BEng. (Hons) Electronic and Electrical Engineering

MICROCONTROLLER BASED ACCIDENT ALERT SYSTEM

SADEEP DILSHAN KASTHURIARACHCHI

REG. NO: 1026262

SUPERVISOR: MR. ISURU B. SAMARAKOON

This report is submitted as a partial fulfillment in module 6355SLT – Engineering Project

January 2023

ACKNOWLEDGMENT

In order to make this project a success, I would like to sincerely thank Mr. Isuru Bandara Samarakon, my project supervisor, for his kind assistance, valuable time, and continuous guidance. Furthermore, his oversight wisdom, experience and counsel helped my project succeed in a variety of ways.

And I want to convey my utmost appreciation to Prof. Indra Dayawansa, the department head of Electronic and Electrical Engineering at SLIIT Academy, as well as the entire SLIIT Academy personnel and our esteemed lecture panel.

Additionally, I deeply appreciate Liverpool John Moors University for offering us this wonderful opportunity and for generally supporting this project to ensure its success. Most importantly, I want to take this chance to express my gratitude to my parents for their unwavering support, wisdom and moral compass. I want to express my gratitude to my friends and to everyone else who helped me throughout the years to complete this project.

ABSTRACT

Road accidents may occur as a result of the development of in-vehicle technology and the infrastructure for transportation, as well as the explosive growth in the number of both commercial and non-commercial vehicles on the road. These accidents typically result in a high death toll. The reason for more than half of these fatalities is a slow reaction time by rescue teams and medical professionals. When a car is involved in an accident, this project suggests a system as a remedy.

The proposed systems have been realistically designed and virtually tested using hardware components, and the outcomes meet expectations. An accelerometer is used to detect any abrupt changes in the vehicle's axes, and a vibration sensor is used to measure the vibration levels of the vehicle. A dangerous flame is identified by also keeping track of the information the flame sensor has collected. The location of the accident and the nature of the event (fire, crash) are included in the alert message that the GSM module delivers to user mobile device. The latitude and longitude from the GPS module are used to create a Google Map link that shows the accident's location. A reset button (switch) is also available to stop the transmission of a message in the unlikely event that there are no casualties. This can help the ambulance avoid wasting valuable time.

The emergency rescue units can therefore immediately trace the location using Google Maps after getting the information. Following confirmation of the location, the required action will be performed.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

The fast change in human civilization over time is due to the development of the transportation system. The development of transportation has greatly facilitated communication and given it a high priority in our everyday lives. But it may also contribute to the loss of life and property. Approximately 1.25 million people die in road accidents each year, or 3,287 people each day, according to the Association for Safe International Road Travel [1]. In addition, 20 to 50 million people worldwide suffer from disabilities. The ninth most common cause of fatalities is automobile accidents. Road traffic accidents cost between 1% to 2% of each country's annual GDP and result in damages of USD $518 billion annually.

A road crash is defined as any collision involving on-road vehicles, obstacles, or pedestrians. The most obvious cause of a person dying after an accident is the absence of first assistance because it takes time to get precise information about the accident location. The likelihood that an injured person will survive the accident depends critically on how quickly an ambulance can get to the site and take them to the hospital. In the majority of cases involving road accidents, the injuries are modest, and the victim's life may be preserved; nevertheless, because rescue crews frequently arrive late, the injuries often prove deadly. In order for the rescue teams to be able to take the required steps to preserve the victim's life, the main goal is to locate the scene of the accident and quickly provide them with the information they need [2].

The insignificant causes of the lack of timely access to on-site medical aid are delayed accident reporting, inaccurate geolocation [3], and unreliable mobile applications caused by broken smartphones [4]. Eyewitnesses frequently fail to alert the hospital and police when an accident occurs. Additionally, when an accident occurs on a roadway, passing automobiles frequently fail to alert the hospital or the police as required by law [5]. As a result, this issue requires rapid attention, and automated accident detection and location tracking systems should be developed that don't need human participation.

This project describes a system, with accident detection as its primary use. At first, the GPS records the latitude and longitude values it continuously gets from the satellite. Sending a message to the

GSM device activates it, allowing us to track the vehicle. It is also activated by the IMU module and knock sensor, both of which are attached to the Arduino board, detecting an accident, and the flame sensor, which is also connected to the Arduino board, detecting a fire. A Reset switch (button) is also included. This command is used to halt the sending of a message in the uncommon situation where there are no casualties.

The proposed system has the potential to identify accidents in a fraction of the time and provides basic data to a first-aid facility in a matter of seconds, including geographic coordinates, the type of event (fire, crash), and a link that opens right away in Google Maps. The central emergency dispatch server receives this warning message quickly, which enables the emergency dispatch server to alert surrounding ambulances and maybe save crucial lives. As another side application of this system, after the accident, according to the order in which the sensors started working, it is determined whether the accident was due to the carelessness of the driver or not. It is sent with the above-mentioned SMS message. This system provides the greatest remedy for inadequate emergency services provided in the event of a road accident.

1.1 LITERATURE REVIEW

The literature study makes it abundantly clear that significant research has been done to identify accidents and alert the emergency medical services. In [6] the vehicle's speed was tracked using a GPS receiver, and an accident was detected by comparing the prior and present speeds of the cars. The location of the accident was relayed to an Alert Service Center by this mechanism. However, the distance between the accident site and the Alert Service Center may influence the amount of time it takes to rescue or send help.

While an accident was identified, the authors [7] recommended a system that contacted emergency contacts, police, or ambulance services. The technology used Fuzzy Logic to train a smartphone for decision support and then stored recordings in the data center for future use. Accident data was gathered using an accelerometer, gyroscope, and four force sensors mounted on each side of the car. However, the system hardware proved too bulky due to the numerous sensors. Even though the system

sent a text message to the emergency contact or public safety, the response time is affected by the distance between the accident site and the public safety service center.

The authors of [8] used a shock sensor to identify an accident and communicated the passengers' full names, blood types, phone numbers, emails, medical histories, dates of birth, and reference phone numbers to the Public Safety Organizations' headquarters. The system's fundamental flaw is that all passengers and cars must be pre-registered. Furthermore, the distance between the Public Safety Organizations' headquarters and the accident site determined the system's success.

