed limit is set at 50KM/H while in some areas within the Central Business District, the speed limit is set to 30KM/H. The speed governor needs to be aware of such speed limit variations and dynamically be able to limit the vehicle’s speed at the predetermined set speed of the location of the vehicle. Human activities such as marathons, demonstrations, constructions, and gatherings may also affect the speed limit. Although these activities are temporary, the speed limit becomes critical to observe. This calls for the speed governor to be aware of such activities and limit speeds as per the required set speed. Dynamic speed limiting therefore requires that the speed governor be aware of the location and any human activities that may cause speed limit variations and be able to restrict the vehicle within the speed limits.

The proposed system works by first preloading speed limit data into the online system. The preloaded data includes polygons of the locations where the speed limit applies and the corresponding speed limits. This information may be updated anytime, and the changes pushed into the speed governor device using MQTT protocol. GNSS receiver will be used in the device to get the position of the vehicle. Once the position of the vehicle is uploaded onto the cloud, the cloud system will check the speed limit for that position and send the speed limit information back to the speed governor device. The speed governor will then compare the speed limit value received with the speed data computed from the speed sensor and hence restrict the vehicle to the speed limit.

In case of human activities, there will be an online portal from which the administrator will be able to set the speed limits of roads and update the details on to the server which will in turn update the mapped speed limit data then push this information to the speed governor devices. The speed governor will limit speed using the new speed limit updated into the system.

In Kenya, Speed Governors have been used to help reduce the number of road accidents in Public Service Vehicles. This was an initiative taken by NTSA after there was a major increase in road accidents. The Kenya Bureau of Standards published the speed governor standards which were to be used in designing and developing the speed limiters. Several companies in Kenya designed and developed the speed limiters according to the standards. The speed limiters were widely adopted by the public service vehicle owners. However, there has been an increase in the number of vehicle accidents and fatalities according to NTSA Road crash reports 2019/2020. There has also been reports of over speeding cases in different regions in Kenya. Sample reports on “daily speed cases with existing regional cameras" by NTSA show that on the 4th of June 2016 and 5th of June 2016, the number of speed cases reported were 163 and 142 respectively. The reported number of accident victims that have been involved in fatal accidents during the 4th and 5th of June are 37 and 40 respectively. The rise in the number of accidents has been ongoing even though the Kenyan law requires that all public service vehicles be fitted with a speed governor. In the year 2015, the final road safety status report from NTSA shows that high speed is one of the major reasons for accidents. The report also indicates that most traffic crashes occur between 1700HRS and 2200HRS with the peak being at 2000HRS. More than 50% of accidents in Kenya happen between Friday and Sunday. The main reason given for the high number of accidents is high travel numbers, high speed and drunk driving. Over speeding has been categorized as a behavioural aspect challenge by NTSA. One of the ways in which NTSA is looking into solving the problem is by leveraging ICT by having a mobile policing gadget and a UPU.

One of the ways that can be used to solve this problem is using a speed governor that limits the speeds according to the location of the vehicle. Since the existing speed governors limit the speed at only 80KM/H, that makes them unsuitable to be used in many regions. Therefore, this study will look at some of the technologies that may be employed to help curb this problem.

1.2 PROBLEM STATEMENT

Over speeding has been categorized as one of the major causes of road accidents in Kenya. Even with the adoption of speed limiters, the rate of accidents is still rising. The existing speed limiters are not able to limit speeds according to locations. NTSA has employed more efforts in erecting speed signs along Kenyan roads but this has not helped lower the rate of road accidents. Several technologies have been used in the design of the existing speed limiters including powering off the fuel pump to lower the speed without affecting normal operation of the engine. However, this has not addressed the problem of over-speeding because vehicles may still move at 80KM/H in areas designated for 30KM/H. With this speed, there is a high likelihood of accident happening in areas that are designated for lower speeds.

This research study will seek to leverage technologies such as GPS, GPRS and MQTT to achieve dynamic speed limiting hence reduce over-speeding in areas designated for lower speeds.

1.3 OBJECTIVES

1.3.1 MAIN OBJECTIVE

The main objective of this works is to investigate how to leverage GPS, GPRS, and MQTT to implement a reliable speed limiter.

1.3.2 SPECIFIC OBJECTIVES

1. To investigate the technologies used in the design and development of the dynamic speed limiter.
2. To investigate the existing speed governors and their drawbacks.
3. To design and develop a prototype of the dynamic speed limiter.
4. To derive an evaluation of the data obtained from the study.

1.4 RESEARCH QUESTIONS

- How can GPS, GPRS, and MQTT be used in the design of the dynamic speed limiter?
- How well does this speed limiter system compare to the existing speed governors?
- What are the other possible ways to implement the dynamic speed limiting system?

1.5 SCOPE

This research study will be limited to the following:

- Investigate and demonstrate the use of GPS, GPRS, and MQTT to provide dynamic speed limiting.
- Investigate and demonstrate the existing speed limiting solutions in Kenya.
- Design and fabricate a hardware prototype of the dynamic speed limiter.
- Design and implement software to control the dynamic speed limiting system.
- Set location speed data using Google Maps.

1.6 JUSTIFICATION

This study seeks to benefit the Public Service Vehicle Industry in Kenya. The study will seek to demonstrate how dynamic speed limiting will reduce over-speeding, which is one of the major causes of road accidents in Kenya. The study will also be helpful to speed limiter manufacturers in Kenya who will benefit from the designs and evaluation by comparing the prototyped speed limiter with the existing speed limiters.

CHAPTER 2: LITERATURE REVIEW

There have been several research attempts to regulate the speed of motor vehicles in the past which have faced major drawbacks. Most speed governors use a single referenced speed limit to be able to control the speed of the vehicle. The reference speed limit cannot be adjusted, and changing it requires reconfiguring the speed governor device. Controlling the reference speed remotely makes the system more user friendly and easier to operate.

One of the issues with earlier devices is that they lack the computing power that is now available and do not anticipate the need for it. However, with more electronic controls in modern vehicles, a more powerful speed limiter is required. This research study looks at dynamic speed limiting, which is simply limiting a vehicle's speed against a predetermined set speed limit value that varies depending on location. The system proposed by this research study has two major components, which include:

1. Speed governor device: - The speed governor should be able to receive the speed limits from different locations. Additionally, the system should be able to determine the vehicle's location and speed in order to limit the vehicle's speed according to the location's speed limit.
2. Online portal: - The portal will keep updating the various speed limits and push the updated information to the speed governor. Different locations may have dynamic or static speed limits. Factors such as travel time, road construction, and social gatherings, may influence the speed changes.

2.1 WORKING PRINCIPLE OF DYNAMIC SPEED LIMITER

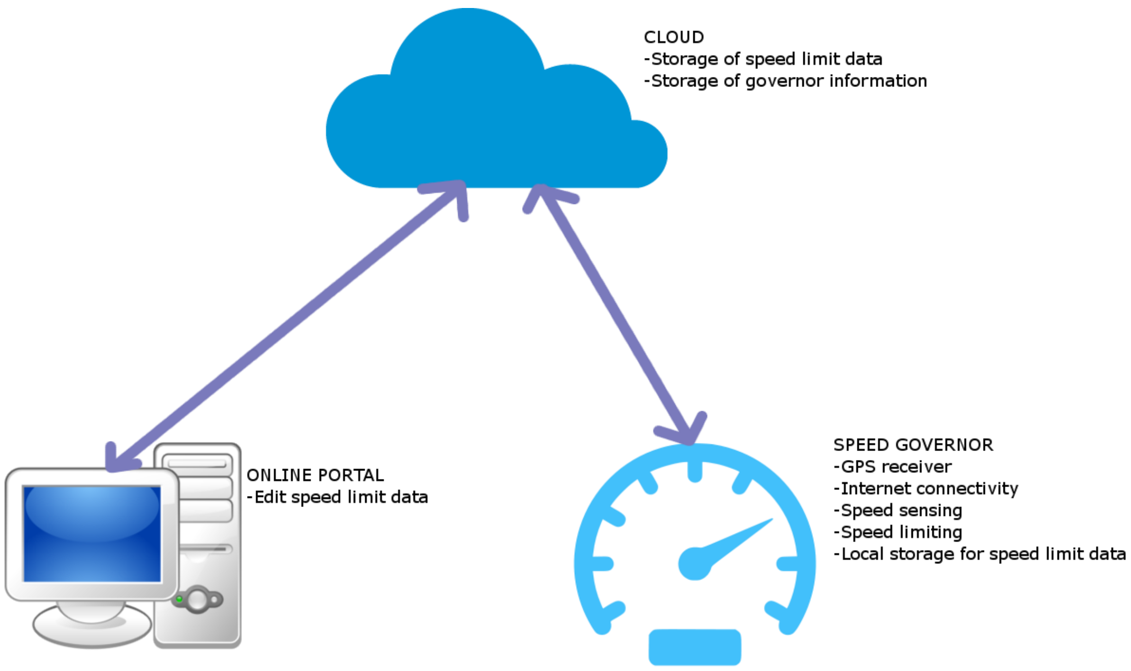


Figure 2.1: System Overview

The speed data is initially uploaded to the cloud database. This speed information can be configured by accessing the portal. The data includes geo-fences of locations and their respective speed limits. The geo-fences can range from simple rectangular shapes to complex polygons that define the coordinates of the vertices and enclose a region with a specified speed limit. Along with the coordinates of the polygons' vertices, the speed limit for the region is stored in the database.

The speed governor establishes a TCP connection with the server before using the the MQTT protocol. The device queries the GPS receiver for position information. Periodically, the device notifies the server of its GPS location via a MQTT topic. The server then verifies whether the position falls within one of the polygons stored in the database to determine the speed limit. The server will then push the speed limit for that location using MQTT protocol to a topic that the speed governor device has subscribed.

Upon receiving the speed limit information, the speed governor device compares this speed limit to the computed speed limit from the speed sensor. If the speed of the vehicle is high, then the speed governor device activates a relay that in turn isolates the gas pedal signal from the ECU and outputs a signal to the ECU that is equal to the gas pedal signal when it has not been pressed.

2.2 APPLIED TECHNOLOGIES

2.2.1 MQTT PROTOCOL

MQTT protocol employs a publish/subscribe architecture as opposed to HTTP's request/response model. Publish/Subscribe is event-driven and allows clients to receive pushed messages. The MQTT broker is the central communication point, which is responsible for routing all messages between senders and their intended recipients. Each client that sends a message to the broker must include a topic within the message. The topic is the broker's routing information. Each client that wants to receive messages subscribes to a particular topic, and the broker delivers all messages that contain the matching topic. Therefore, the clients are not required to know one another since they only communicate over the topic. This architecture enables highly scalable solutions independent of data producers and data consumers. A research study comparing the AMQP and MQTT protocols over unstable and mobile networks recommended the MQTT protocol for supporting connections with edge nodes (simple sensors/actuators) in constrained environments (low-speed wireless access) (Luzuriaga et al., 2015). In this research study, the MQTT protocol will be used for instantaneous speed limit data updates from the cloud. All speed governors must therefore be subscribed to a MQTT broker.

2.2.2 ELECTRONIC THROTTLE CONTROL

The Electronic Throttle control is also known as the drive by wire. In modern vehicles, electronic controls have largely replaced mechanical controls. The gas pedal is connected to a position sensor that is wired to the Engine Control Module in Electronic Throttle control. The position sensor measures the force exerted on the accelerator pedal. The position sensor transmits the signal to the engine control module, which then rotates the motor. The motor is connected to the throttle body's butterfly valve. The motor will either open or close the butterfly valve, which is connected to a throttle position sensor that provides feedback to the ECM (Grajkowski, 2015).

The speed governor will utilize Electronic Throttle control to limit the vehicle's speed. This will be accomplished through the installation of the speed governor between the position sensor and ECM. The speed governor monitors the position sensor and transmits the sensor's data to the ECM. When the sensor reading exceeds the value corresponding to a predetermined speed limit, the speed governor generates and transmits the data corresponding to the lower speed limit to the ECM.

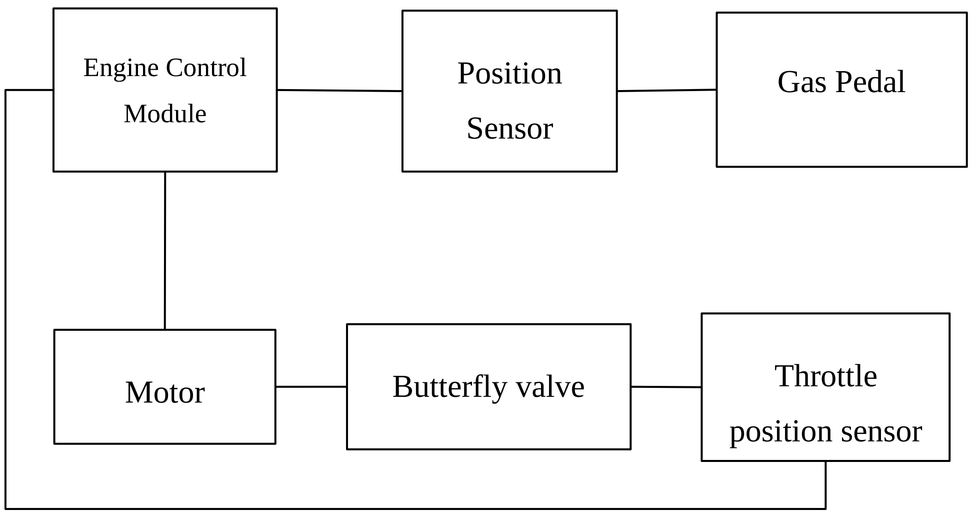


Figure 2.2: Electronic Throttle control

2.2.3 GPRS

For the speed governor device to communicate with the server, it must connect to the internet. The device can connect via cellular connection, Wi-Fi, Low power Radio frequency waves, among other ways. Since the speed limiter device is mobile and requires a constant internet connection, cellular connectivity is the preferred method of Internet access.

GPRS, 3G, 4G, and 5G are the various technologies available when using a cellular connection for the internet. General Packet Radio Service (GPRS) is a GSM-based packet-based mobile data service. GPRS enables the transport of enormous amounts of data between the device and the Internet. Since the amount of data transferred from the device is minimal and hence, GPRS will be sufficient for data transfer. The other technologies may be deployed, but they will be costly and underutilized in this research project.

2.2.4 GPS RECEIVER

The signal acquisition stage of a GPS receiver detects visible GPS satellites and provides tracking loops with a coarse estimate of the GPS signal's Doppler frequency shift and code delay (Tamazin, 2015). The function of a GPS receiver is to find at least four of these satellites, calculate their distances, and utilize this information to determine its own location. This process is based on the simple mathematical principle of trilateration. This research study will utilize a GPS receiver to determine the vehicle's exact location and acquire the speed limit for that area. The receiver's GPS coordinates will be read and then transmitted to the server.

2.2.5 CALIBRATION OF SPEED LIMITER

The speed sensor installed in a car typically generates a square wave signal at a frequency set by the vehicle's speed. At a given speed, different vehicle models create distinct frequencies. For this reason, the speed limiter device's speed signal frequency must be set to a specific speed. This enables the speed governor device to interpolate the frequency and limit the vehicle's speed when it exceeds the specified speed limit. In this research, the process of matching the speed signal frequency to the speed is referred to as the speed limiter's calibration. Calibration of the speed limiter device can occur in a variety of ways, as described below.

2.2.5.1 Calibration using GPS receiver

A study on GPS calibration methods (Bai et al., 2015) validates the use of GPS speedometers as reference equipment if at least seven satellites are accessible. In this method, the speed limiter device is installed in the vehicle. As the vehicle travels, the speed limiter device retrieves the speed data from the GPS receiver and the speed frequency from the car's speed sensor, and then matches the two data sets. This indicates that the speed limiter device will be able to determine the vehicle's speed. This approach has the disadvantage of GPS inaccuracy if it connects to fewer satellites.

2.2.5.2 Using Vehicle Speed sensor

This method involves connecting the speed limiter to the vehicle's speed sensor signal. Depending on the model of the sensor, the speed sensor typically emits square wave signals with varied amplitudes. The frequency of the square wave transmissions varies depending on the vehicle's speed. During the initial configuration of the speed governor, a speed that will be used for calibration is input into the device. The signal from the speed sensor is wired to the speed input of the speed governor. A calibration device is then attached to the speed governor's calibration input. For calibration, the vehicle is driven to the specified speed, and then the calibrator is activated to indicate that the frequency of the pulses received at the speed governor's input should match the set speed. The governor then calibrates by recording the frequency's value in a memory location. From the speed sensor, the governor uses linear interpolation for all other frequencies to determine the vehicle's speed.

2.2.5.3 Using Calibration device

This method involves a speed sensor and is similar to using a vehicle speed sensor. In this method, the calibration device emulates the vehicle speed sensor by generating pulses that are fed into the speed governor's speed input and the vehicle's speedometer. During this process, the vehicle is not operated. When these pulses are generated, the speedometer deviates in accordance with their frequency. When the calibration speed is reached, the calibrator is triggered and sends a signal to the speed governor, just like it did in the previous system, which utilized the vehicle's speed sensor.

2.2.6 CLOUD SERVER ARCHITECTURE

All speed data information is stored and retrieved by devices from the cloud server. The architecture of cloud servers is illustrated in Figure 2.3 below.

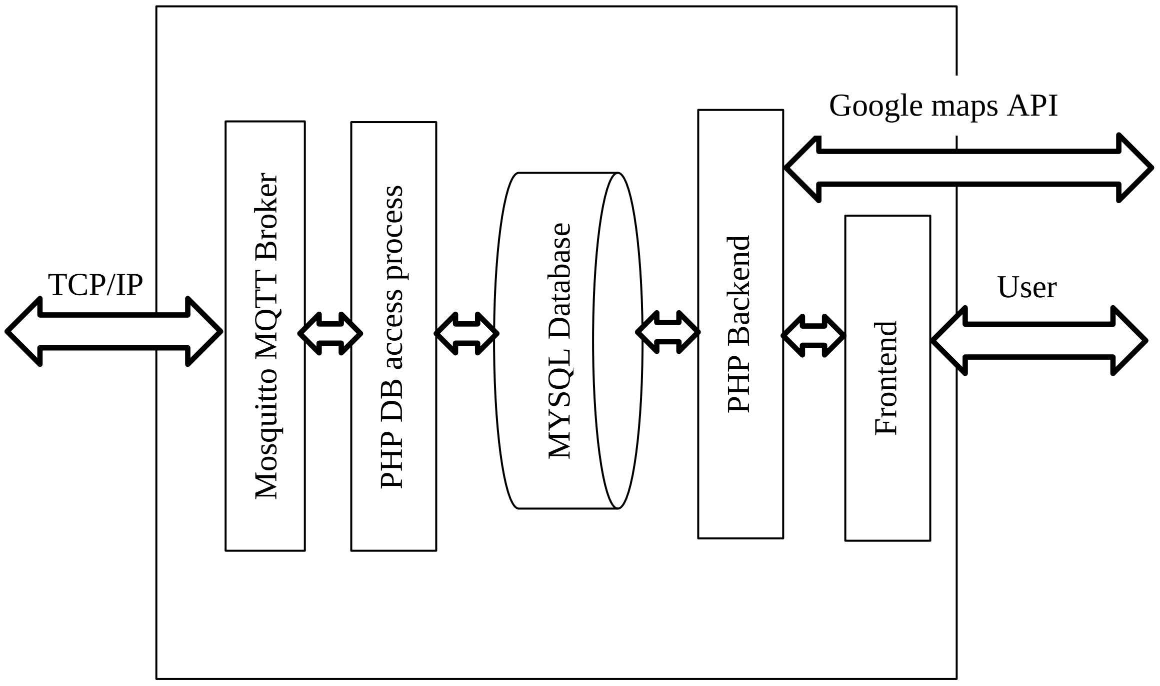


Figure 2.3: Cloud server architecture

2.2.6.1 Mosquitto MQTT broker

Eclipse Mosquitto is an open source (EPL/EDL licensed) message broker that supports MQTT version 5.0, 3.1.1, and 3.1. Mosquitto is lightweight and compatible with all devices, from low-power single-board computers to full servers. The cloud server uses the Mosquitto MQTT broker to implement the MQTT protocol, and devices subscribe and publish to topics within the broker. The speed governor device connects to the server and sends GPS coordinates to a MQTT broker topic.

2.2.6.2 PHP DB access process

The PHP DB access process is a PHP process that subscribes to the topic where the device publishes, retrieves the speed limit data from the database, and then publishes to the topic where the device has subscribed in the broker.

2.2.6.3 MySQL Database

MySQL is an open-source Structured Query Language relational database management system. The benefits of using a MySQL database include high efficiency, reliable uptime, and low cost. This research study will store its data in a MySQL database. The database will contain speed information and the coordinates of the polygons that enclose a region.

2.2.6.4 PHP backend and frontend

This refers to the system that users access to set speed limit data for different locations. The system will use Google Maps API data to help map the regions and obtain the GPS coordinates for the polygons' vertices. These coordinates will subsequently be saved in the database alongside the speed limit information corresponding to them.

2.3 RELATED WORK

Several studies have been conducted on dynamic speed limiting, or speed limiting with GPS and GPRS capability.

1. Vennela Priyadarshni and his team conducted a research study on GPS and GSM Enabled Embedded Vehicle Speed Limiting Device (Priyadarshni et al., 2016). The purpose of the research was to develop a simple and effective method for automatically controlling the vehicle's speed. The system employs GPS positioning and compares it to a list of predefined speed limits for various locations. However, this research does not provide a platform for dynamic speed adjustment.
2. A research study by Yuji KII presents a speed limiter in which a vehicle's speed is identified and a speed limit displayed on a speed sign. In this system, the speed is detected based on a front monitoring camera's image (KII et al., 2015). A speed difference is calculated between the vehicle's speed and the speed limit, and an accelerator sensitivity gain is adjusted using the speed difference as a parameter. An accelerator opening degree is then adjusted using the accelerator sensitivity gain to

produce a pseudo accelerator opening degree. The desired throttle opening degree is determined based on the virtual accelerator opening degree and the engine's revolutions per minute. However, this technique does not fix the dynamic speed governing problem.

2.4 SPEED LIMITERS CONCEPTS

2.4.1 Speed Detection

There are various methods for detecting the speed of vehicle speed governors. The primary methods for detecting speed include GPS, Vehicle speed sensor, and CAN bus.

2.4.1.1 GPS speed detection

This method utilizes a GPS receiver module. The module receives data from multiple satellites and can provide the position, speed, and direction of the object to which it is attached, among other parameters. The major shortcoming associated with this method is that it does not work when the GPS receiver takes a long time to find the satellites or in the event of GPS outages.

2.4.1.2 Vehicle speed sensor

The signal generated by the vehicle speed sensor is a square wave. The frequency of the square wave signal varies with the vehicle's speed. The faster the speed, the higher the signal frequency. Some car speed sensors have a frequency-to-voltage converter that can serve as an analog input for the speed limiter. Using the vehicle speed sensor necessitates the calibration of the speed limiter. This is the method recommended by Kenyan standards for speed limiters, and it will be utilized in this research.

2.4.1.3 CAN bus speed data

The majority of modern automobiles are equipped with CAN bus, hence the speed can be acquired via the CAN bus. A controller and CAN transceiver module are required to get speed from CAN bus. The speed sensor of a vehicle is typically equipped with a controller and CAN transceiver module. The controller broadcasts the vehicle's speed to the CAN bus, allowing all controllers connected to the bus to receive this information. If the speed limiter is equipped with the CAN transceiver module and CAN controller, it can therefore receive the speed data.

2.4.2 Speed Limiting

There are two main methods for limiting speed.

1. Electronic Throttle Control method
2. Injection Valve control method

2.4.2.1 Electronic Valve Control speed limiting method

Modern vehicles are equipped with an Electronic Valve Control unit to determine the vehicle's speed. Typically, two or more signal wires deliver speed signals from the gas pedal position sensor to the ECU. These signals are typically analog voltages dependent on the gas pedal position. Maximum and minimum signal voltages vary based on the design of the gas pedal position sensor. Depending on the sensor model, pressing the gas pedal may also lower or enhance the signal strength.

The signals from the gas pedal position sensor to the ECU are intercepted and disconnected by an ECU switch to achieve speed control. A separate control unit transmits signals equivalent to the signals produced by the gas pedal sensor when the gas pedal is not pressed. This concept is illustrated in Figure 2.4 below.

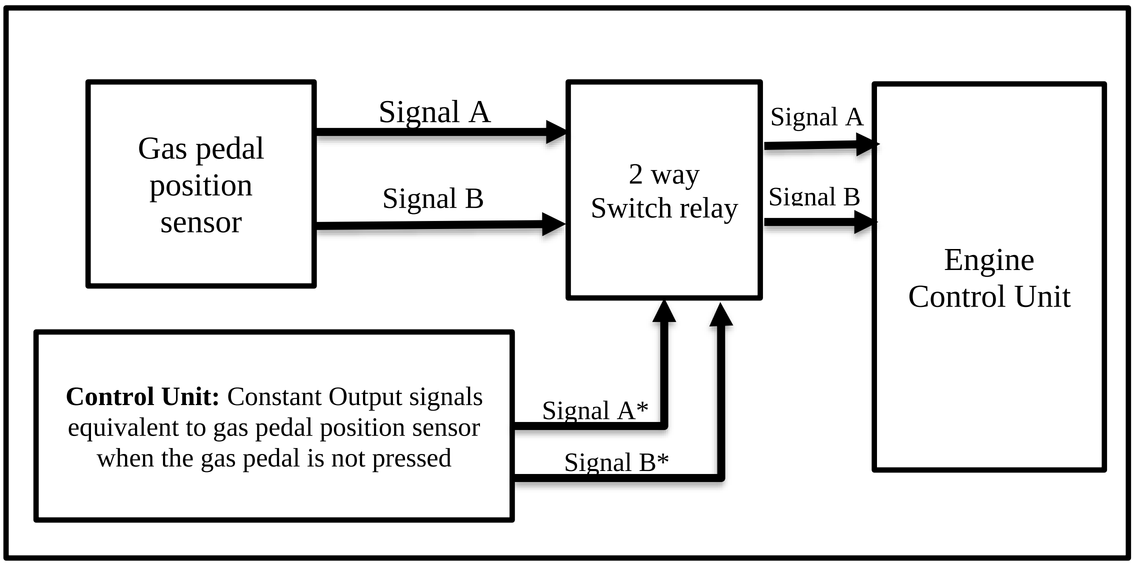

Gas pedal position sensor
Signal A
Signal B
2 way Switch relay
Signal A
Signal B
Engine Control Unit
Control Unit: Constant Output signals equivalent to gas pedal position sensor when the gas pedal is not pressed
Signal A*
Signal B*

Figure 2.4: Electronic Throttle Speed Control

The ECU receives signals from the gas pedal position sensor during normal operation. When a certain maximum speed is exceeded, the switch is actuated and disconnects the gas pedal position sensor from the ECU, therefore connecting the control unit to the ECU. When the gas pedal is not pressed, the control module outputs signals that correspond to the gas pedal position sensor signals. This is analogous to releasing the accelerator pedal. Therefore, the ECU will interpret the signals as though the gas pedal has been released and will not accelerate, gradually closing the butterfly valve in the throttle control unit and consequently slowing the vehicle.

2.4.2.2 Injection Valve speed limiting Method

Electronic Throttle control unit in older automobiles are uncommon. To limit the speed of these vehicles, the injection valve must be closed so that only a small amount of fuel flows through the valve. This process is not always seamless, as the car jerks forward when the injection valve is abruptly closed. A relay is normally activated to restrict power to the vehicle's Injection valve in order to limit the vehicle's speed using this technique. Typically, the injection valve does not fully close, allowing a little amount of fuel to flow into the engine. However, the vehicle jerks forward due to the rapid power loss at the valve.

2.4.3 Calibration

Calibration is required for speed governors, particularly when using the vehicle speed sensor. This is because the vehicle speed sensor is frequency-based, and without calibration it is impossible to determine which frequency corresponds to a given speed. For different speeds, different speed sensors use different frequencies. During installation, it is therefore essential to calibrate the speed governor. Earlier in this research study, the various calibrating techniques have been discussed.

2.4.4 Positioning

Normally, locating systems are categorized into three groups:

- **Receptive locating systems:**

Position information is ubiquitously disseminated, and mobile devices can deduce their own location from this data.

- **Transmissive Locating Systems**:

The position is determined by a fixed station that either sees or receives a signal from the mobile device (GSM positioning is a prime example).

- **Hybrid:**

Several research efforts have been done on hybrid techniques combining these principles are possible as described in a research study in Hybrid Positioning Strategy for Vehicles in a Tunnel Based on RFID and In-Vehicle Sensors (Xiong, et al., 2013).

2.4.5 Initialization

During initialization of the governor, several parameters need to be set up in the governor. These parameters include:

1. The registration number of the vehicle on which the governor is installed: - Useful for identifying the governor when data is sent to the cloud.
2. The calibration speed: - Useful when using vehicle’s speed sensor to determine the speed of the vehicle.
3. The maximum speed limit: - This also acts as a fail-safe mechanism when the governor is unable to retrieve location-based speed data.

2.4.6 Data Capture and storage

According to Kenyan requirements, speed limiters must be capable of capturing and storing data. In addition, flags should be added to the saved data when the vehicle exceeds the speed limit. The speeding records are typically the result of human behavior. Some drivers may chose to detach the governor's speed limiting unit, although the governor can still maintain the records. The storage of data in the speed governor device, as required by the speed governor standards in Kenya, is outside the scope of this research.

2.4.7 Communication

There are multiple communication options available for the speed governor. Communication with computers is essential for debugging, scripting, capturing data, and updating. This is a conventionally wired mode of communication. It could be UART, USB, SPI, or any other physical communication protocol. Communication with the server is also crucial for the governor designed in this research study. Due to their portability and configuration simplicity, cellular networks are suited for server communication. GSM/GPRS/3G modules are hence suitable for network communication.

2.5 CONSTRAINTS FOR DESIGN OF SPEED GOVERNORS

2.5.1 Standards Constraints

In June 2011, the Kenya Bureau of Standards issued specifications for the design of speed governors for motor vehicles. These specifications impose some limits on the design of the speed governors. Some of the clauses from the standard include:

1. Clause 4.1.1: The systems shall be "failsafe" such that if the plug/s to the electronic controller is/are removed, power is disconnected. Or, if the speed signal and/or wire is disconnected, the vehicle shall automatically default to a limp mode or the engine shall stall or return to idle, or, at the very least, shall not be able to exceed twenty kilometers per hour (20km/h).
2. Clause 4.1.3: The equipment shall monitor the road speed of the vehicle by being integrated to the electronic speedometer signal input, where possible or by the installation of an electro-mechanical speed sensor installed onto the speedometer cable take off at the gearbox or by use of other speed signal method as may be specified by manufacturer.
3. Clause 4.1.5: The equipment shall pre-warn the driver through a high-frequency alarm when the vehicle is almost attaining the maximum speed. This warning shall occur at

5% prior to the set speed and shall continue buzzing once the set speed is achieved/exceeded.

4. Clause 4.1.6: When the set speed is attained, the speed limiter shall temporarily cause the engine to lose power. Engine power shall be reinstated at a speed not less than the pre-warning switch point, which is 5% below the set speed.
5. Clause 4.1.12: The road speed limiter system shall be equipped with a system that shall accept an external device for testing the speed limiter and verifying its functionality.

2.5.2 Other constraints

Several other restrictions influenced the design decisions of the speed governor. These constraints include:

2.5.2.1 Cost

After reviewing the existing governors in Kenya, the cost falls within the same range as shown in the table below:

Table 2.1: Cost comparison of speed governors

GOVERNOR BRAND	MARKET COST
Omata	15,000
PGL	15,000
Pinnacle	18,000

This cost limitation also influenced the selection of components utilized in the design of the speed governors.

2.5.2.2 Form Factor

The form factor of the governor must be considered when developing the speed governor device. This is due to the extremely limited space available in the car for installing the speed governor. As indicated in the table below, the existing speed governors have a compact size:

Table 2.2: Speed governor form factor

GOVERNOR	FORM FACTOR(cm^3)
Omata	12*10*3
PGL	15*11*3
Pinnacle	10*8*4

In this research study, the form factor will not be a constraint as the study seeks to prove the concept and build a prototype rather than designing a speed governor device optimized for production.

2.6 STUDY OF THE EXISTING SPEED GOVERNORS

2.6.1 Omata Speed Governor

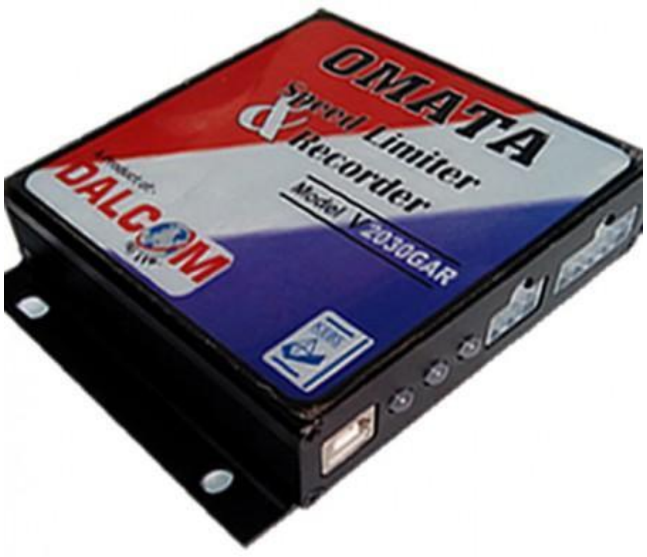


Figure 2.5: Omata speed limiter

I was provided with an Omata speed governor and instructed on its installation. First, the technician had to configure the speed limiter by inputting the vehicle's information, the speed restriction (40 KM/H), the time, and the calibration speed (40 KM/H). The governor was then ready for installation.

Initially, the governor was calibrated using a governor calibration device. The dashboard of the vehicle had to be opened in order to locate the speed signal wire that is attached to the speedometer. Once the cable was identified, it was cut to break the speedometer's connection. The square wave output of the calibration device was connected to the speedometer input and the speed input signal of the speed governor. The objective was to generate signals for operating the speedometer while also providing input for the speed governor. Once the desired calibration speed was reached, the calibration device's button was to be pressed to send a high signal to the speed governor, indicating that the frequency being received at the speed input of the speed limiter corresponds to the desired calibration speed. However, this procedure failed. The main reason behind this failure was that the voltage amplitude of the calibration device was insufficient to power the speedometer and therefore, the signal from the vehicle's speed sensor was used for calibration. In order to calibrate the vehicle, the output of the speed sensor was connected to both the governor and the speedometer. The had to be elevated using hydraulic jacks in order to calibrate it while stationary. The vehicle was started and the gas pedal was pressed. Once the calibration frequency was established, the calibration device's button was pressed to notify the governor that the process was successful.

The speed governor was then installed after calibration. The governor uses injection valves and therefore, the injection valve's connection was identified and cut. One side was attached to the relay input of the speed governor, while the other was connected to the normally closed relay output. As soon as the connection was made, the vehicle was taken out for a test drive. The speed limit had been established at 40KM/H. Once the vehicle reached 37KM/H, the governor began to beep repeatedly. At 40KM/H, the engine lost power and the vehicle slowed to approximately 35KM/H before the engine power was restored. The main challenge with the speed limiter was sudden loss of engine power during speed limiting. The governor could only limit up to the set speed and was not dynamic.

2.6.2 PGL Speed governor

The installation technique for this speed governor is similar to the Omata speed governor. However, I did not install this governor to test it. I simulated the vehicle's speed sensor with the calibrator and tested the relay output of the injection valve driver for continuity. The primary difficulty I encountered with this governor was the limited range of speed limiting, that is, when I set the speed restriction to 40KM/H, the warning beep sounded when the speedometer read 39KM/H and the speed was restricted at 40KM/H. In addition, despite having GPS and GPRS capabilities, the governor did not use dynamic speed control.

2.6.3 Pinnacle Systems speed governor

This speed governor is equipped with a GPS calibration mechanism, so there was no need to use a calibrator to calibrate it. The installation was simple because the technician simply had to configure the parameters for the governor. The biggest problem with this speed limiter was that the car kept jerking when it got close to the limit.