The crash detection system was created by M. S. Amin [9] using GPS data and determining the deceleration. This technology only uses GPS connectivity to detect accidents, which makes it less dependable in areas with dense canopy, subterranean tunnel systems, and low GPS connectivity. Another technique created by J. Zaldivar suggested using the On Board Diagnostics (OBD) system to monitor speed, engine status, and airbag deployment to identify crashes. This increases the system's dependability because the majority of airbag deployment systems utilized by contemporary automakers have undergone extensive testing and provide a clear indicator when a fatal crash will occur. However, only new, high-tech cars that come standard with an airbag system and onboard diagnostics can use this system.

Another strategy employed by H. Sharma [10] makes use of contemporary smartphone sensors to find an accident and alert the EMS online. A system like this provides a straightforward and economical means of reaching the goal. However, the smartphone sensor's dynamic range and sampling rate are constrained. In other circumstances, it is also impractical to utilize the application because it must be launched each time the driver embarks on a journey. The method is less dependable than a dedicated stand-alone system designed especially for accident detection because of the possibility that the smartphone will malfunction owing to battery problems.

[11] In here proposes an intelligent accident detection system based on the vehicle's tilt and aberrant passenger heartbeat rates. If the threshold value was surpassed, the data from an accelerometer and heartbeat sensor identified an accident and sent SMS to the phone numbers registered in the smartphone. It also looked up the contact information for neighboring emergency medical assistance.

Sending SMS to all of the contacts contained in the smartphone, on the other hand, would be redundant and raise system overhead. The possibility of smartphone problems due to power issues may make the system unstable, and the requirement to activate the smartphone application every time the driver starts the journey is nonsensical [8].

IoT and cloud computing are used in other existing system [12]. SVM (support vehicle machine) developed by Ant Colony Algorithm is used to detect vehicles (ACA). Magneto resistive sensors will be used to monitor the automobiles through IOT. This project's major goal is to distinguish between traffic and non-traffic incidents.

To keep the standard consistent across all vehicles, a stand-alone system is required. Without requiring significant car modifications, the standalone system must be affordable and compatible with older and less expensive vehicles.

The aforementioned literature evaluations clearly demonstrate that there is still a scope in this area even when the victim is good and does not need all of these services because all approaches are detecting the accident and also reporting it to the respected ones. In addition to the accident detection and reporting system, the project "Vehicle Accident Alert and Tracking System" has been offered to prevent such situations.

1.2 AIM

The main aim of this project is to build a system which can detect an accident, analyze it, and send the message alert to the fire & rescue unit or emergency rescue service center.

1.3 OBJECTIVES

- Identifying the suitable sensors required for vehicle accident alert and tracking system.
- Design and install a vehicle unit with a sensor system that identifies the location of an accident, nature of the event (fire, crash) and alerts the necessary services at the right moment.
- GSM notification mechanism design and implementation.
- To handle the cancellation of the accident alarm, design and deploy wired switch.
- Carry out Kalman filter simulations in MATLAB for different true-rotations and noise levels for both the accelerometer and gyroscope in the IMU module.

2. METHODOLOGY

This section explains the proposed system's architecture and functionality The projected system contains three major parts Accident Detection (AD), Location Tracking (LT) and Smart Reporting System (SRS).

2.1 ACCIDENT DETECTION (AD)

The Knock sensor, IMU module, and Flame sensor are the three primary sensors used in Accident Detection (AD). These sensors are installed in the car and are used as input parameters for Accident Detection (AD). They are sent to the main system, which includes an Arduino Nano board, GPS, and GSM.

IMU (inertial measuring unit) is a collection of sensors that monitor specific force and rotation rate and sends a signal to the microcontroller if it detects any unusual force or rotation. The knock sensor is employed here to detect a vehicle collision. When the object's Strike exceeds the critical value, the Knock sensor produces an output voltage. As a result, when a low signal output is generated, it is sent to the microcontroller. If a fire breaks out at the accident scene, the flame sensor will detect it and send a signal to the microcontroller.

As another side application of this system, after the accident, according to the order in which the sensors started working, it is determined whether the accident was due to the carelessness of the driver or not.

If the system detects a false accident (regarding accidents nature), a 05 to 15-second override option (switch) will be provided to prevent the system from sending an alert.

2.2 LOCATION TRACKING

The GPS sensor can determine the vehicle's present location. The GPS device is used in this suggested system to identify the exact site of the accident. When the microcontroller receives an accident signal, it asks the GPS for the current location of the accident site. The microcontroller receives the position of the accident through GPS.

2.3 SMART REPORTING SYSTEM (SRS)

GSM delivers a text message to the fire & rescue unit or emergency rescue service center room when an accident occurs. A message will be sent to the fire & rescue unit and emergency rescue service center, including geographic coordinates, nature of the event (fire, crash), driver carelessness message and a link that opens instantly in Google maps. Simultaneously, the nearest emergency rescue service center receives an accident notification message with a link to a Google map. It is sent with the mentioned SMS message. With these facts, the ambulance may take the quickest route to the accident site and save the lives of the victims in less time.

2.4 PICTORIAL REPRESENTATION OF PROPOSED SYSTEM

2.4.1 ARCHITECTURE

Figure 2.4.1.1 provides an illustration of the accident detection and alert system's design. The system makes use of several sensors for data collecting, including an IMU sensor, knock sensors, and flame sensors. A microcontroller processes and categorizes the events using the collected data. The microcontroller uses a GSM transmitter to convey an emergency alert in the event of an accident.

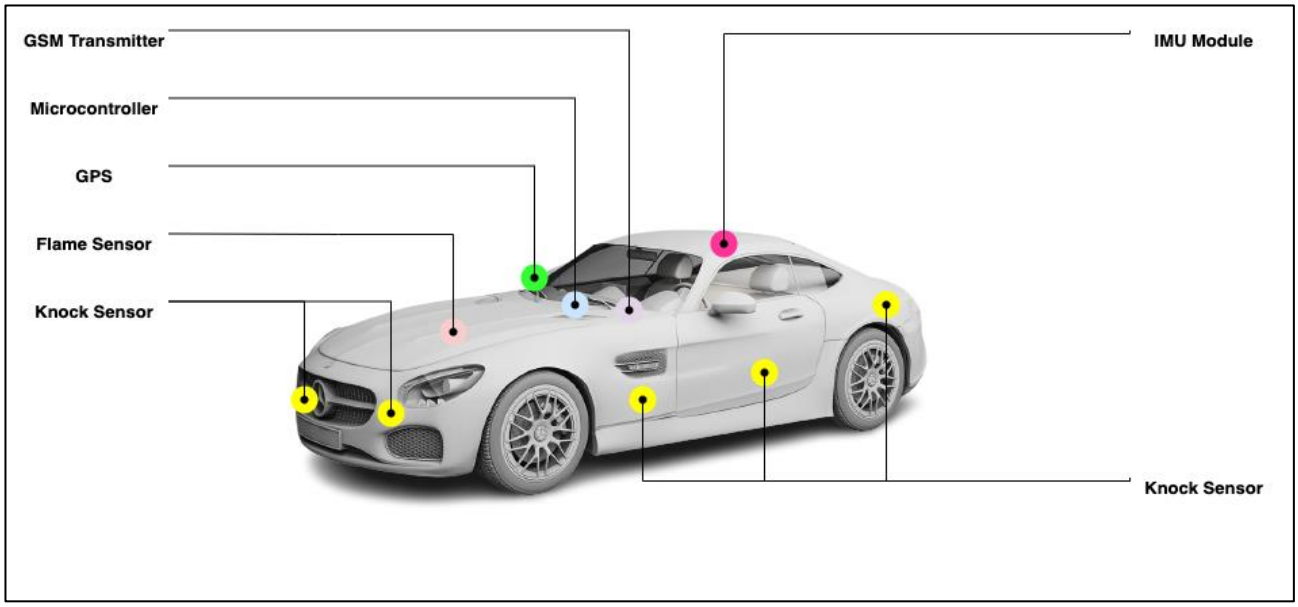


Figure 2.4.1.1: Architecture of proposed system

2.4.2 PICTORIAL REPRESENTATION

Accident detection systems determine which services (Emergency Rescue Unit, Fire & Rescue Unit) should be dispatched to the accident area based on which sensors are triggered. GPS locates the accident's coordinates, and GSM provides this position connection via message to the emergency rescue unit or fire & rescue unit for possible fire rescue. Figure 2.4.1 shows a pictorial representation of the proposed system.

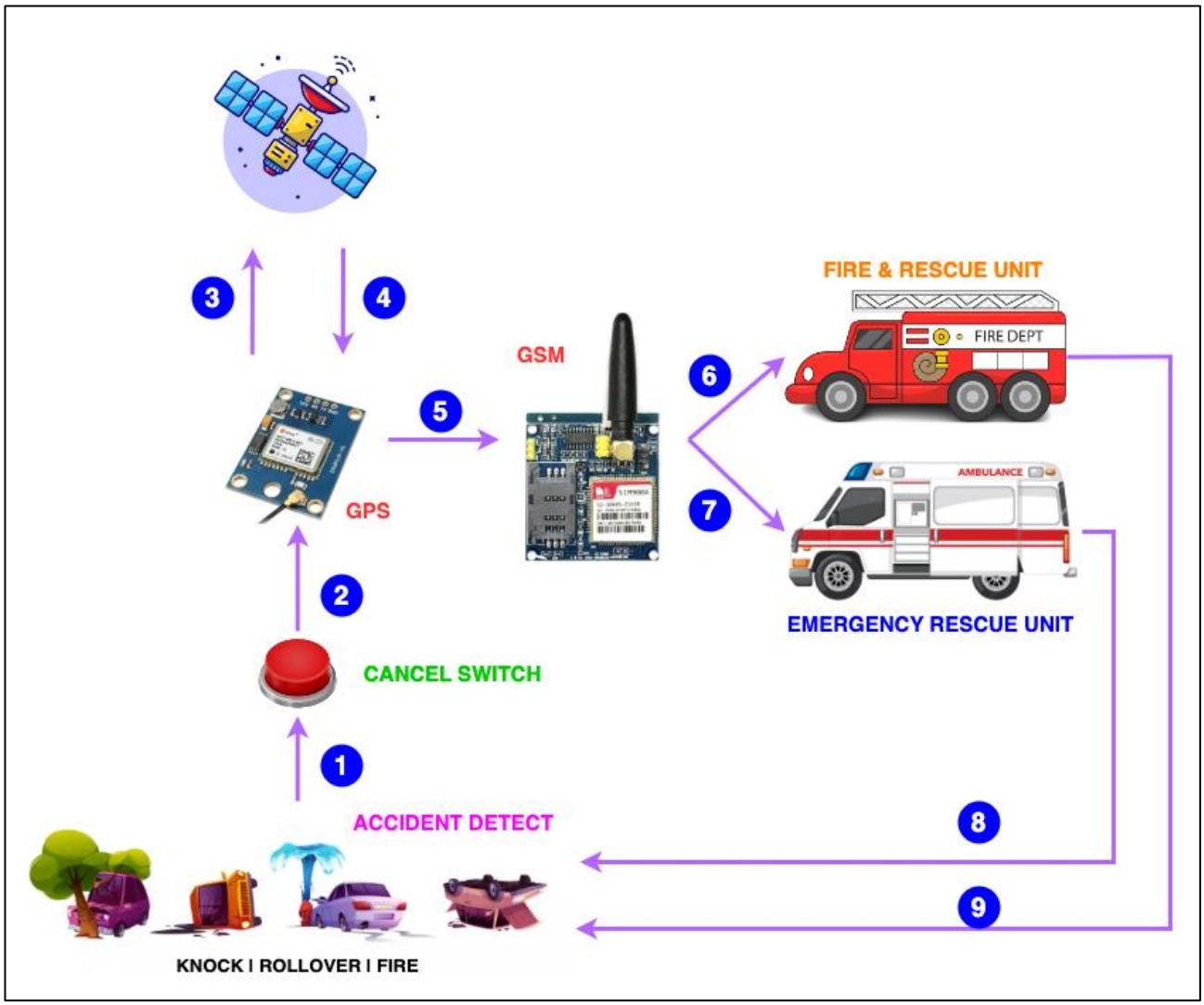


Figure 2.4.2.1: Pictorial Representation of System

2.5 BLOCK DIAGRAM

Figure 2.5.1 illustrates the system's basic block diagram. The Arduino NANO board is used as the primary microcontroller in this system, which is designed for accident detection and will be

implemented in the vehicle itself. The GPS latitude and longitude of that specific location are also obtained, allowing the exact location of the accident site to be determined. In this case, the GSM modem is linked to a microcontroller. As a result, if an accident occurs, the SMS will be sent automatically to the numbers specified in the code. Also, if the accident is normal, the delay time to give the person a chance to press the switch.