2.7 CONCEPTUAL DESIGN

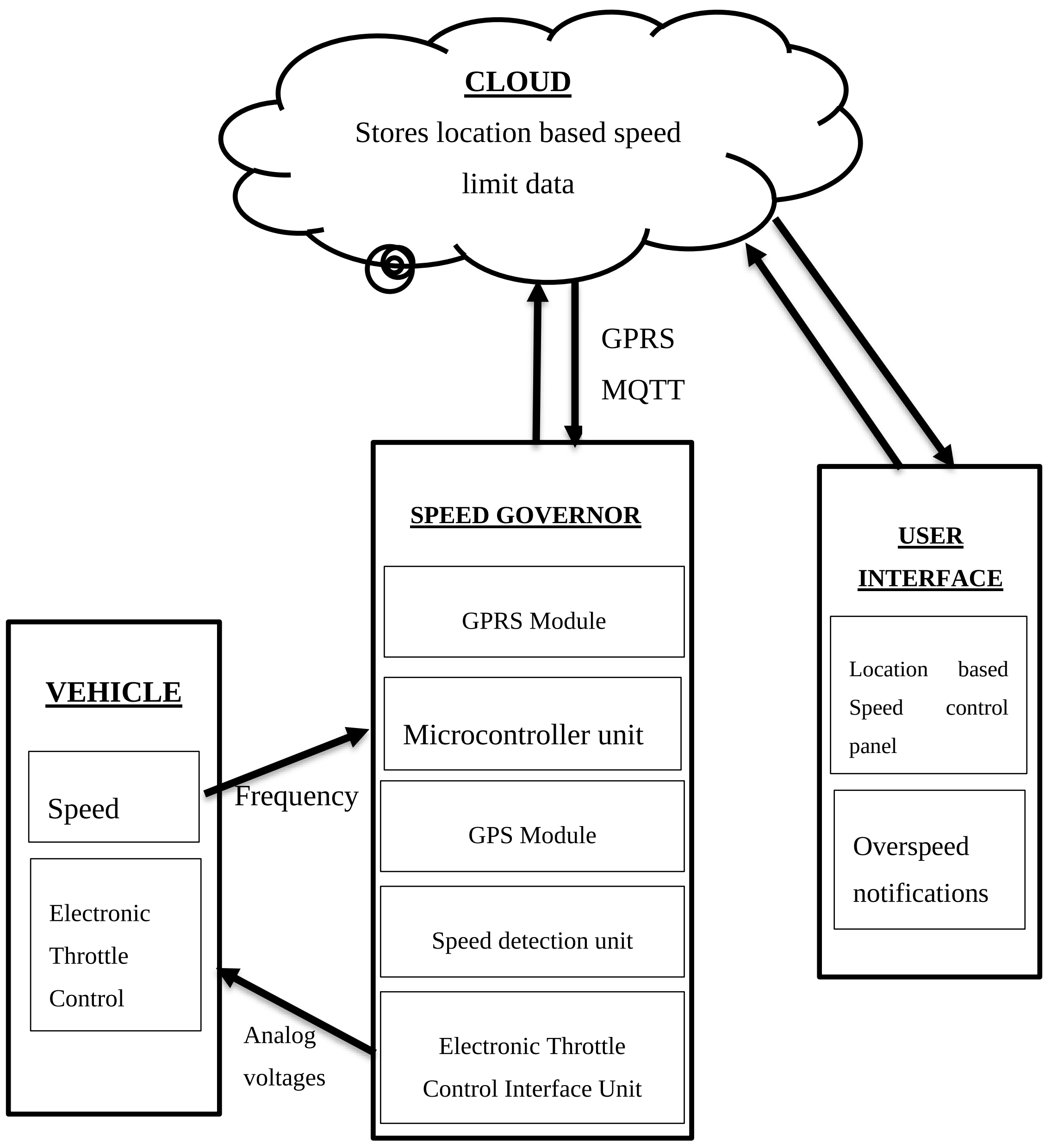


Figure 2.6: Conceptual design of the speed limiter

The concept design has four main parts as shown in Figure 2.6 above.

2.7.1 Speed Governor

The speed governor unit has five modules: a GPRS module, a Microcontroller module, a GPS module, an SD card memory module, and an Electronic Throttle Control Interface module.

2.7.1.1 GPRS Module

This module connects the speed governor unit to the cloud through cellular network. The connection is based on TCP/IP.

2.7.1.2 Micro-controller module

This is the primary control module for the governor unit. Programming allows the microcontroller to manage all inputs, outputs, calculations, and logic involved.

2.7.1.3 GPS module

This module uses a GPS receiver to get the location of the speed governor.

2.7.1.4 Speed detection Unit

This unit detects the speed signal provided by the speed sensor. It is composed of a transistor and a series resistor at the transistor's base.

2.7.1.5 Electronic Throttle Control Interface

This module is comprised of analog voltages equivalent to the idle voltages measured from the throttle position sensor of the vehicle when the vehicle is in an idle state. This unit's primary function is to reset the Electronic Throttle Control Unit to idle voltage if the speed limit has been exceeded.

2.7.2 Cloud

A server in the cloud stores location-based speed limits, which are then updated on the devices.

2.7.3 Vehicle

When the speed is above the speed limit, the car sends speed signals and is controlled by the Electronic Throttle Control.

2.7.4 User Interface

This enables the user or administrator to set location-specific speed limits at any time using a Google map interface.

CHAPTER 3: METHODOLOGY

3.1 Speed governor hardware design

After analyzing the constraints that influenced the speed governor's design specifications, a conceptual design of the speed governor was implemented as illustrated in Figure 2.6. Arduino was used as the microcontroller platform, and the rest of the circuitry was developed on top of it, as described in the following subsections.

3.1.1 Speed detection system

For detecting speed, the vehicle's speed sensor was utilized. The vehicle's speed sensor is based on frequency and emits a digital square wave signal whose frequency increases as the vehicle's speed increases. At different speeds, different car models emit different signal frequencies. In general, the frequency grows linearly for the majority of vehicles. The Toyota Wish model was used for tests during this study. The car was elevated using jacks to perform the speed test and frequency measurements while stationary. The dashboard cover was then removed and the dashboard meter ejected to uncover the connected cables as shown in Figure 3.1 below.

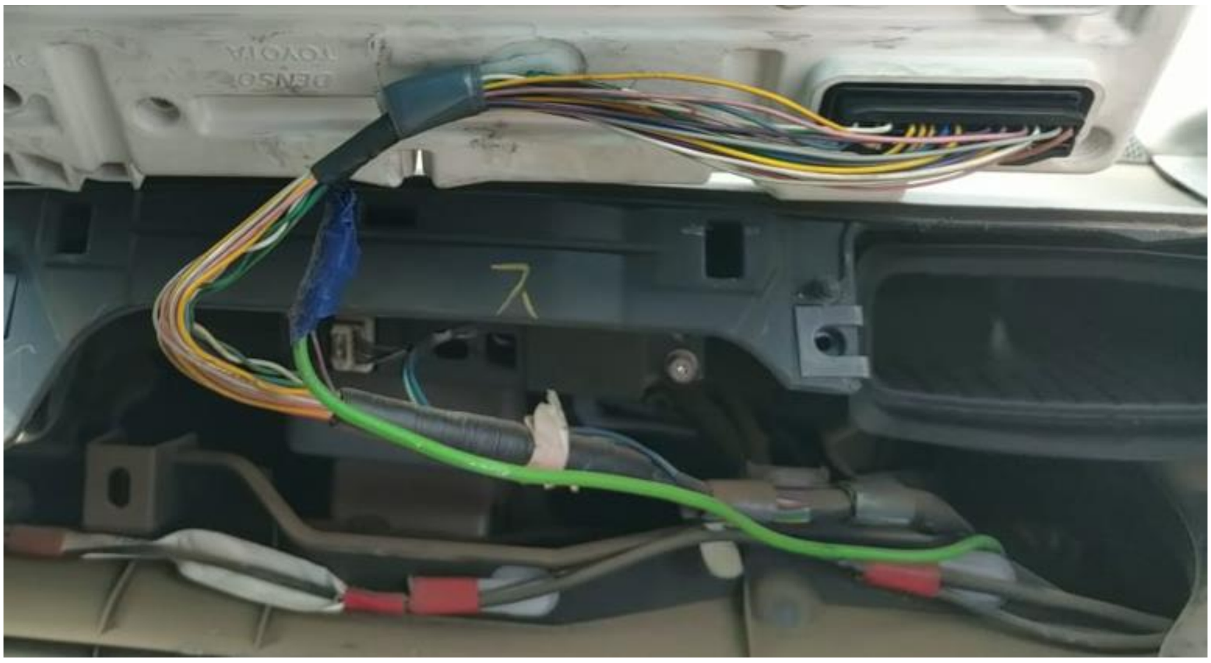


Figure 3.1: Cable connection in the speedometer

A Fluke digital multi-meter was used to determine which cable was carrying the speed signal and the frequency of the speed signal by measuring the frequencies of the signals from the various cables connected to the meter. The cables were slightly stripped to allow for the connection of the digital multi-meter lead. The ground lead of the digital multi-meter was connected to the car's body throughout this frequency measuring procedure, while the other lead was connected to each cable one at a time as shown in Figure 3.2 below.

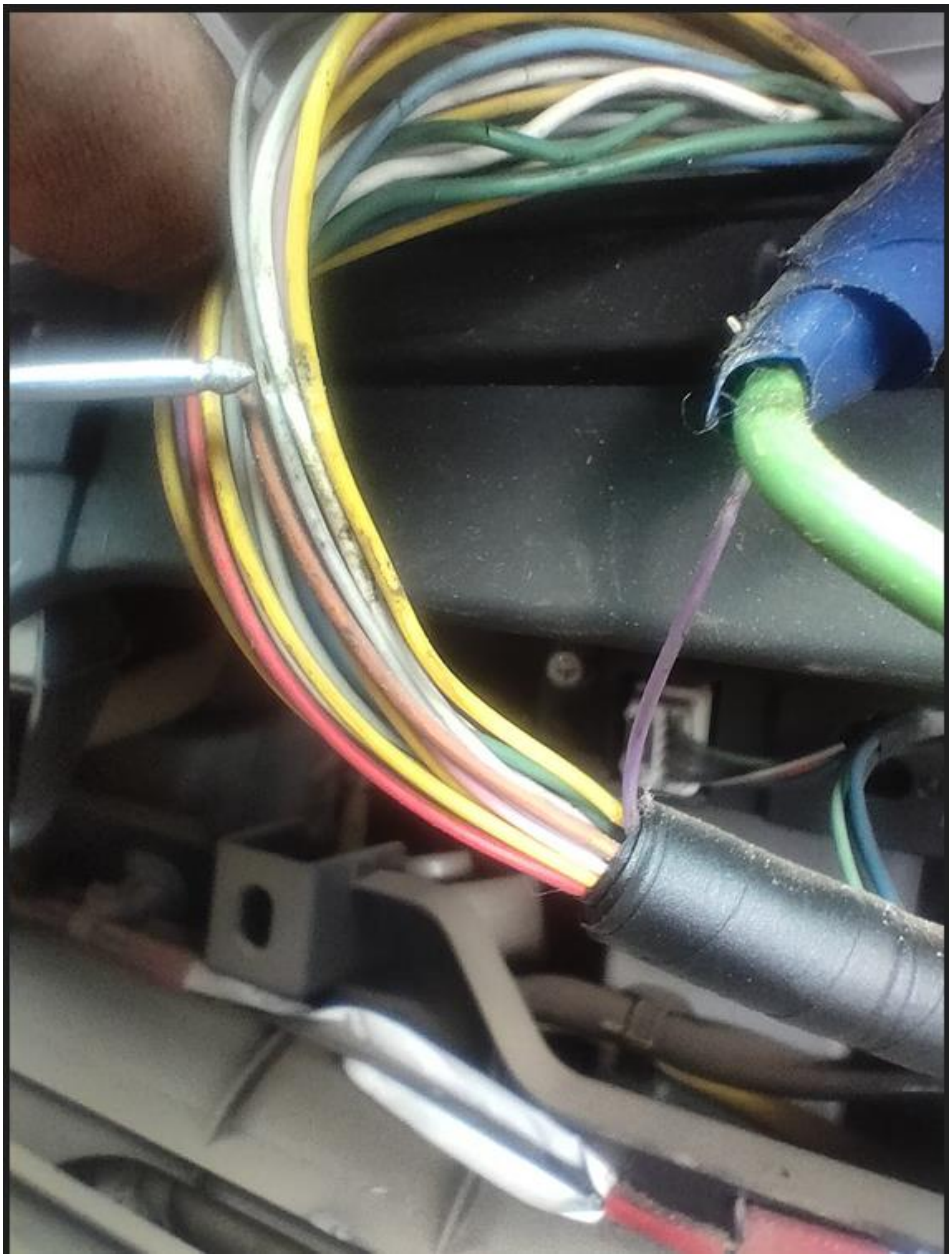


Figure 3.2: Frequency measurement using a digital multi-meter

During the speed test at 80KM/H, the majority of the cables produced a frequency reading of 0Hz, with the purple cable yielding a frequency of 53Hz. These frequency readings in the digital multi-meter are illustrated in Figure 3.3 and Figure 3.4 below.

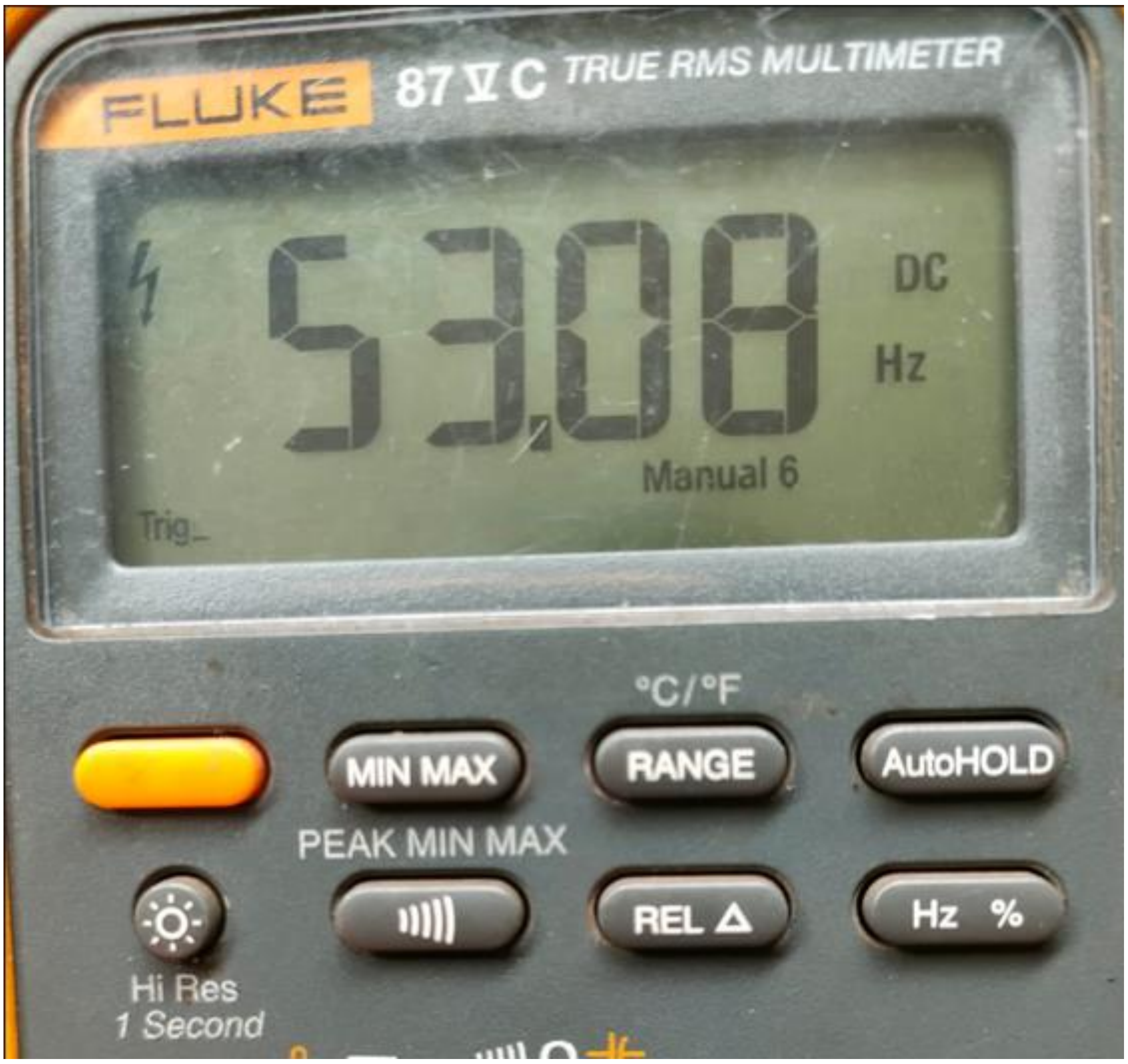


Figure 3.3: Frequency value of the purple cable during the speed test

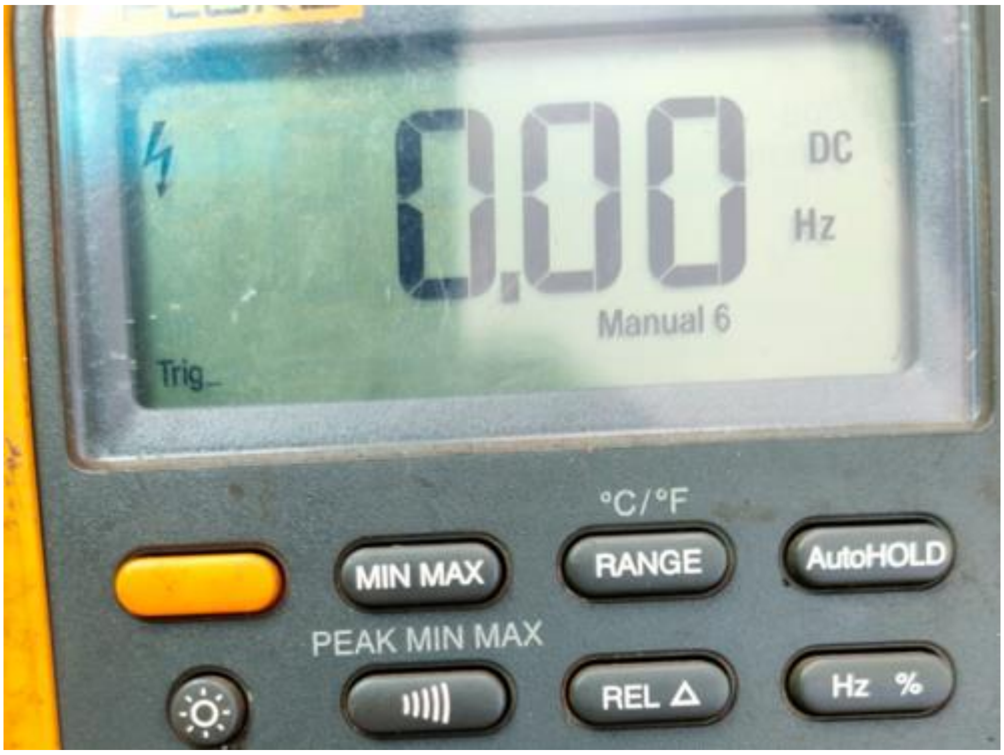


Figure 3.4: Frequency value of the other cables during the speed test

To detect the square wave frequency signal, two primary Arduino microcontroller board functions were used. The first one is the external interrupt function that ensures that all signals generated by the speed sensor are received as interrupts by the microcontroller. The second function is a timer that calculates the interval between two successive impulses.

The design utilized an Arduino digital interrupt input pin (INT0) to detect the speed signal. As indicated in the schematics below, a Darlington transistor (TIP122) was used to accommodate for the varying amplitude levels of the signals from different cars. A 1KΩ resistor was connected to the base of the transistor to reduce the signal's surplus voltage. The collector of the transistor is pulled to 5V by a 10KΩ resistor. This indicates that the logic of the speed signal received from the base of the transistor will be inverted at the collector output as shown in Figure 3.5 below. Fritzing software was used in the development of the hardware's schematic design.