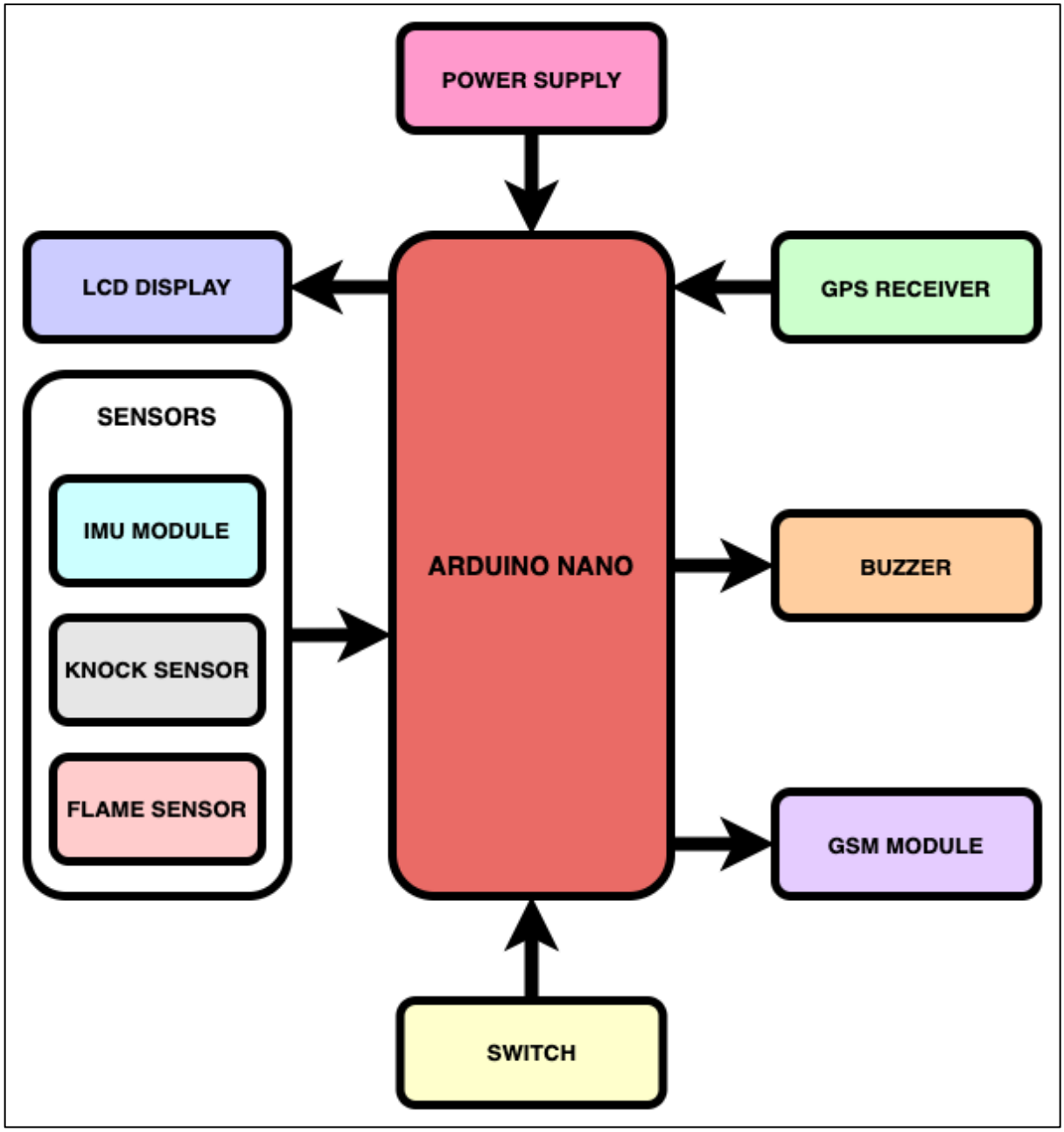


Figure 2.5.1: Block Diagram

2.6 FLOW CHART

Figure 2.6.1 shows the algorithm's flowchart. The algorithm provides a brief overview of how the proposed system works; it begins by detecting the accident via the Knock sensor, IMU module, and Flame sensor; once detected, it checks whether any of the notification cancel switches are pressed within 05 to 15 seconds; if pressed, the service to which the message is about to be sent is canceled; otherwise, notification is sent to the respective services automatically.

In more detail, all the sensors are initialized and updated on a regular basis. The emergency alert is transmitted to the emergency services and the nearby vehicles if the collision detection is greater than the predetermined threshold. In contrast, the system will get data from the IMU sensor if the knock sensor's indication of a collision does not reach the threshold value. An emergency alert is sent if the sensor circuitry notices a knock that is more intense than the threshold value. The system will use the data from the flame sensor if there is any flame. Flame sensor only works if one the above sensors are triggered. The alert message will be delivered if the above conditions meet. An emergency alert is sent if the driver doesn't switch off the alarm since it is thought that they are not in a fit state to do so. In order to include location coordinates in an emergency message, the GPS module must first be accessed. Also, according to the order in which the sensors started working, it is determined whether the accident was due to the carelessness of the driver or not and this data also enclosed in the emergency message.

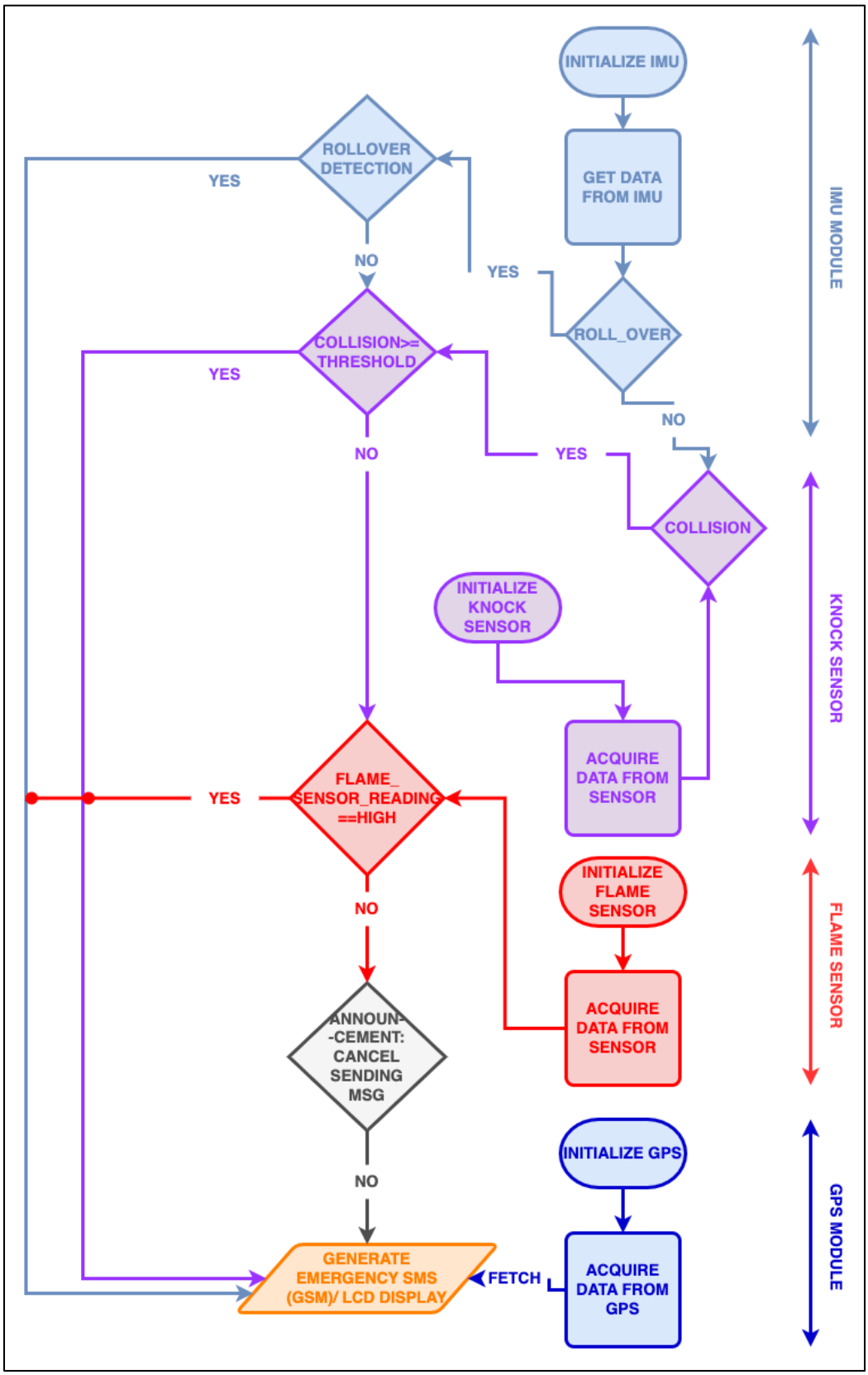


Figure 2.6.1: Flow Chart

2.7 HARDWARE SET-UP

This accident detection, location tracking, and notification system's circuit connection, which is illustrated in fig. 2.5.1, is quite simple. The D3 pin of the Arduino NANO is directly connected to the Tx pin of the GPS module [Appendix 1,2]. Here, every module is powered by a 5-volt source. The TxD and RxD pins of the GSM module are directly linked to pins D5 and A1 on the Arduino NANO. Use a software serial library as well for GSM connectivity. A 5v supply is also used to power the GSM module. The data pins SDA and SCL of an optional LCD are connected to MPU 6050 pins SDA and SCL, and then they are connected to pins A4 and A5 of the Arduino NANO. The Arduino NANO's pins A2 and D6 are connected to the vibration sensor and flame sensor, respectively. Finally, the buffer pin is connected to Arduino NANO pin D11.