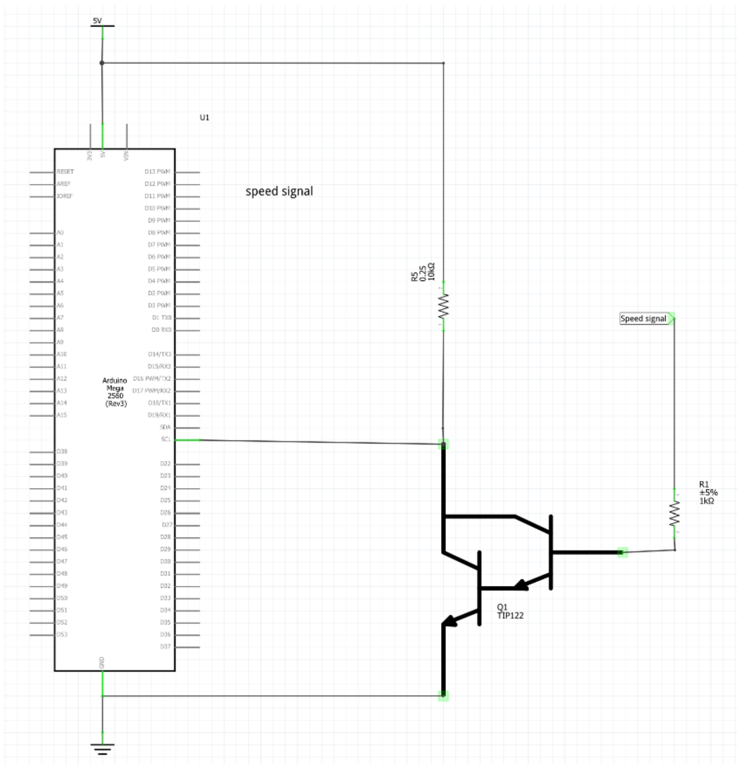


Figure 3.5: Speed detection schematic design

When the vehicle speed signal is logic LOW, the base of the transistor TIP122 will be off, and hence the transistor will act like an open circuit. This causes Arduino pin 21 (INT0) to be connected to VCC (5V) via a pull-up resistor of 10kΩ. This corresponds to connecting a logic HIGH to the interrupt pin.

When the vehicle speed signal is at logic HIGH, the base of the transistor TIP122 will be ON, forming a closed circuit that connects Arduino pin 21 (INT0) to ground through the transistor. This will result in a logic LOW being transmitted to the interrupt pin. For determining the frequency of the speed signal, either the rising or falling edge is utilized. The Arduino is set to perform an interrupt service function whenever it detects a rising edge of the speed signal. Arduino starts a timer after detecting the rising edge of the speed signal. When another signal is detected, the Arduino then identifies the frequency by computing the time difference between the two signals.

$$F(Hz) = 1/T(S)$$

Where F = frequency in Hz and T = time in Seconds

3.1.2 Speed Limiting

The Electronic Throttle Control method of regulating speed was utilized to limit speed. This was recommended by the research conducted on current speed limiters, which determined that this method did not result in jerky vehicle movement. To identify the position signal cables of the pedal, the speed signal wires from the position sensor of the gas pedal to the ECU were investigated first. This was accomplished by measuring the voltages of each cable from the pedal position sensor and pushing the gas pedal to ascertain whether the voltages change when the pedal is pressed. The vehicle type and model on which these tests were conducted is a Toyota Wish 2010 model. The results of the test are as shown in Table 3.1:

Table 3.1: Voltage levels for gas pedal sensor

Colour of the cable	Voltage at idle state	Voltage when gas pedal is pressed
Red	5V	No change
Black	5V	No change
Yellow	0.8V	Increases to 4.2V
Blue	1.6V	Increases to 5V
Green	0V	No change
Brown	4V	No change

The yellow and blue wires were determined to be the gas pedal sensor position signal wires to the ECU based on the data contained in Table 3.1 above.

In order to achieve speed limiting, the speed limitation system disconnects the position sensor of the gas pedal from the ECU and provides an analog signal with voltages equal to the initial voltages of the gas pedal sensor before pressing.

The voltage of the system is 5V. To obtain the idle voltages of 0.8V and 1.6V, the voltage was divided using potentiometers. To ensure that the voltages produced by the resistor dividers were not impacted by external resistance, the operational amplifier was configured with unity gain, as depicted in the schematic in Figure 3.6 below. A relay was used to switch the signals between the resistor divider circuits' static voltages and the gas pedal position sensor's signals.

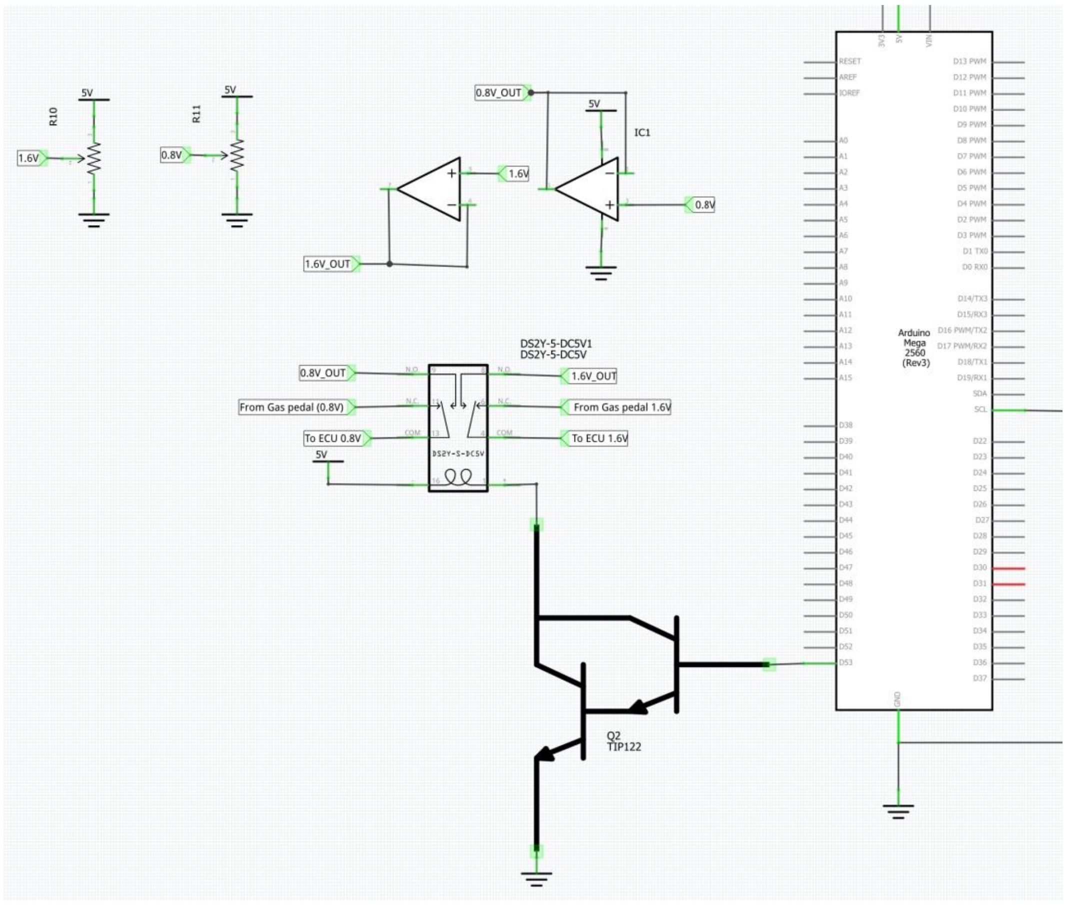


Figure 3.6: Speed limiting circuit

This circuit's speed-limiting principle is that the relay is actuated when the controller senses that the speed limit has been exceeded. The signal wires from the pedal position sensor are disconnected from the ECU and then the voltage dividers are connected to the ECU. The ECU detects the voltage as the idle voltage, so it will close the butterfly fuel valve gradually without jerky movements of the vehicle.

3.1.3 Connection to cloud

SIMCOM's SIM808 GPRS module, which communicates serially through UART was used to establish a cloud connection. According to the module's datasheet, AT commands were used to communicate with the module. The TX pin of the module was linked to Arduino's RX1 and its RX pin was attached to Arduino's TX1 pin. The 5V Step down voltage module was used to supply power to the module. The communication baud rate between SIM808 and Arduino was set to 115200. The connection is as shown in the circuit in Figure 3.7 below.

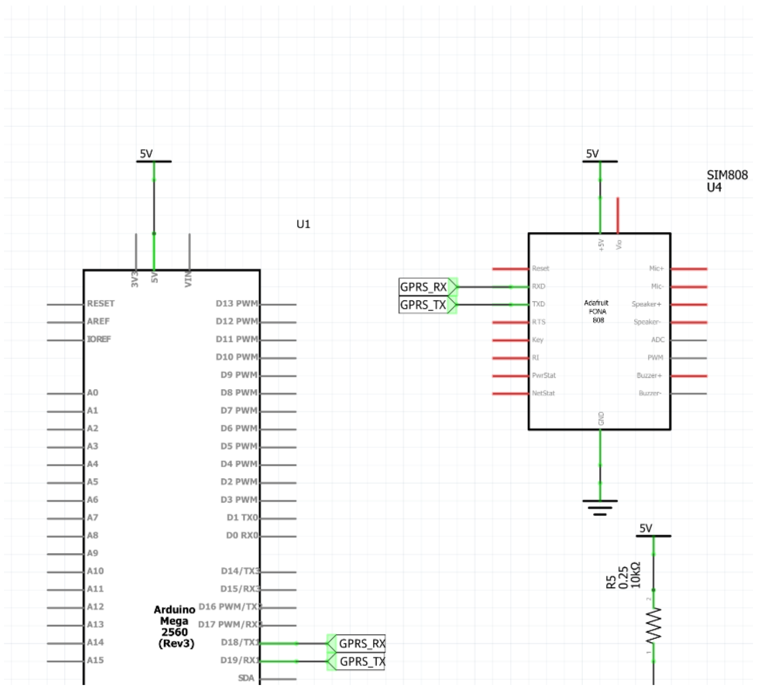


Figure 3.7: SIM808-Arduino connection

3.1.4 Positioning

As the GNSS positioning module, a SIM808 module with serial UART communication was used. This module incorporates both GPRS and GNSS technologies. The GPS antenna was connected to the module and AT commands were used to retrieve latitude and longitude coordinates. The procedure is described in detail in the firmware development section below.

3.2 Speed Governor Firmware design

The firmware design of the dynamic speed governor device includes all of the modules connected to it. These include:

- Speed detection unit
    - Receives square wave signals from the vehicle’s speed sensor.
- Speed limiting unit
    - Uses relays to connect calibrated voltages to the ECU and disconnect the position sensor of the gas pedal from the ECU.
- GNSS unit
    - Acquires the GPS coordinates.
- GPRS unit
    - Communicates to the cloud using MQTT.

The flow chart in Figure 3.8 describes the firmware design of the dynamic speed limiter.

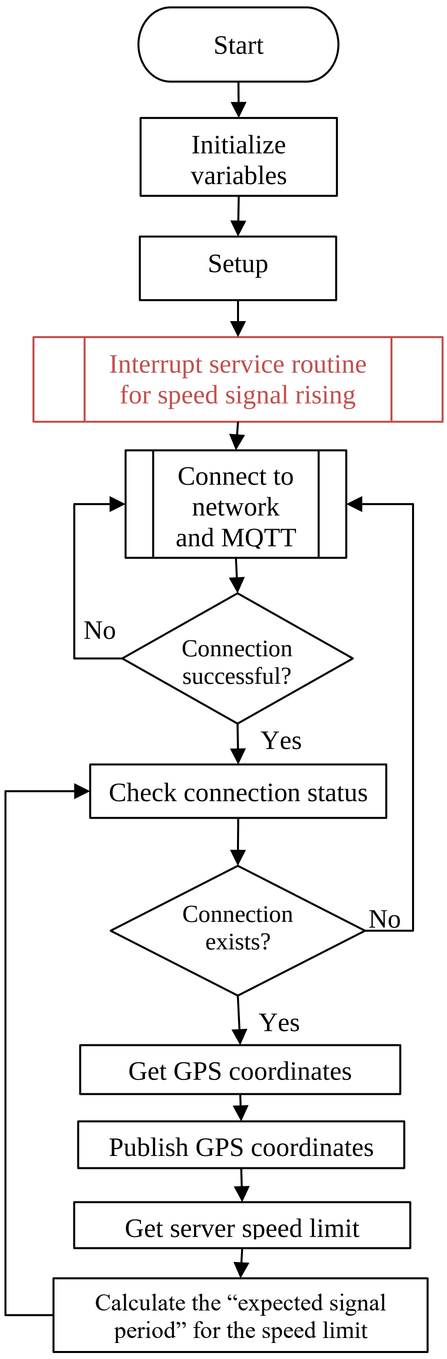


Figure 3.8: Firmware design flowchart

The flowchart in Figure 3.9 describes the Interrupt service routine for the speed signal. This operation interrupts the primary process when a signal from the speed sensor is received.

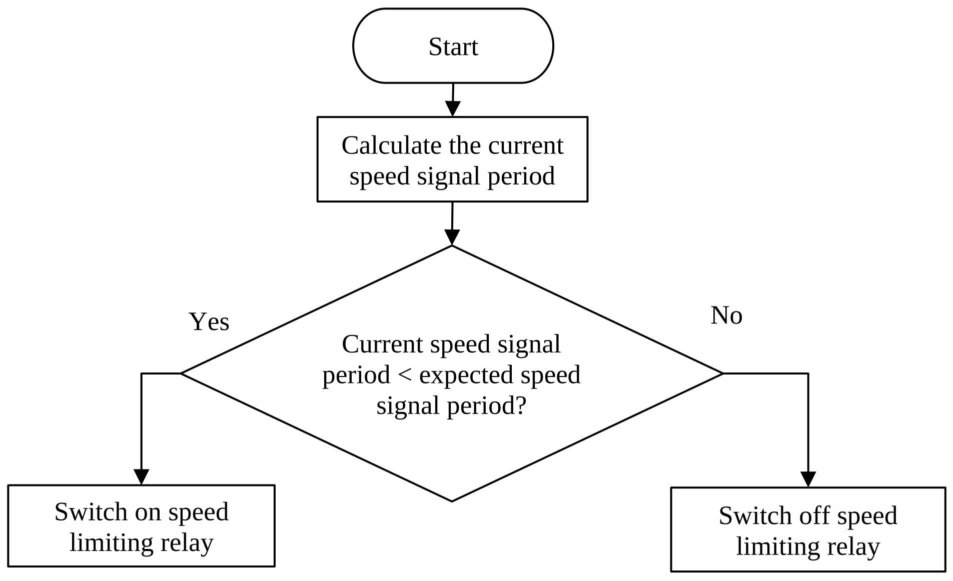

Start
Calculate the current speed signal period
Current speed signal period < expected speed signal period?
Yes
No
Switch on speed limiting relay
Switch off speed limiting relay

Figure 3.9: Interrupt service routine for speed signal

The function of limiting speed occurs within the interrupt service routine. This ensures that the speed-limiting task will always execute even when the primary function is executing other activities. This is necessary because the speed-limiting function must occur in real-time, as opposed to within the main function, which may have delays.

3.3 Online Cloud Platform Design

A Virtual Private Server (VPS), obtained from Linode LLC, was used to implement the Cloud platform. The first step in the Linode website was creating an account and then logging into the new account. After logging into the new account, the next step was to create a Linode and provision a server, where a server with the IP address 139.177.181.188 was provisioned. Before obtaining the SSH access from the created Linode, the status of the Linode has to turn from the provisioning status, to the booting status, and then finally to the running status to indicate that the provisioning was successful. Once the status turned to running as shown in Figure 3.10 below, the SSH access was copied from the Linode and used to login to the root using the Command Prompt using the same credentials.

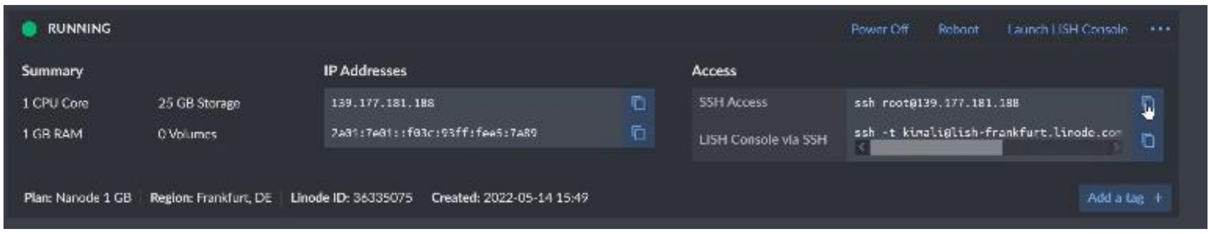

RUNNING
Power Off Reboot Launch LISH Console
Summary
1 CPU Core 25 GB Storage
1 GB RAM 0 Volumes
IP Addresses
139.177.181.188
2a01:7e01::f03c:93ff:fee5:7a89
Access
SSH Access ssh root@139.177.181.188
LISH Console via SSH ssh -t kimali@lish-frankfurt.linode.com
Plan: Nanode 1 GB | Region: Frankfurt, DE | Linode ID: 36335075 Created: 2022-05-14 15:49
Add a tag +

Figure 3.10: Linode status after provisioning the server

After logging into the root in the Command Prompt, LAMP Server was installed using the command shown in Figure 3.11 below.

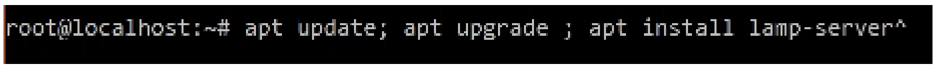

```
root@localhost:~# apt update; apt upgrade ; apt install lamp-server^
```

Figure 3.11: Command for installing LAMP Server

After installing the LAMP server, the next step was to log into the MySQL database system using the command shown in Figure 3.12 below.

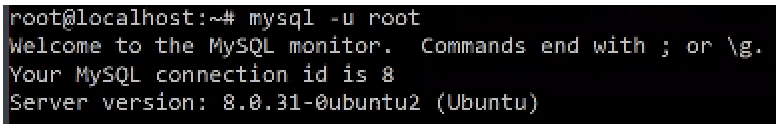
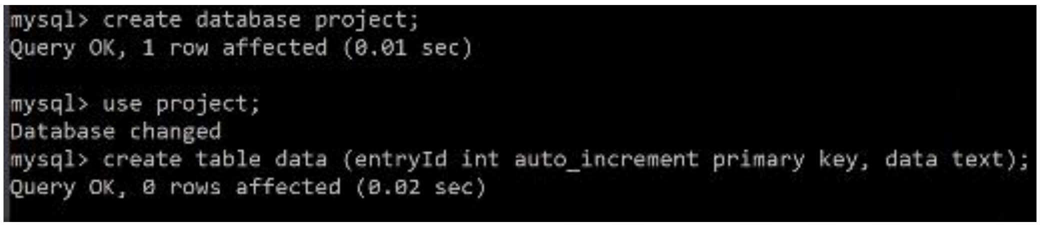

```
root@localhost:~# mysql -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.31-0ubuntu2 (Ubuntu)
```

Figure 3.12: Logging into MySQL system

Once in the MySQL database management system, the next step was to create a database named project and then a data table within the project database with the attributes shown in Figure 3.13 below.