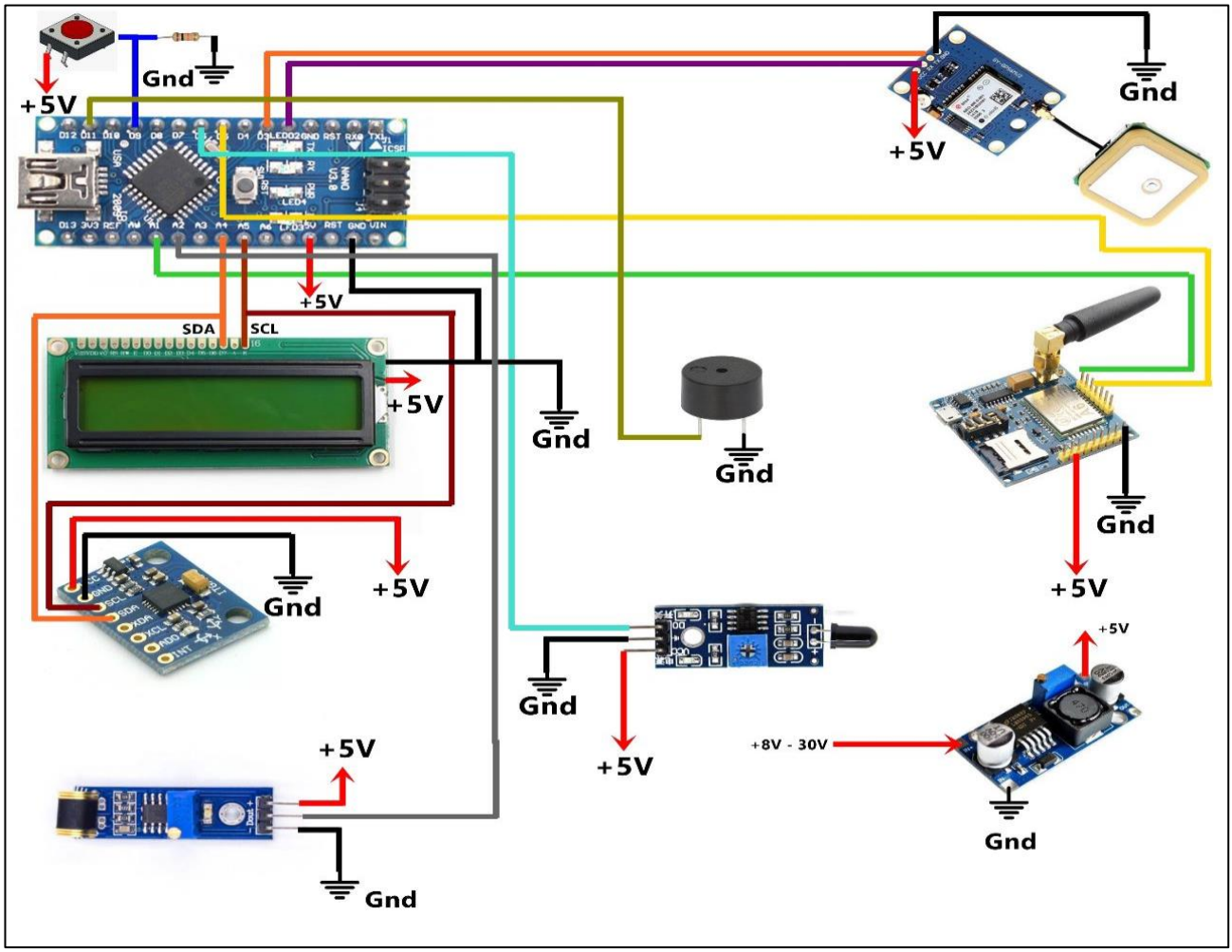


Figure 2.5.1: Circuit Diagram of proposed system

2.8 ROTATION AND ALTITUDE

By calculating the rotation and altitude, an accelerometer was used to detect the rollover and fall-off of a vehicle because of an accident. Any accelerometer's data sheet identifies the positive x, y, and z axes on its packaging. Accelerometers measure the discrepancy between an object's linear acceleration and the gravitational field in the immediate area. Therefore, a linear acceleration along these axes produces a positive accelerometer output, whereas a gravitational field component along the same axes produces a negative output. The employed coordination in our system is depicted in Figure 2.8.1. While the z-axis is in line with gravity, the x-axis runs parallel to the vehicle's body axis. Furthermore, the x and z axes are perpendicular to the y-axis. In order to estimate the rollover occurrence, the roll φ and pitch θ angles of a vehicle around the x and y axes are calculated, and a rollover occurs when roll φ or pitch θ angles are greater than 90^{o} (chapter 2.9.4).

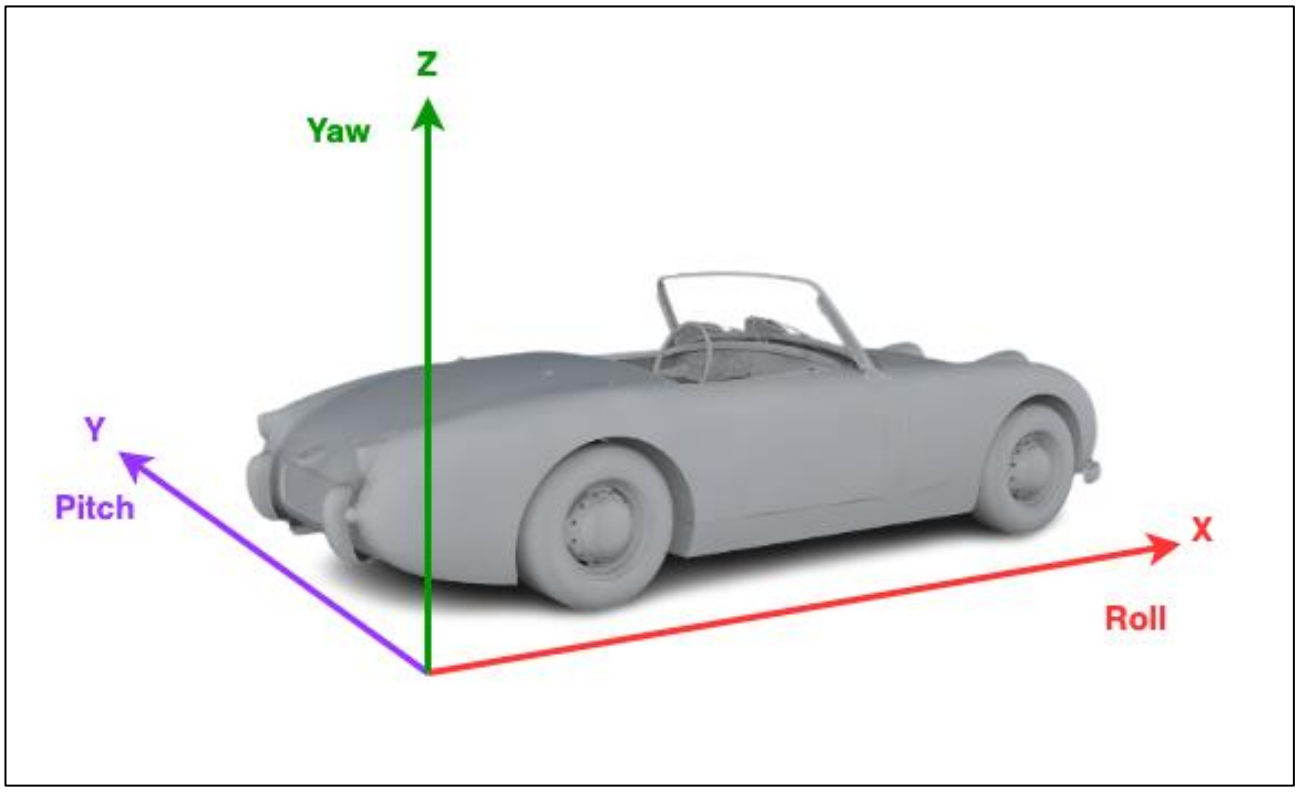


Figure 2.8.1: Coordination & Rotation axes

2.9 ACCELERATION & GYROSCOPE CALCULATIONS

2.9.1 ACCELERATION [13]

The ability to measure acceleration is provided by an accelerometer. The phenomena of weight being felt by a mass that is in the reference frame of the device causes this acceleration, which is not always the same as an object moving through space. In other words, it detects how much a mass presses against something when a force acts on it in order to determine acceleration by the measurement of the force. For instance, when the sensor is still on the surface of the Earth, it will detect an acceleration of 9.81 [m/s^2] caused by gravity.

With the development of technology, GPS's accuracy and dependability have grown while its cost and size have decreased. However, it still has issues with line of sight and update pace. Because of the slow update rate, the acceleration data from GPS is lacking in immediate acceleration. This is crucial in figuring out whether an abrupt slowdown was caused by accident. As opposed to GPS, the IMU can be used to capture continuous and instantaneous data at high data rates.

An accelerometer monitors forces that create acceleration, which can be either a dynamic force brought on by motion or vibration, or a static force like the constant force of gravity. Equation can be used to simulate the output of a rigid body-mounted accelerometer (1).

$$y_a = \frac{1}{m}(F - F_g) - - - - - - - - - - - (1)$$

Where y_a is the measured acceleration, m is the mass, F_g is the force due to gravity, and F is the sum of all forces.