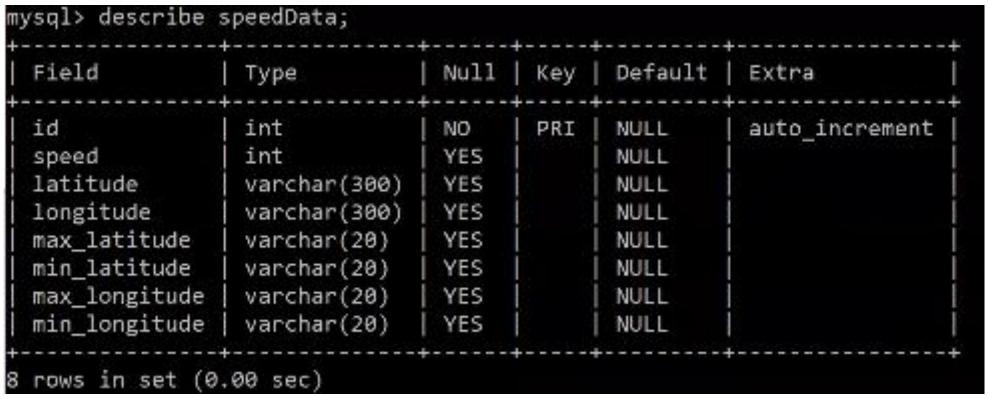

```
mysql> create database project;
Query OK, 1 row affected (0.01 sec)

mysql> use project;
Database changed
mysql> create table data (entryId int auto_increment primary key, data text);
Query OK, 0 rows affected (0.02 sec)
```

Figure 3.13: Creating database and data table

To show the data captured in the speedData table, the command shown in Figure 3.14 was used. The speed entry recorded in the table represents the speed limit specified for the selected region. The latitude and longitude entries respectively refer to the latest latitude and longitude values of the vertex of the polygon that has been drawn on the map using the user interface.

The max_latitude entry represents the top most latitude value of the drawn polygon whereas the min_latitude entry represents the bottom most latitude value. The min_longitude entry refers to the left-most longitude value of the polygon in question while the max_longitude entry refers to the right-most longitude value.



```
mysql> describe speedData;
+--------------+--------------+------+-----+---------+----------------+
| Field        | Type         | Null | Key | Default | Extra          |
+--------------+--------------+------+-----+---------+----------------+
| id           | int          | NO   | PRI | NULL    | auto_increment |
| speed        | int          | YES  |     | NULL    |                |
| latitude     | varchar(300) | YES  |     | NULL    |                |
| longitude    | varchar(300) | YES  |     | NULL    |                |
| max_latitude | varchar(20)  | YES  |     | NULL    |                |
| min_latitude | varchar(20)  | YES  |     | NULL    |                |
| max_longitude| varchar(20)  | YES  |     | NULL    |                |
| min_longitude| varchar(20)  | YES  |     | NULL    |                |
+--------------+--------------+------+-----+---------+----------------+
8 rows in set (0.00 sec)
```

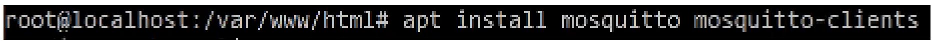
Figure 3.14: SpeedData table

The cloud platform was designed with the following modules:

1. Mosquitto MQTT broker
2. User Interface
3. Speed limiter interface
4. MySQL Database

3.3.1 Mosquitto MQTT broker setup

To install Mosquitto MQTT broker to the Ubuntu VPS, the command shown in Figure 3.15 below was used.



```
root@localhost:/var/www/html# apt install mosquitto mosquitto-clients
```

Figure 3.15: Command for installing Mosquitto MQTT broker

Linux screen command was used to create background monitored processes to be used in monitoring the mosquito topics. This was achieved by the following command:

```
screen -S mqtt
```

This command created a background process named mqtt to be used to monitor the MQTT topics. Within the screen, terminal multiplexer (tmux) was used to split the terminal to be able to monitor multiple events on the Linux terminal. To achieve this, the tmux command was used. Two MQTT topics were monitored in the screen process using the terminal multiplexer. The topics were:

1. GPS topic: - This topic was used by the device to send GPS coordinates to the server
2. Speed topic: - This is the topic that was subscribed to by the device where the speed limit is sent by the server

To monitor these two processes, the following commands were used:

```
mosquitto_sub -t test/gps
mosquitto_sub -t test/speed
```

3.3.2 User Interface

The frontend system consists of a Google Map in which the user draws polygons and then specifies the speed limit for each region. Though the use of Google Maps APIs, when a user draws a polygon, they are prompted to input the specific speed limit for the region covered by the polygon.

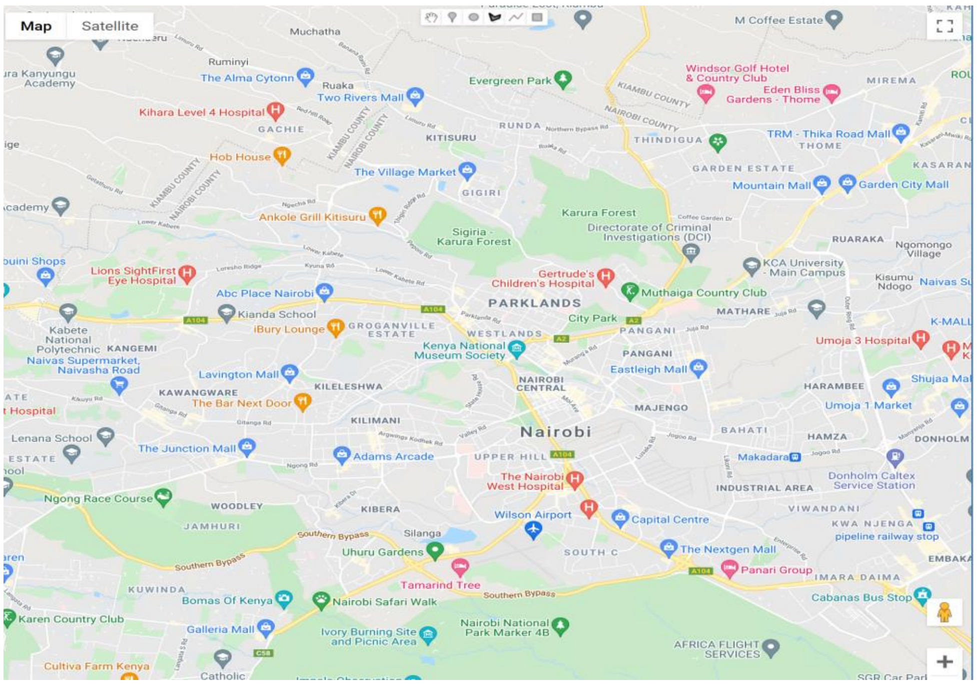


Figure 3.16: Frontend map for the user interface

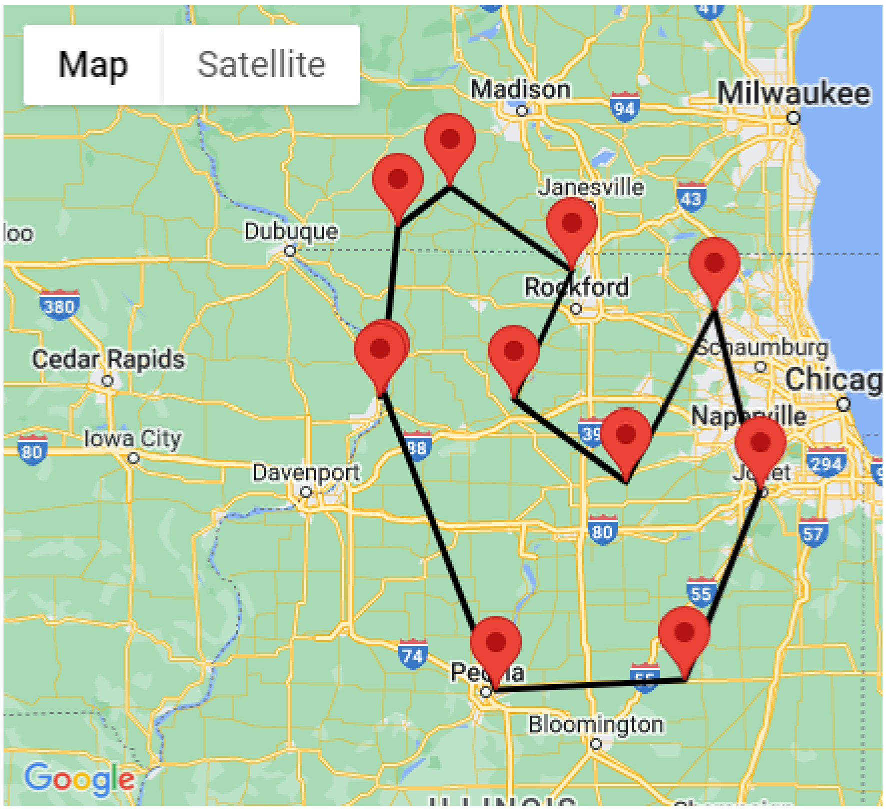
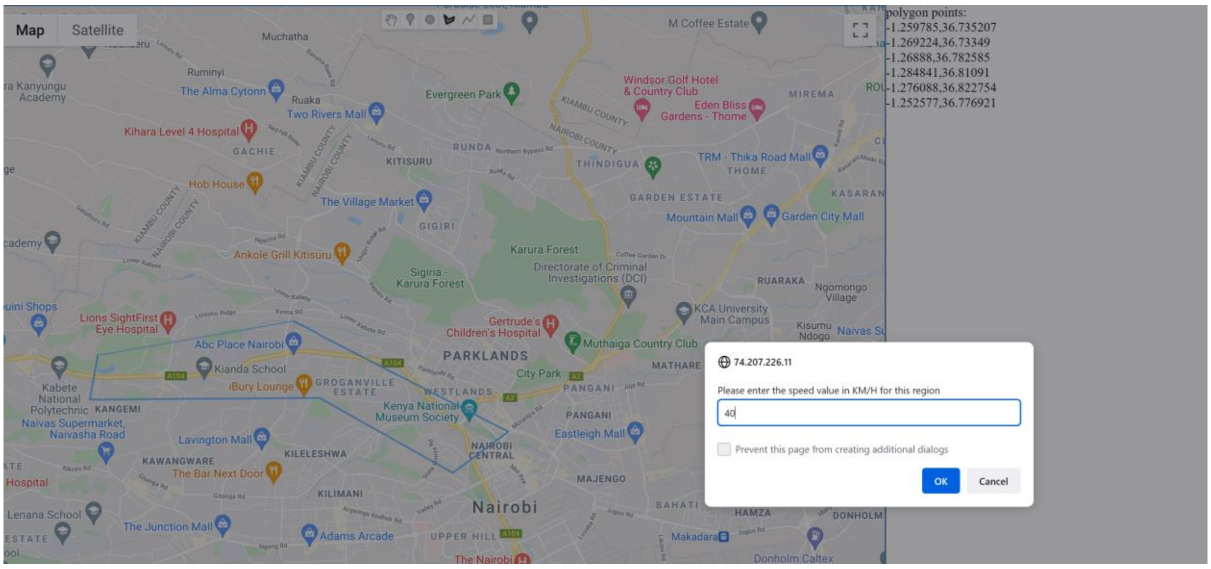


Figure 3.17: Google Map API for drawing complex polygons



```
polygon points:
-1.259785,36.735207
-1.269224,36.73349
-1.26888,36.782585
-1.284841,36.81091
-1.276088,36.822754
-1.252577,36.776921
```

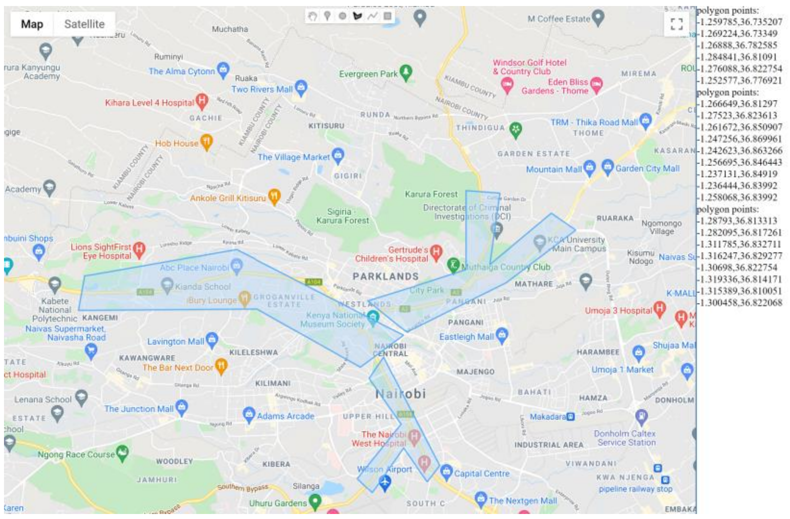
Please enter the speed value in KM/H for this region

40

Prevent this page from creating additional dialogs

Figure 3.18: Specifying the speed limit after drawing the polygon

The user can view the coordinates of the vertices of the drawn polygon on the right side of the map. The user will then enter the region speed limit and press OK. To draw additional polygons, the user should repeat the same procedure and specify unique speed limitations for each region.



```
polygon points:
-1.259785,36.735207
-1.269224,36.73349
-1.26888,36.782585
-1.284841,36.81091
-1.276088,36.822754
-1.252577,36.776921
polygon points:
-1.266649,36.81297
-1.27523,36.823613
-1.261672,36.850907
-1.247256,36.869961
-1.242623,36.863266
-1.256695,36.846443
-1.237131,36.84919
-1.236444,36.83992
-1.258068,36.83992
polygon points:
-1.28793,36.813313
-1.282095,36.817261
-1.311785,36.832711
-1.316247,36.829277
-1.30698,36.822754
-1.319336,36.814171
-1.315389,36.810051
-1.300458,36.822068
```

Figure 3.19: Multiple polygons for different speed limits

When the user sets the speed limits, the MySQL database is updated with the vertices of the polygons that correspond to the speed limit values. These values are recorded in the speedData table shown in Figure 3.14 above. The max_latitude, min_latitude, max_longitude, and min_longitude entries can be illustrated in Figure 3.20 below.

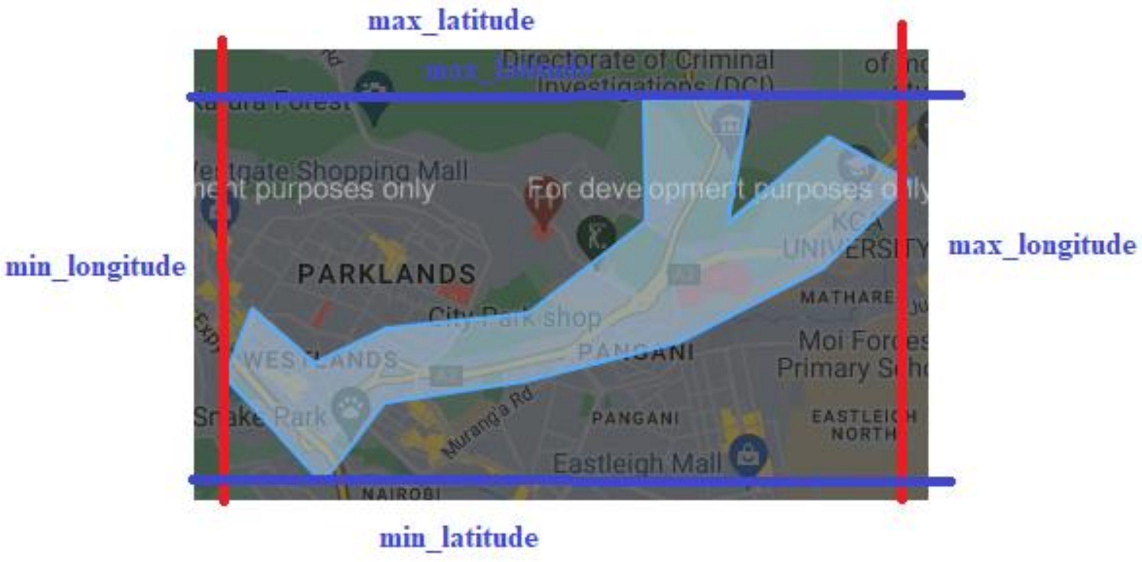
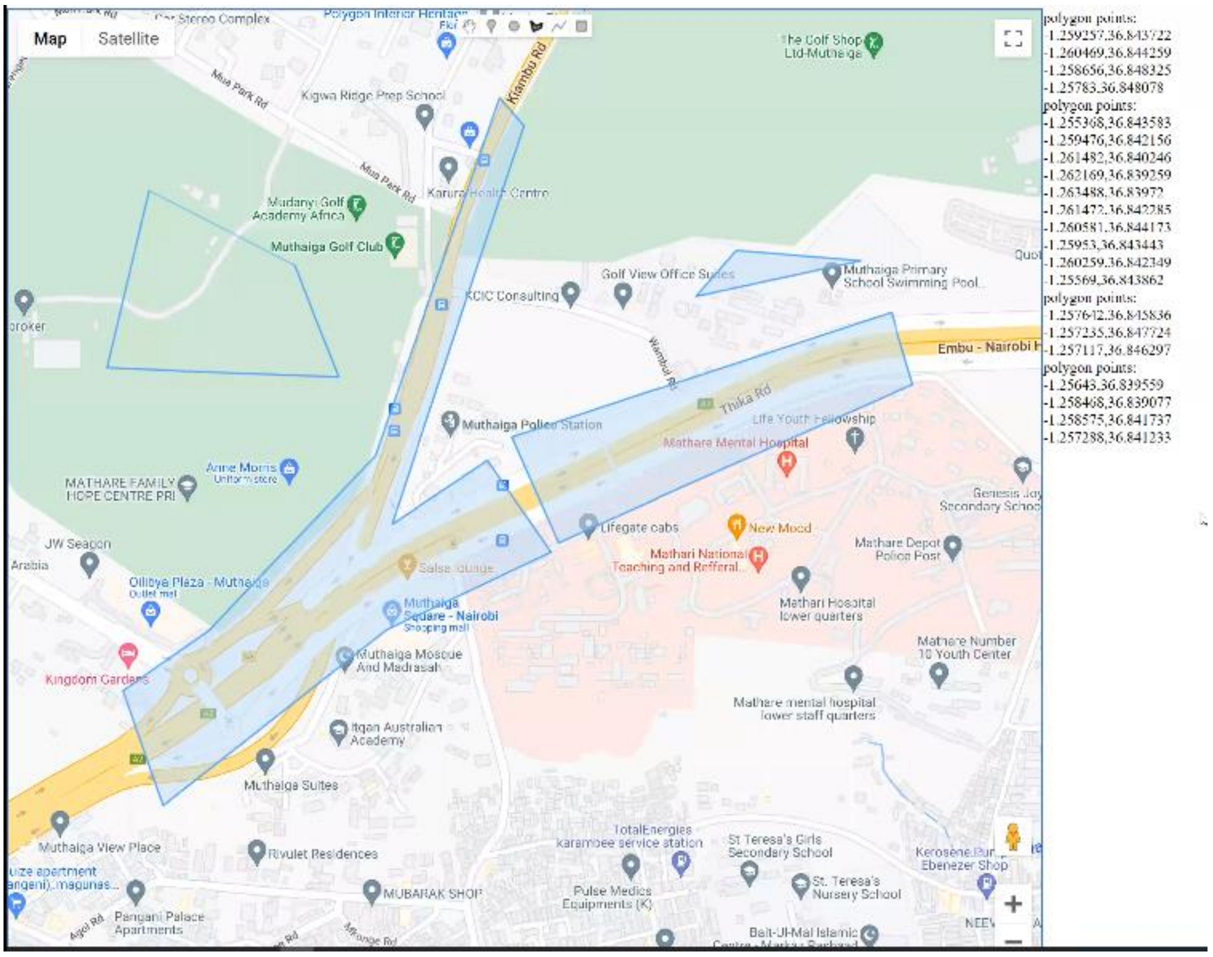
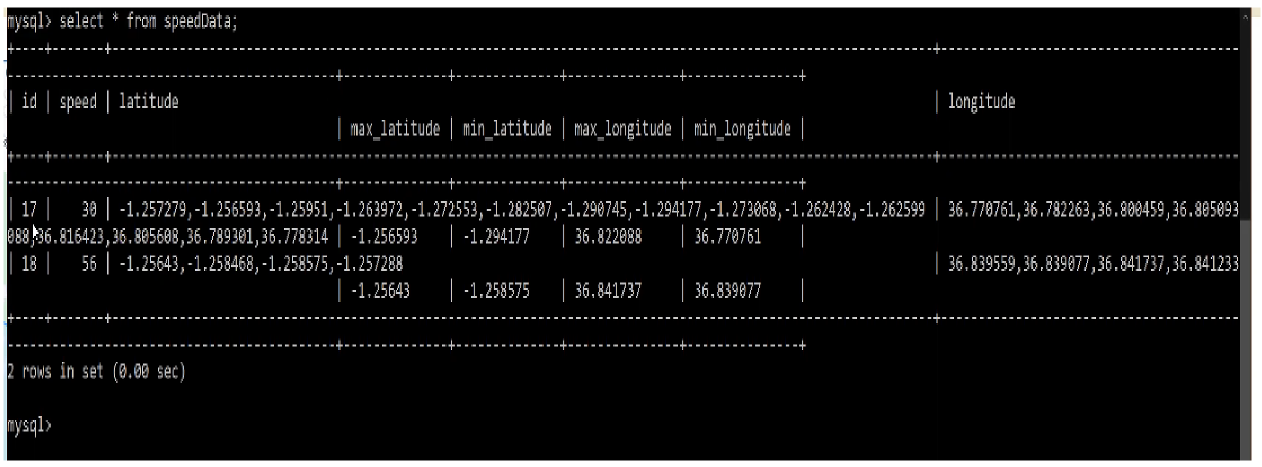