2.9.2 GYROSCOPE [13]

The various rotation angles of structures can be measured by gyroscopes. Most frequently, measurements of rotational angular speeds are made with a predetermined frame of reference. Degrees per second (^{o}/s) or revolutions per second (RPS) are the units used to express the rates. The information required for the stability and orientation of the measured object is provided in the output.

2.9.3 ACCELEROMETERS & GYROSCOPES ERRORS [13]

- Constant Bias: The output average while the gyroscope is not rotating in /h. The output signal's integration results in an angular inaccuracy that increases linearly with time.
- Thermo-Mechanical White Noise: The output signal will be disturbed by a sequence of zero-mean, uncorrelated random variables called thermomechanical white noise, each of which is identically distributed and has a finite variance σ^2.
- Flicker Noise: The result of this inaccuracy is that the gyroscope's bias wanders over time. At high frequencies, the white noise overpowers the flicker noise, making it less visible.
- Calibration errors: It speaks about inaccuracies in the gyros' linearities, scale factors, and alignments. These mistakes only become apparent when the device is spinning and cause the drift in the integrated signal to accumulate.

2.9.4 ORIENTATION CALCULATION [15]

The usage of the three Euler Angles, which go through an orderly succession of turns indicating the movement from one coordinate system to another, is one of the most prominent methods for specifying the angular orientation of one coordinate system regard another [15].

As shown in Fig. 2.9.4.1 [15], the angles phi, theta, and psi correspond to the typical angles of roll (φ), pitch (θ), and yaw (ψ) that are employed in navigation to indicate the attitude of a mobile in the axes X, Y, and Z, respectively.

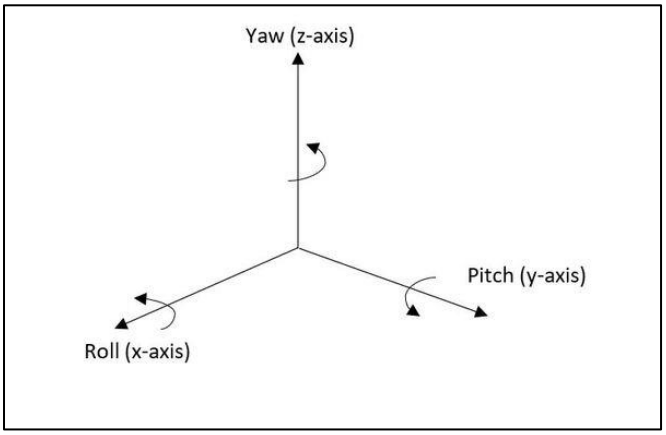


Figure 2.9.4.1: Roll (φ), Pitch (θ), and Yaw (ψ)

It is possible to rebuild a specific vector specified by the bases of a reference system A, based on vectors that comprise a reference system B, using rotation matrices. For a three-dimensional reference system with X, Y, and Z axes, we can build a 3x3 rotation matrix for each base vector, allowing us to describe the system's rotation around that axis of rotation.

The three rotations are theoretically described as three different direction matrices [15]:

$$\text{Roll } \varphi, \text{C1: } \begin{matrix} 1 & 0 & 0 \\ 0 & \cos(\varphi) & \sin(\varphi) \\ 0 & -\sin(\varphi) & \cos(\varphi) \end{matrix}$$

$$\text{Pitch } \theta, \text{C2: } \begin{matrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{matrix}$$

$$\text{Yaw } \psi, \text{C3: } \begin{matrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{matrix}$$

A transformation from reference to body axes can be described as the following multiplication of three matrices:

$$C_n^b = C_1 C_2 C_3 \quad ------[1]$$

The inverse transformation from the body to the reference axes is given by:

$$C_b^n = C_n^{bT} = C_3^T C_2^T C_1^T \quad ---[2]$$

2.9.5 KALMAN FILTER

Despite being more commonly recognized as a sensor fusion, it is implemented as a discrete accelerometer noise filter that uses the gyroscope and accelerometer readings to prevent drift.

The Kalman filter algorithm, as it is provided in this project, guesses the statement which is desired in discrete points of time. Since repeating is a part of the process, the search value is quickly estimated when it is masked by uncertainties and random variations by using various equations and the entry of successive values. Because it supports estimates of the past, present, and even future states, the filter is particularly effective. Consequently, the Kalman filter involves two steps [14]:

1) Prediction
2) Correction

The dynamic model is used to anticipate the state in the first stage. The error covariance of the estimate is reduced in the second stage by correcting it using the observation model. It serves as the best estimate in this regard [14].

1) Prediction

$$\hat{x}_k = A\hat{x}_{k-1} + Bu_k + w_{k-1} - - - - - (1)$$

In Eq. 1, $\hat{x}_k$ is calculated from A, which is equal to 1, the filter's previous output (which is set as zero), the multiplication of B and the optional control input, which in this situation is zero, therefore B is set to zero. The perturbation process's representation w_{k-1} is also set to zero.

$$P_k = AP_{k-1}A^T + Q - - - - - - - - (2)$$

In Eq. 2, P_k is first set up as an identity matrix, and Q is set to 13^2.

2) Correction

$$K_k = P_{k-1}H^T(HP_{k-1}H^T + R)^{-1} - - (3)$$

The Kalman gain at the k instant is determined in Eq. 3 from P_{k-1}, H, and R. H is set to 1 to reflect the relationship between the model and the system's state, and R is set to the value 13.

$$\hat{x}_k = \hat{x}_{k-1} + K_k(z_k - H\hat{x}_{k-1}) - - - (4)$$
$$z_k = H_k x_k + v_k - - - - - - - - - - (5)$$

In Eq. 3.4, $\hat{x}_k$ is computed from the previous output $\hat{x}_{k-1}$, the Kalman gain K_k, H_k and the new measurement from the sensor z_k, which is acquired from Eq. 5, where v_k is the value associated with the measurement noise, which in this case is equal to zero.

$$P_k = (I - K_k H)P_{k-1} - - - - - - - - (6)$$

Finally, P_k in Eq. 6 is changed from K_k and P_{k-1}. There are a few conditions for the algorithm:

- $\hat{x}_k = x_0$
- $P_{k-1} = P_0$
- Q and R, whose values are determined through trial and error after an initial suggestion.