Figure 3.20: Illustration of the various entries in the data table

To illustrate this, the polygons in Figure 3.21 below show the entries recorded in the speedData table.



```
polygon points:
-1.259257,36.843722
-1.260469,36.844259
-1.258656,36.848325
-1.25783,36.848078
polygon points:
-1.255368,36.843583
-1.259476,36.842156
-1.261482,36.840246
-1.262169,36.839259
-1.263488,36.83972
-1.261472,36.842785
-1.260581,36.844173
-1.25953,36.843443
-1.260259,36.842349
-1.25569,36.843862
polygon points:
-1.257642,36.845836
-1.257235,36.847724
-1.257117,36.846297
polygon points:
-1.25643,36.839559
-1.258468,36.839077
-1.258575,36.841737
-1.257288,36.841233
```



```
mysql> select * from speedData;
+----+-------+--------------------------------------------------------------------------------------------------+-----------------------------------------
----------------------------------------+-------------+-------------+--------------+--------------+
| id | speed | latitude                                                                                         | longitude
                                        | max_latitude | min_latitude | max_longitude | min_longitude |
+----+-------+--------------------------------------------------------------------------------------------------+-----------------------------------------
----------------------------------------+-------------+-------------+--------------+--------------+
| 17 |    30 | -1.257279,-1.256593,-1.25951,-1.263972,-1.272553,-1.282507,-1.290745,-1.294177,-1.273068,-1.262428,-1.262599 | 36.770761,36.782263,36.800459,36.805093
088,36.816423,36.805608,36.789301,36.778314 | -1.256593    | -1.294177    | 36.822088     | 36.770761     |
| 18 |    56 | -1.25643,-1.258468,-1.258575,-1.257288                                                           | 36.839559,36.839077,36.841737,36.841233
                                        | -1.25643     | -1.258575    | 36.841737     | 36.839077     |
+----+-------+--------------------------------------------------------------------------------------------------+-----------------------------------------
----------------------------------------+-------------+-------------+--------------+--------------+

2 rows in set (0.00 sec)

mysql>
```

Figure 3.21: Illustration of the data recorded in the data table

3.3.3 MySQL Database

In this research study, mapped speed data were stored on the server using MySQL database. The mapped speed data were stored in a table named speedData.

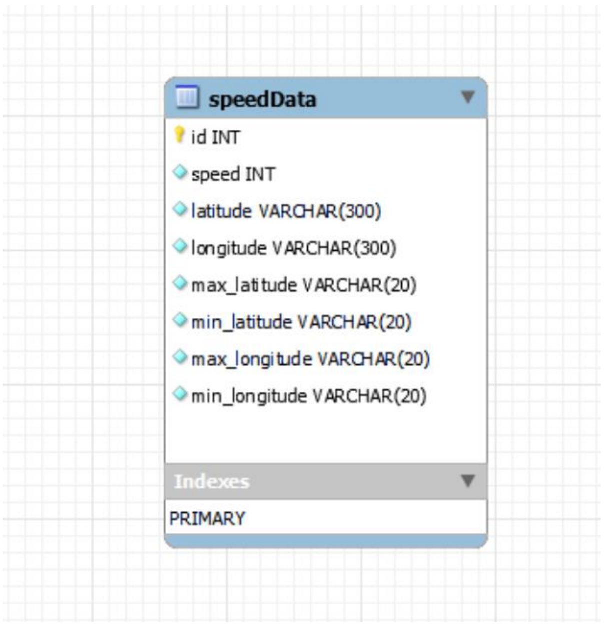

speedData
id INT
speed INT
latitude VARCHAR(300)
longitude VARCHAR(300)
max_latitude VARCHAR(20)
min_latitude VARCHAR(20)
max_longitude VARCHAR(20)
min_longitude VARCHAR(20)
Indexes
PRIMARY

Figure 3.22: Database structure

After the user draws a polygon and clicks OK, the following data format is transmitted to the backend of the server:

Latitude(0),longitude(0):latitude(1),longitude(1) : …:latitude(n),longitude(n)

Map data example: -1.259785,36.735207:-1.269224,36.73349:-1.26888,36.782585:-1.284841 ,36.81091:-1.276088,36.822754:-1.252577,36.776921.

Once this information reaches the backend, the latitude and longitude values are stored in the respective comma-delimited strings in the database. Through iteration, the minimum and maximum latitude and longitude values of the polygon are extracted from the strings. The database stores these values as min_latitude and min_longitude for the minimum values and max_latitude and max_longitude for the maximum values. The speed value is transmitted separately from the speed index and is recorded in the database as speed.

3.3.4 Speed limiter interface

The speed limiter published the coordinates to the MQTT GPS topic, which was then compared against the information stored in the database. The database access script was developed using the PHP programming language. The script obtained the GPS coordinates published by the device, accessed the database, determined in which section of the database the location of the device resides, obtained the speed limit for that place, and published it to the test/speed topic to which the device was subscribed. In this study, only one gadget was employed. In the event that multiple devices were utilized, each device would subscribe to a distinct topic, and the PHP script would therefore publish to a topic subscribed to by the device that published the GPS data.

In the speed-computing algorithm, the device published comma-separated location data in the format given in the GNSS data format table described in the SIM808 GNSS application note datasheet. The comma-separated string is then placed in an array that holds each element of the string separately. The latitude and longitude are then extracted from positions 4 and 5 as shown in the SIM808 GNSS application note datasheet in Appendix 1. In the firmware, this corresponds to indexes 3 and 4 in the array as it begins at position 0.

The system then queries the database for locations where the device's latitude falls between the maximum and minimum latitude and longitude falls between the minimum and maximum longitude. This results in the selection of one or more polygons from the database.

The system then verifies whether the device's coordinates fall within a polygon by counting the number of times the device's latitude intersects the polygon's vertices if the latitude is extended laterally. As shown in Figure 3.24, if the latitude crosses an odd number of vertices, the point is inside the polygon, whereas if it crosses an even number of vertices, the point is outside the polygon. Once the exact polygon in which the point resides has been discovered, the associated speed limit value is obtained and published to the MQTT speed topic. The device then receives the speed limit for that region. The flowchart describing the development of the speed-computing algorithm is shown in Figure 3.23.

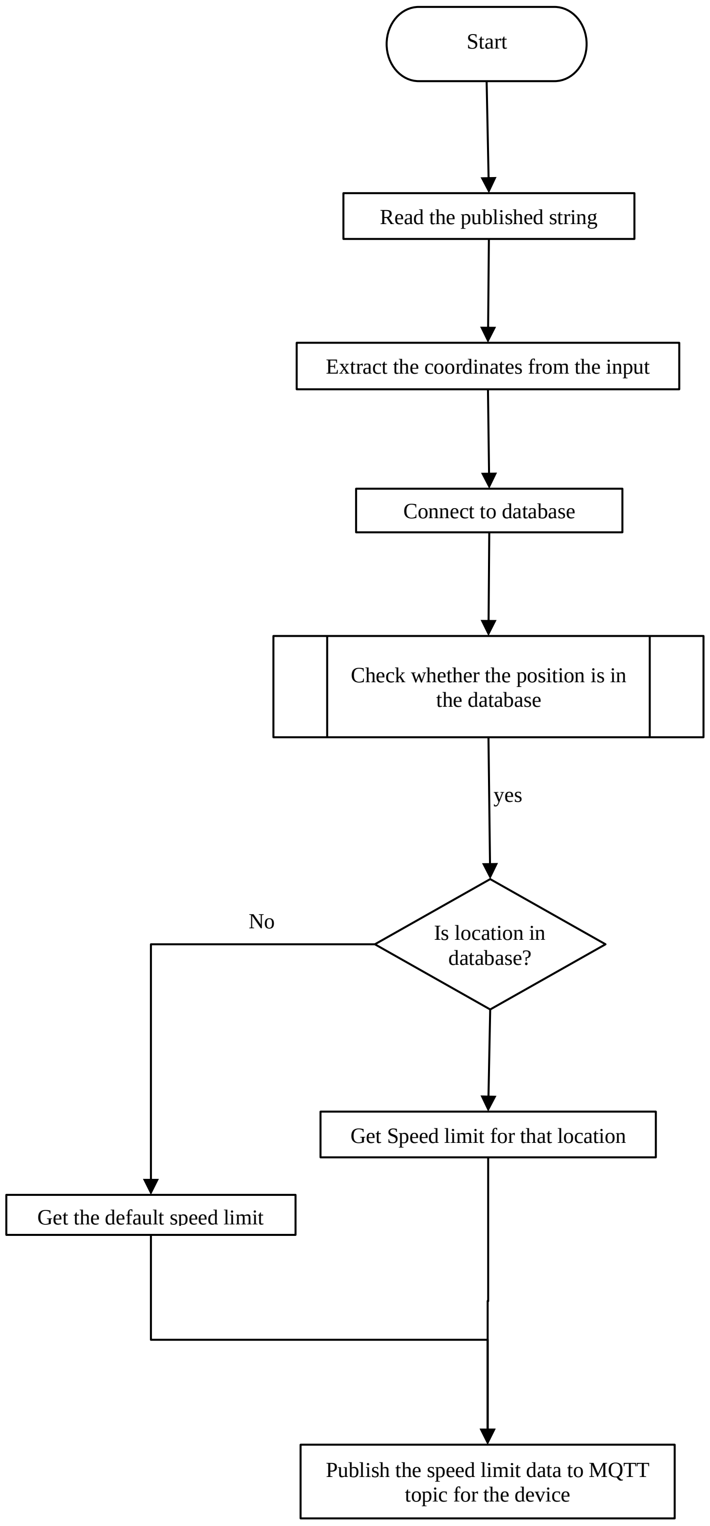

Start
Read the published string
Extract the coordinates from the input
Connect to database
Check whether the position is in the database
yes
Is location in database?
No
Get Speed limit for that location
Get the default speed limit
Publish the speed limit data to MQTT topic for the device

Figure 3.23: Speed computing algorithm

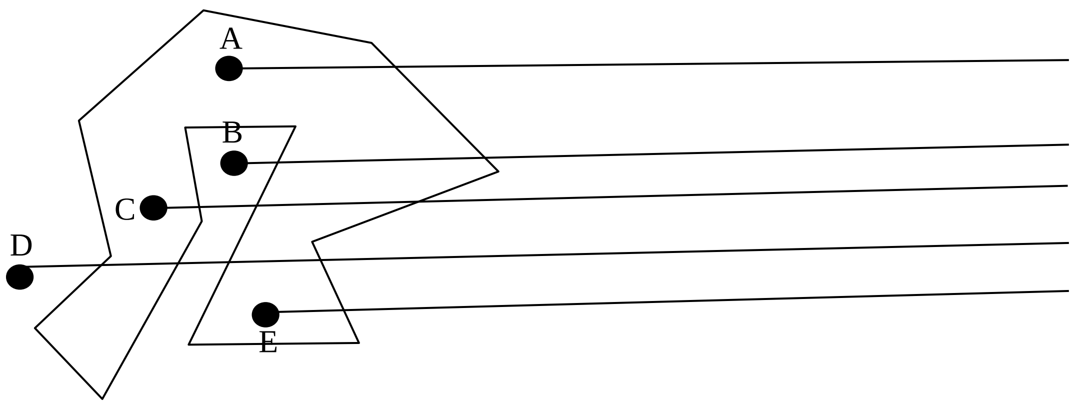


Figure 3.24: Determining the position of points in the polygon

From the Figure 3.24 above, the data can be summarized as shown in Table 3.2 below.

Table 3.2: Determining the position of points in the polygon

POINT	VERTICES CROSSED	POSITION
A	1	Inside
B	2	Outside
C	3	Inside
D	4	Outside
E	1	Inside

The algorithm was implemented using PHP to determine if the device's position was within the polygon and return the corresponding speed value for that polygon. The device would then compare the current speed to the specified speed limit for the region. If the speed was above the speed limit, the device would limit the speed to fall within the limit; if the speed was within the limit, no action was taken.

CHAPTER 4: RESULTS AND DISCUSSION

A speed-limiter user interface was developed that allows users to draw polygons over areas of interest and specify speed limits. When two or more polygons overlap in this research study, the speed limit of the most recently defined polygon is used as the default speed limit for the region of overlap. The picture below depicts two polygons, the first with a speed limit of 80KM/H and the second with a speed limit of 50KM/H. The default speed limit of the circled overlap zone is 50KM/H, which is obtained from the second region. In addition, when the test vehicle travels through an undefined zone without a polygon, the default speed limit of 80KM/H is imposed. The user interface allows a user to define as many regions as possible with distinct speed limit values. When network connectivity is poor, for example, due to blockages in certain regions, the speed-limiter system defaults to the 80KM/H speed limit. After reconnecting to the network, the smart speed limiter system operates in accordance with the current zone's speed limit. If the region is undefined, the system reverts to the default speed limit.

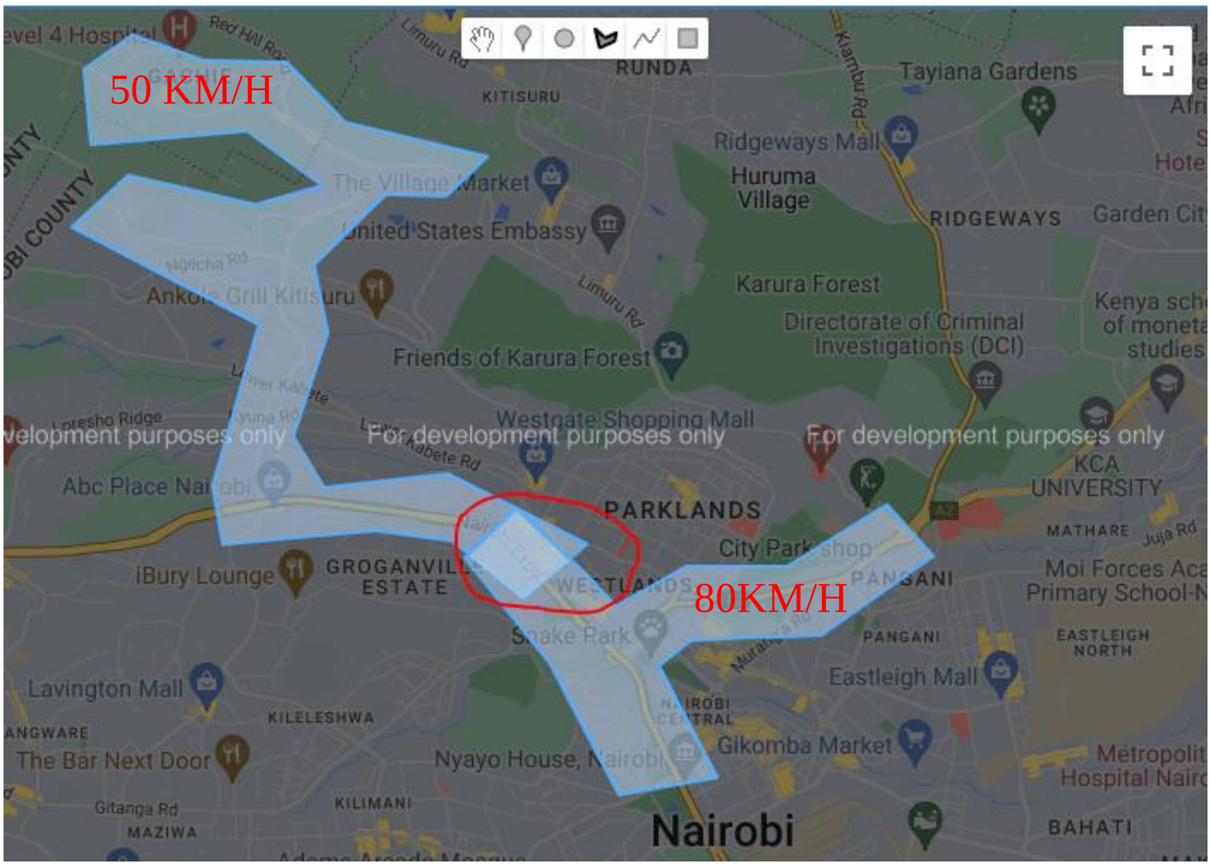


Figure 4.1: Google Map user interface illustrating overlapping regions

A smart speed-limiter prototype was developed and integrated with the signal cables from the gas pedal in the test car. These signal cables were drawn from the car's dashboard. As illustrated in Figure 4.2, the test car model has two signal wires that were cut and extended using two yellow and two blue cables. To enable the car to run normally, the ends of these cable extensions were connected through a relay in a normally closed configuration. The 12 volt and ground cables were tapped from the fuse box during the connection of the developed prototype with the car.

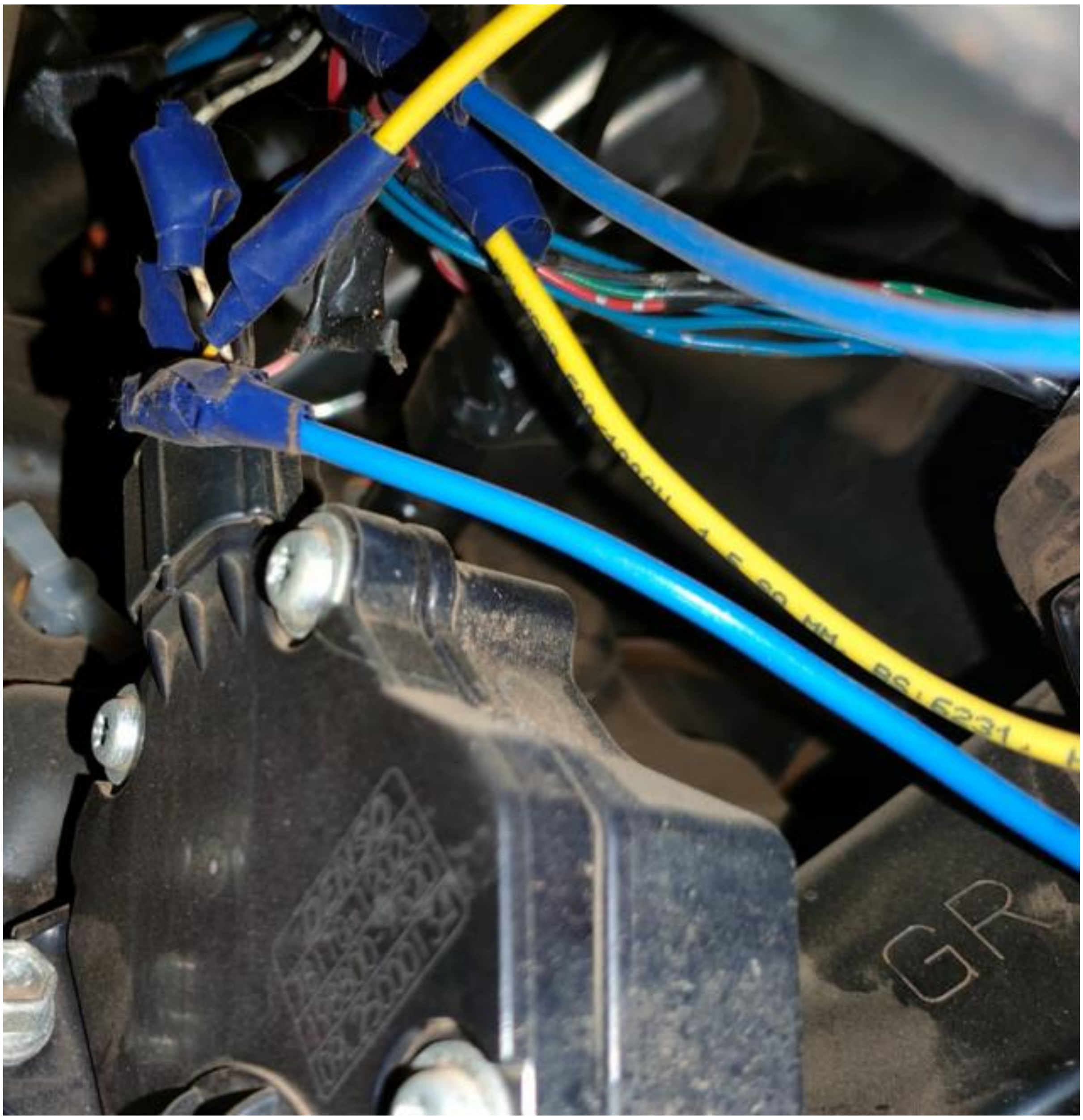


Figure 4.2: Signal cables extension

The various modules that comprised the smart speed limiter hardware prototype were coupled as depicted in Figure 4.3 below. The smart speed limiter was covered on top, and GPS and GSM antennas were attached. The switch on top of the smart speed limiter allows the user to activate or deactivate the limiter system by turning it on or off respectively.

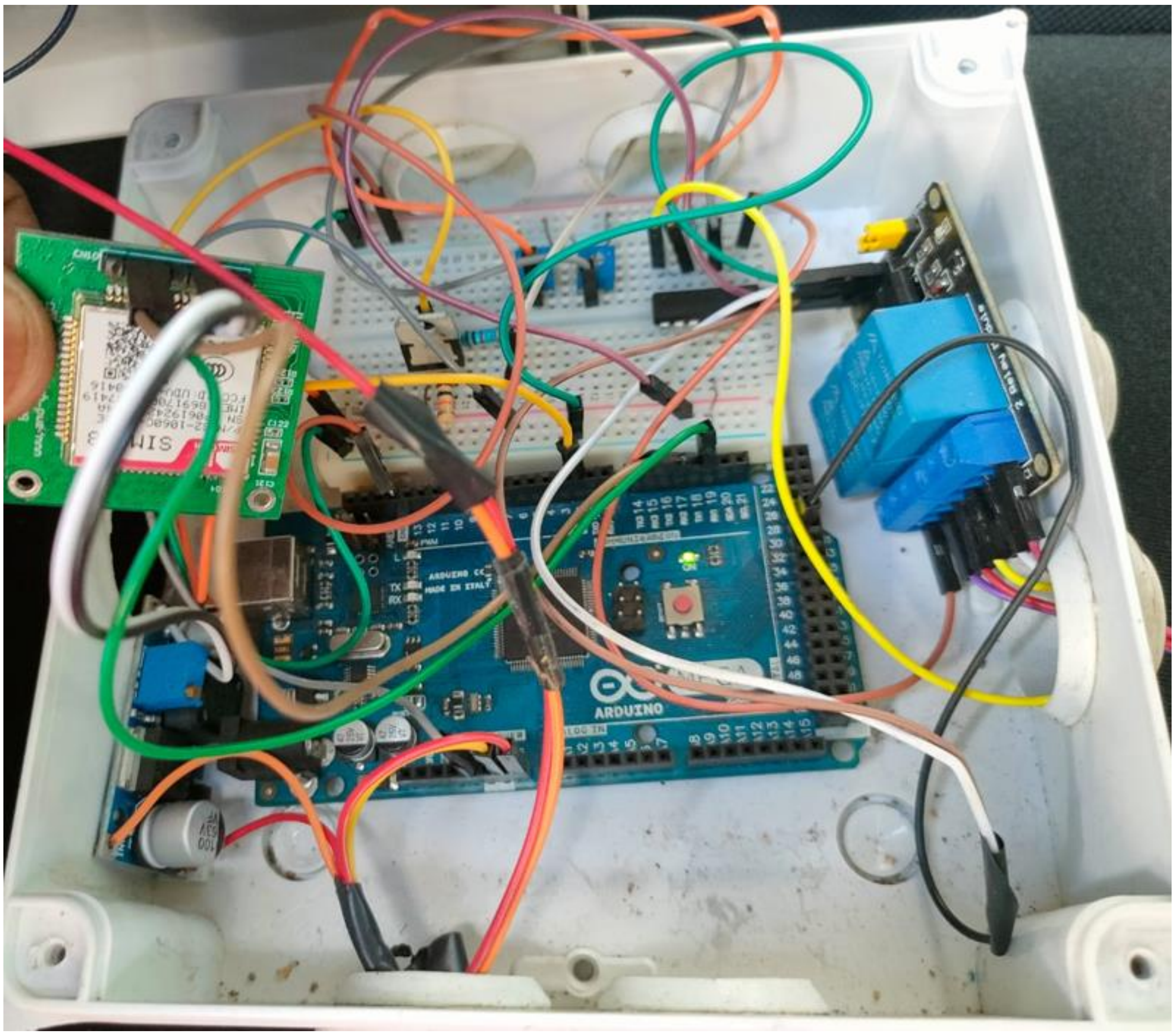


Figure 4.3: Interconnection of hardware modules in the speed limiter system

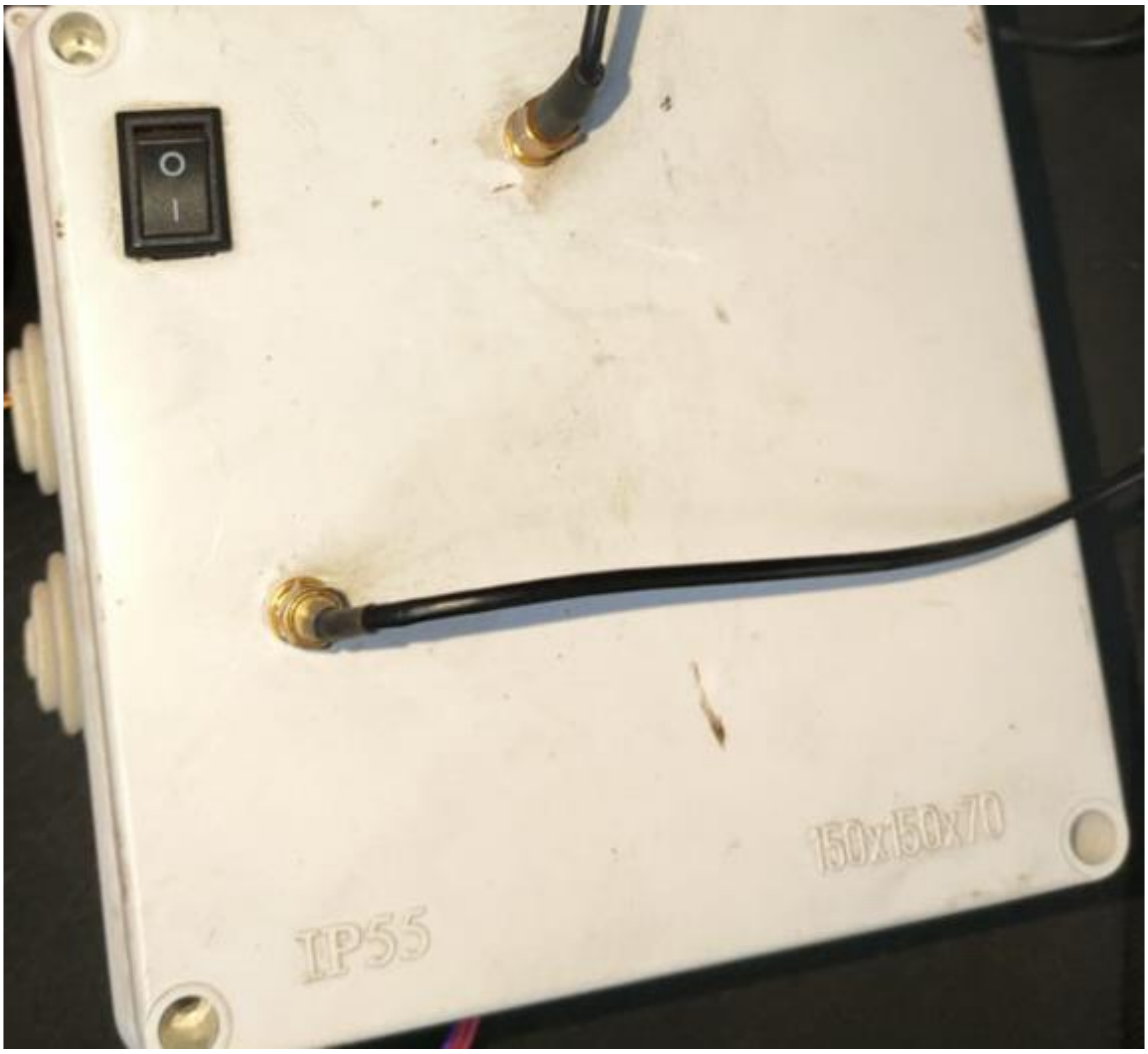


Figure 4.4: The speed limiter switch on the cover

The green wire represents the speed signal cable from the speedometer dashboard, the blue cables represent the signals from the gas pedal, and the yellow cables represent the signals to the ECU in the limiter integration with the test car. These yellow cables to the ECU are connected to the blue cables through a relay in the normally closed configuration. The red cable serves as the 12v signal cable, while the black cable operates as the ground connection.

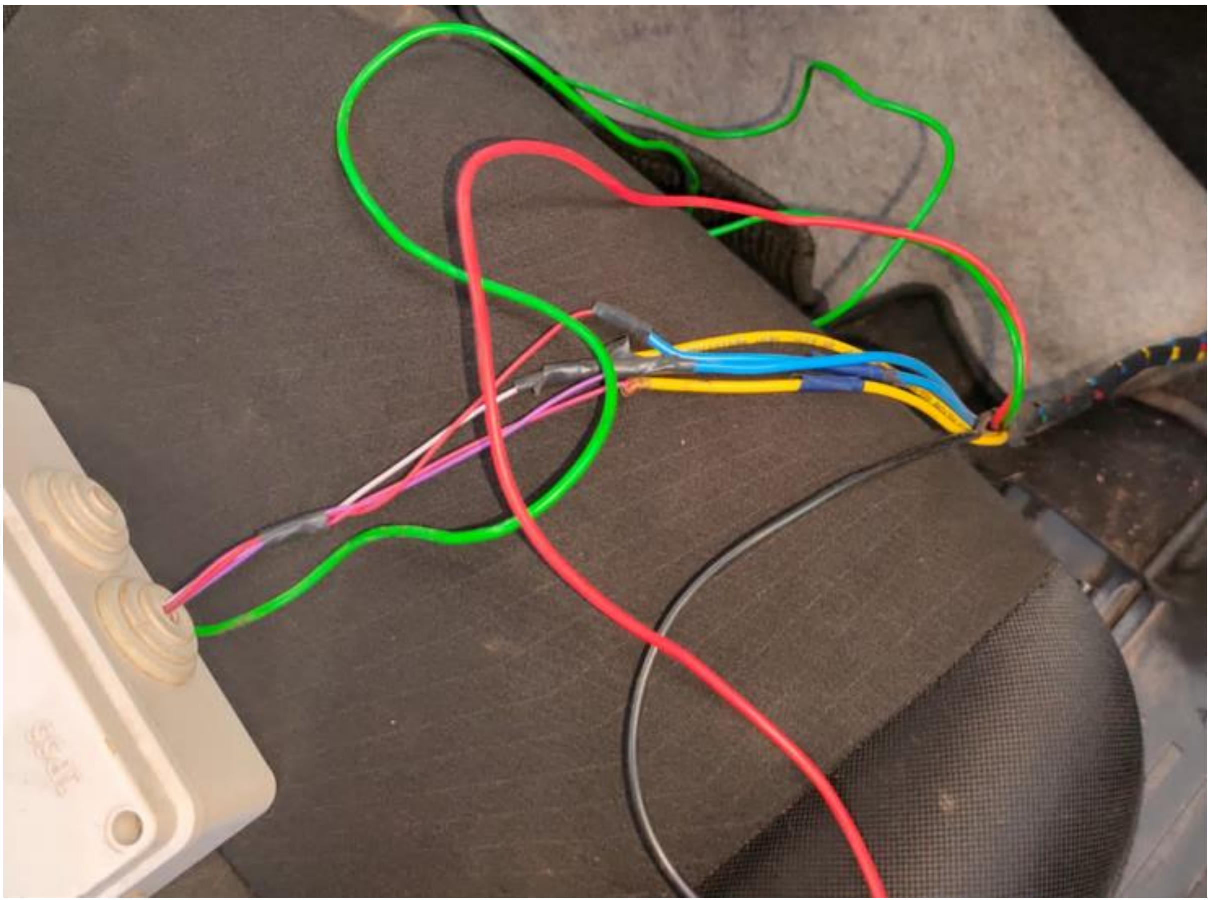


Figure 4.5: Cable connection in the speed limiter integration

As indicated in the appendices section, the firmware utilized to manage the built prototype as well as the interface to the server was developed. The prototype was able to communicate with the server through the GSM and GPS modules and record the various values in tabular format, as shown in Figure 3.21 in the methodology section above.

CHAPTER 5: CONCLUSION

Dynamic speed limiters are essential for guaranteeing safety in busy areas and smooth traffic flow. This research study's smart speed limiter allows a user to select multiple speed limits for different locations in a particular area. This limiter addresses the issues raised by existing speed governors, which are easily circumvented by drivers. In addition, the system delivers real-time data that is stored on the server. This research study demonstrates how to use GPS, GPRS, and MQTT to construct a reliable speed limiter.

REFERENCES

Bai, Y., Sun, Q., Du, L., Yu, M., & Bai, J. (2014). Two laboratory methods for the calibration of GPS speed meters. *Measurement Science and Technology*, *26*(1), 015005.

Luzuriaga, J. E., Perez, M., Boronat, P., Cano, J. C., Calafate, C., & Manzoni, P. (2015, January). A comparative evaluation of AMQP and MQTT protocols over unstable and mobile networks. In *2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC)* (pp. 931-936). IEEE.

Grajkowski, K.J., Erickson, S. C., Koenig, D. J. (2015). Electronic throttle control. *EURASIP Journal on Embedded Systems*. **https://jes-eurasipjournals.springeropen.com/articles/10.1186/1687-3963-2013-6**

Tamazin, M., Noureldin, A., Korenberg, M. J., & Massoud, A. (2016). Robust fine acquisition algorithm for GPS receiver with limited resources. *GPS solutions, 20*(1), 77-88.

Priyadarshni, V., Gopi Krishna, P., & Sreenivasa Ravi, K. (2016). GPS and GSM enabled embedded vehicle speed limiting device. *Indian Journal of Science and Technology*, *9*(17), 1-6.

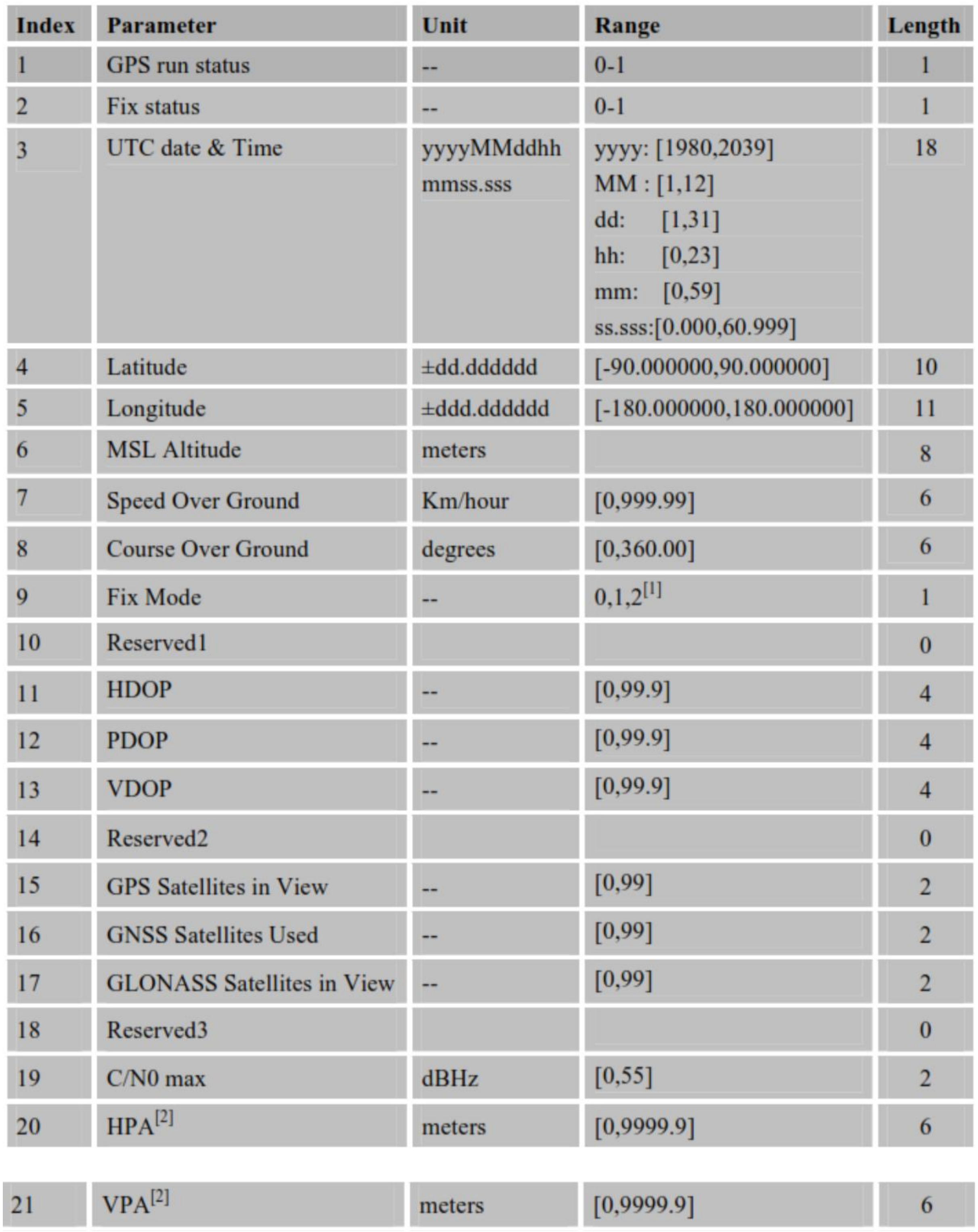
To, V. Q., Naghshtabrizi, P., Hashemi, S., Yan, Z., & Blankenship, J. R. (2015). *U.S. Patent No. 9,056,550*. Washington, DC: U.S. Patent and Trademark Office.

KII, Y., Maruyama, T., Matsuura, M. (2015). Speed limiter. *Google patents*. **https://patents.google.com/patent/US9085237**

Xiong, Z., Song, Z., Scalera, A., Ferrera, E., Sottile, F., Brizzi, P., Tomasi, R., & Spirito, M. A. (2013). Hybrid WSN and RFID indoor positioning and tracking system. *J Embedded Systems* **2013**, 6. **https://doi.org/10.1186/1687-3963-2013-6**

APPENDICES

Appendix 1: GNSS Application Note Datasheet



Index	Parameter	Unit	Range	Length
1	GPS run status	--	0-1	1
2	Fix status	--	0-1	1
3	UTC date & Time	yyyyMMddhh mmss.sss	yyyy: [1980,2039] MM : [1,12] dd: [1,31] hh: [0,23] mm: [0,59] ss.sss:[0.000,60.999]	18
4	Latitude	±dd.dddddd	[-90.000000,90.000000]	10
5	Longitude	±ddd.dddddd	[-180.000000,180.000000]	11
6	MSL Altitude	meters		8
7	Speed Over Ground	Km/hour	[0,999.99]	6
8	Course Over Ground	degrees	[0,360.00]	6
9	Fix Mode	--	0,1,2[1]	1
10	Reserved1			0
11	HDOP	--	[0,99.9]	4
12	PDOP	--	[0,99.9]	4
13	VDOP	--	[0,99.9]	4
14	Reserved2			0
15	GPS Satellites in View	--	[0,99]	2
16	GNSS Satellites Used	--	[0,99]	2
17	GLONASS Satellites in View	--	[0,99]	2
18	Reserved3			0
19	C/N0 max	dBHz	[0,55]	2
20	HPA[2]	meters	[0,9999.9]	6
21	VPA[2]	meters	[0,9999.9]	6

Appendix 2: MQTT Code

```
<?php
while(1){
$handle = fopen ("php://stdin","r");
$line = fgets($handle);
//echo "\n php-->".htmlentities($line)." yes\n";

$db_name = "root";
$db_pswd="Fedha@2015";
$db_database = "project";
$db_host = "localhost";
$con = mysqli_connect("$db_host", "$db_name", "$db_pswd","$db_database") or die("database error");

$data=explode(",",$line);

$latitude_x=$data[3];
$longitude_y=$data[4];

$sql="select * from speedData where max_latitude>=$latitude_x and min_latitude<=$latitude_x and max_longitude>=$longitude_y and min_longitude<=$longitude_y";
echo "\r\n".$sql."\r\n";
$query=mysqli_query($con,$sql);
if(!@$query) echo(mysqli_error($con).":".$sql);
```

```
$count=mysqli_num_rows($query);
if($count<1){
        $speed=80;
}
else{
        $speed=80;
        while($row=mysqli_fetch_array($query)){
                $vertices_x = explode(",",$row['latitude']);    // x-coordinates of the vertices of
the polygon
                $vertices_y = explode(",",$row['longitude']); // y-coordinates of the vertices of
the polygon

                $points_polygon = count($vertices_x) - 1;  // number vertices - zero-based array
                if              (is_in_polygon($points_polygon,              $vertices_x,
$vertices_y,$latitude_x,$longitude_y)){
                        echo "Is in polygon!";
                        $speed=$row['speed'];
                        break;

                }
                else echo "Is not in polygon";
        }


}
```

```
echo "\n"."Speed is: ".$speed."KM/H\n";
$command='mosquitto_pub -t test/speed -m '.$speed;
shell_exec($command);


}


function is_in_polygon($points_polygon, $vertices_x, $vertices_y, $longitude_x, $latitude_y)
{
  $i = $j = $c = 0;
  for ($i = 0, $j = $points_polygon ; $i < $points_polygon; $j = $i++) {


              $vertices_y[$i]=(double)$vertices_y[$i];
              $vertices_y[$j]=(double)$vertices_y[$j];
              $latitude_y=(double)$latitude_y;
              $longitude_x=(double)$longitude_x;
    if ( (($vertices_y[$i]  >  $latitude_y != ($vertices_y[$j] > $latitude_y)) && ($longitude_x < ($vertices_x[$j] - $vertices_x[$i]) * ($latitude_y - $vertices_y[$i]) / ($vertices_y[$j] - $vertices_y[$i]) + $vertices_x[$i]) ) )
       $c = !$c;
  }
  return $c;
}



?>
```

Appendix 3: Speed PHP Code

```
<?php
	$db_host="localhost";
	$db_name = "kimali";
	$db_pswd="mscDct";
	$db_database = "project";

	$db_connection = mysqli_connect("$db_host", "$db_name",
"$db_pswd","$db_database") or die("database error");


	if(isset($_GET['speed'])){
		$speed = $_GET['speed'];
		$coords = $_GET['coords'];

		$data=explode(":",$coords);
		$i=0;
		$lat_max=-360;
		$long_max=-360;
		$lat_min=360;
		$long_min=360;
		$latitude="";
		$longitude="";

		for($i=0;$i<sizeof($data);$i++){
```

```
                $lat_long=explode(",",$data[$i]);
                $lat=$lat_long[0];
                $long=$lat_long[1];
                if($lat>$lat_max) $lat_max=$lat;
                if($lat<$lat_min) $lat_min=$lat;
                if($long>$long_max) $long_max=$long;
                if($long<$long_min) $long_min=$long;
                if($i>0) {$latitude .= ","; $longitude .= ",";}
                $latitude.=$lat;
                $longitude.=$long;
            }

            $sql="insert into speedData values(NULL,'$speed','$latitude','$longitude','$lat_max','$lat_min','$long_max','$long_min')";
            echo $sql;
            $query=@mysqli_query($db_connection,$sql);
            if(!$query) echo "speed setting error";
            else echo "successful ";
            exit();
        }

        if(isset($_POST['clear_data'])){
            $sql="delete from speedData where 1";
            mysqli_query($db_connection,$sql) or die(mysqli_error($db_connection));
            echo('<script> alert("Data cleared"); </script>');
        }
?>
<html>
        <head>
                <title>Set the speed limits</title>
                <style type="text/css">
                        html, body, #map_canvas {
                           height: 100%;
                           width: 80%;
                           margin: 0px;
                           padding: 0px
                        }
                </style>
                <script
src="http://maps.googleapis.com/maps/api/js?key=AIzaSyCOU_BozG6JWBgmVm6y3890-
WP-TFkjGYY&libraries=places,drawing"></script>

                <script type="text/javascript">
                        var geocoder;
                        var map;
                        var polygonArray = [];

                        function initialize() {
                           map = new google.maps.Map(
                           document.getElementById("map_canvas"), {
```

```
                    center: new google.maps.LatLng(-1.274870, 36.808060),
                    zoom: 13,
                    mapTypeId: google.maps.MapTypeId.ROADMAP
                });
                //alert("1");
                var drawingManager = new
google.maps.drawing.DrawingManager({
                    drawingMode: google.maps.drawing.OverlayType.POLYGON,
                    drawingControl: true,
                    drawingControlOptions: {
                        position: google.maps.ControlPosition.TOP_CENTER,
                        drawingModes: [
                        google.maps.drawing.OverlayType.MARKER,
                        google.maps.drawing.OverlayType.CIRCLE,
                        google.maps.drawing.OverlayType.POLYGON,
                        google.maps.drawing.OverlayType.POLYLINE,
                        google.maps.drawing.OverlayType.RECTANGLE]
                    },
                    markerOptions: {
                        icon: 'images/arrow.png'
                    },
                    circleOptions: {
                        fillColor: '#ffff00',
                        fillOpacity: 1,
                        strokeWeight: 5,
                        clickable: false,

                        editable: true,

                        zIndex: 1

                    },

                    polygonOptions: {

                        fillColor: '#BCDCF9',

                        fillOpacity: 0.5,

                        strokeWeight: 2,

                        strokeColor: '#57ACF9',

                        clickable: false,

                        editable: false,

                        zIndex: 1

                    }

                });


                console.log(drawingManager)

                drawingManager.setMap(map)


                google.maps.event.addListener(drawingManager,
'polygoncomplete', function (polygon) {

                        var coords="";

                    document.getElementById('info').innerHTML    +=    "polygon
points:" + "<br>";

                    for (var i = 0; i < polygon.getPath().getLength(); i++) {
```

```
                                        document.getElementById('info').innerHTML          +=
polygon.getPath().getAt(i).toUrlValue(6) + "<br>";
                                        if(i>0) coords+=":";
                                  coords+=polygon.getPath().getAt(i).toUrlValue(6);


                                }
                                alert(coords);
                                var speed=prompt("Please enter the speed value in KM/H for this
region");
                            if(speed==null){
                                  alert("no speed value entered");
                            }
                            else {
                                  alert("the speed is " + speed);
                                  if (window.XMLHttpRequest)
                                              {// code for IE7+, Firefox, Chrome, Opera, Safari
                                                    xmlhttp=new XMLHttpRequest();
                                              }
                                              else
                                              {// code for IE6, IE5
                                                    xmlhttp=new
ActiveXObject("Microsoft.XMLHTTP");
                                              }


                                              xmlhttp.onreadystatechange=function()
                                        {
                                             if         (xmlhttp.readyState==4          &&
xmlhttp.status==200)
                                             {

                                                  alert(xmlhttp.responseText);

                                             }
                                        }

                                        xmlhttp.open("GET","speed.php?speed="         +
speed + "&coords=" + coords,true);
                                        xmlhttp.send();
                         }
                            polygonArray.push(polygon);
                          });

                         }
                         google.maps.event.addDomListener(window, "load", initialize);
                    </script>

          </head>

          <body>
                    <form action="" method="POST">
```

```
                              <button                    name="clear_data"                    type="submit"
value="clear_table">CLEAR DATA</button>
                    </form>


                    <div id="map_canvas" style=" border: 2px solid #3872ac; float: left;"></div>
                    <div id="info" style="float:left; width:15%;"></div>
          </body>


</html>
```

Appendix 4: Smart Limiter Arduino Code

```
#include "mqtt.h"
#include "speed_signal.h"
#include <string.h>


char gps_buffer[256];
int gps_buffer_pos;
char mqtt_buffer[256];
int mqtt_buffer_pos;
char imei[20];
char mqttString[256];


void clear_gps_buffer(){
  int i;
  for(i=0;i<256;i++){
    gps_buffer[i]=0x00;
  }
  gps_buffer_pos=0;
}

void clear_mqtt_buffer(){
  for(mqtt_buffer_pos=0;mqtt_buffer_pos<256;mqtt_buffer_pos++){
    mqtt_buffer[mqtt_buffer_pos]=0x00;
  }
  mqtt_buffer_pos=0;
}

void mqttCallback(char* topic, byte* payload, unsigned int len) {
  mqtt_buffer_pos=0;
  while(len){
    len--;
    mqtt_buffer[mqtt_buffer_pos]=(char) *payload;
    payload++;
    mqtt_buffer_pos++;
  }

  SerialMon.print("Message arrived [");
  SerialMon.print(topic);
  SerialMon.print("]: ");
  SerialMon.write(mqtt_buffer);
  SerialMon.println();
  // Only proceed if incoming message's topic matches
  if (String(topic) == topicGPS) {
    ledStatus = !ledStatus;
    digitalWrite(LED_PIN, ledStatus);
    //mqtt.publish(topicGPSStatus, ledStatus ? "1" : "0");
  }
}

void setup() {
  delay(5000);
  // Set console baud rate
  SerialMon.begin(115200);
  delay(10);

  pinMode(LED_PIN, OUTPUT);

  // !!!!!!!!!!!
  // Set your reset, enable, power pins here
  // !!!!!!!!!!!

  SerialMon.println("Wait...");

  // Set GSM module baud rate
  TinyGsmAutoBaud(SerialAT, GSM_AUTOBAUD_MIN, GSM_AUTOBAUD_MAX);
  // SerialAT.begin(9600);
  delay(6000);

  // Restart takes quite some time
  // To skip it, call init() instead of restart()
  SerialMon.println("Initializing modem...");
  modem.restart();
  // modem.init();

  String modemInfo = modem.getModemInfo();
  SerialMon.print("Modem Info: ");
  SerialMon.println(modemInfo);

#if TINY_GSM_USE_GPRS
  // Unlock your SIM card with a PIN if needed
  if (GSM_PIN && modem.getSimStatus() != 3) { modem.simUnlock(GSM_PIN); }
#endif

#if TINY_GSM_USE_WIFI
  // Wifi connection parameters must be set before waiting for the network
  SerialMon.print(F("Setting SSID/password..."));
  if (!modem.networkConnect(wifiSSID, wifiPass)) {
    SerialMon.println(" fail");
    delay(10000);
    return;
  }
  SerialMon.println(" success");
#endif


#if TINY_GSM_USE_GPRS && defined TINY_GSM_MODEM_XBEE
  // The XBee must run the gprsConnect function BEFORE waiting for network!
  modem.gprsConnect(apn, gprsUser, gprsPass);
#endif


  SerialMon.print("Waiting for network...");
  if (!modem.waitForNetwork()) {
    SerialMon.println(" fail");
    delay(10000);
    return;
  }
  SerialMon.println(" success");


  if (modem.isNetworkConnected()) { SerialMon.println("Network connected"); }


#if TINY_GSM_USE_GPRS
  // GPRS connection parameters are usually set after network registration
  SerialMon.print(F("Connecting to "));
  SerialMon.print(apn);
  if (!modem.gprsConnect(apn, gprsUser, gprsPass)) {
    SerialMon.println(" fail");
    delay(10000);
    return;
  }
  SerialMon.println(" success");

  if (modem.isGprsConnected()) { SerialMon.println("GPRS connected"); }
#endif

  // MQTT Broker setup
  mqtt.setServer(broker, 1883);
  mqtt.setCallback(mqttCallback);

  pinMode(speed_signal_pin, INPUT_PULLUP);
  pinMode(limit_switch, OUTPUT);
  digitalWrite(limit_switch,HIGH);
  calibrated_speed=80;
  calibrated_frequency=53; //Frequency for Toyota
  calibrated_period=1000/calibrated_frequency;
  attachInterrupt(digitalPinToInterrupt(speed_signal_pin), get_frequency, RISING);



   Serial1.println("AT+CGNSPWR=1\r\n");
  delay(500);
  while(Serial1.available()){
    Serial.write(Serial1.read());
  }

  Serial1.println("AT+GSN\r\n");
  delay(500);
  char ch;
  int i;
  i=0;
  while(Serial1.available()){
    ch=Serial1.read();
    if(ch=='\r' || ch=='\n' || ch=='O' || ch=='K');
    else{
      imei[i]=ch;
      i++;
    }
  }
  Serial.write("IMEI: ");
  Serial.println(imei);
}

void loop() {
  // Make sure we're still registered on the network
  if (!modem.isNetworkConnected()) {
    SerialMon.println("Network disconnected");
    if (!modem.waitForNetwork(180000L, true)) {
      SerialMon.println(" fail");
      delay(10000);
      return;
    }
    if (modem.isNetworkConnected()) {
      SerialMon.println("Network re-connected");
    }

#if TINY_GSM_USE_GPRS
    // and make sure GPRS/EPS is still connected
    if (!modem.isGprsConnected()) {
      SerialMon.println("GPRS disconnected!");
      SerialMon.print(F("Connecting to "));
      SerialMon.print(apn);
      if (!modem.gprsConnect(apn, gprsUser, gprsPass)) {
        SerialMon.println(" fail");
        delay(10000);
        return;
      }
      if (modem.isGprsConnected()) { SerialMon.println("GPRS reconnected"); }
    }
#endif
  }

  if (!mqtt.connected()) {
    SerialMon.println("=== MQTT NOT CONNECTED ===");
    // Reconnect every 10 seconds
    uint32_t t = millis();
    if (t - lastReconnectAttempt > 10000L) {
      lastReconnectAttempt = t;
      if (mqttConnect()) { lastReconnectAttempt = 0; }
    }
    delay(100);
    return;
  }

  mqtt.loop();
  SerialMon.println("=== THE END ===");
  Serial1.println("AT+CGNSINF\r\n");
  delay(500);
  clear_gps_buffer();
  char ch;
  while(Serial1.available()){
    ch=Serial1.read();
    if(ch=='\r' || ch=='\n');
    else{
      gps_buffer[gps_buffer_pos]=ch;
      gps_buffer_pos++;
    }
  }
```

```
  current_speed=atoi(mqtt_buffer);
  if(current_speed<30) current_speed=20;
  current_period=(calibrated_period*calibrated_speed)/current_speed;
  Serial.print("Current period: ");
  Serial.println(current_period);
  Serial.print("Speed period: ");
  Serial.println(speed_period);
  clear_mqtt_buffer();



  strcpy(mqttString,imei);
  strcat(mqttString,"%");
  strcat(mqttString,gps_buffer);
  //mqtt.publish(topicGPS, "hello world");
  mqtt.publish(topicGPS, mqttString);
  Serial.write("MQTT String: ");
  Serial.println(mqttString);
  //digitalWrite(limit_switch,HIGH);
  delay(1000);
  //digitalWrite(limit_switch,LOW);
  delay(1000);
}
```