2.9.6 APPROACH

The Kalman Filter techniques were first implemented in a MATLAB simulation. Then, during a ten-second period, define a true rotation for the yaw, pitch, and roll angles. The predicted gyroscope and accelerometer data were then derived for every millisecond from the true-rotation, resulting in a six-element measurement model with 10,000 accurate measurements (figure 2.9.6.1). The gyroscope and accelerometer measurements were then subjected to random, normally distributed zero mean noise. Based on these measurements, the anticipated angle for the Kalman filter was calculated based on accuracy and noise reduction.

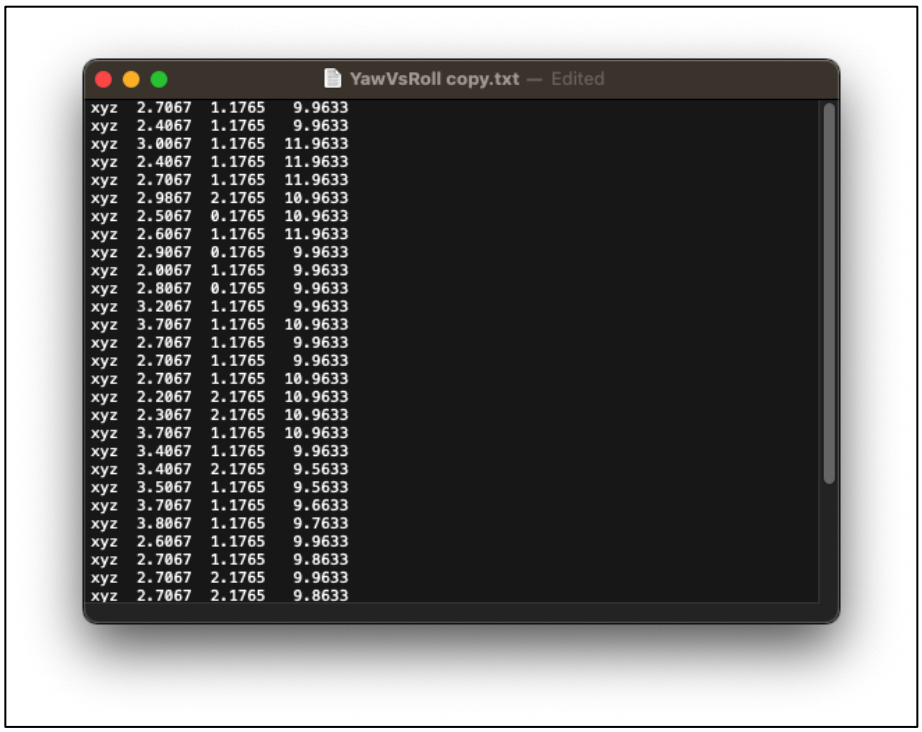


Figure 2.9.6.1: Accelerometer & Gyro Data

As a second stage, create a filter on an Arduino Nano using real accelerometer and gyroscope data, and compare the output to the accelerometer values. The amount of latency accumulated by the Kalman filter computation was also measured.

For the prediction step of the Kalman filter, the following equations were utilized [chapter 2.9.5].

$$\hat{x}_k = A\hat{x}_{k-1}$$

$$P_k = AP_{k-1}A^T + Q$$

2.9.7 EVALUATION

The figures below show a variety of MATLAB simulation cases. The responsiveness of the filters to varying true-rotations and noise levels was examined here for both the accelerometer and gyroscope.

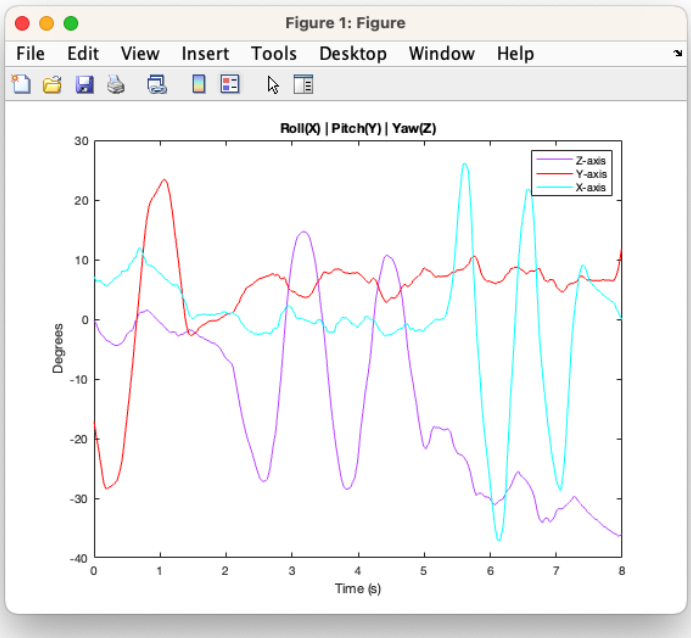


Figure 2.9.7.1: Plot of Roll, Pitch and Yaw from MPU6050

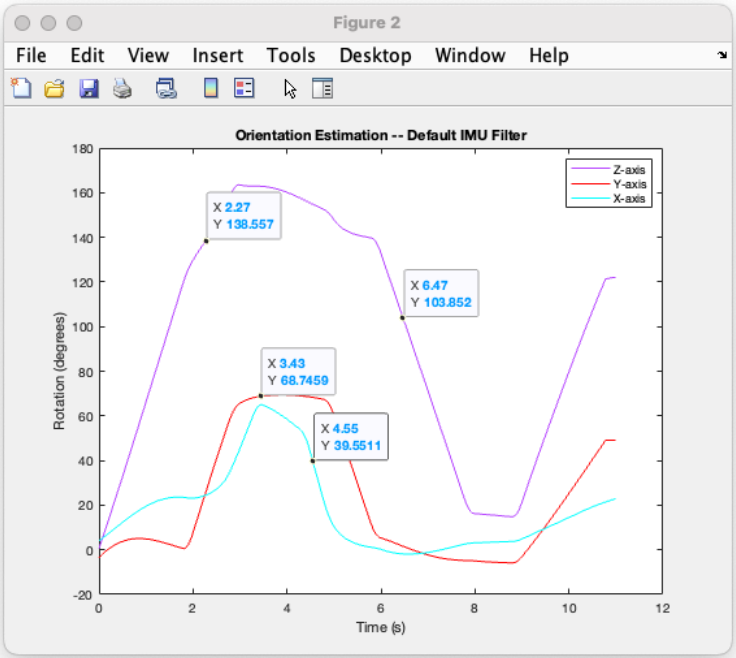


Figure 2.9.7.2: Default Filter in IMU-6050

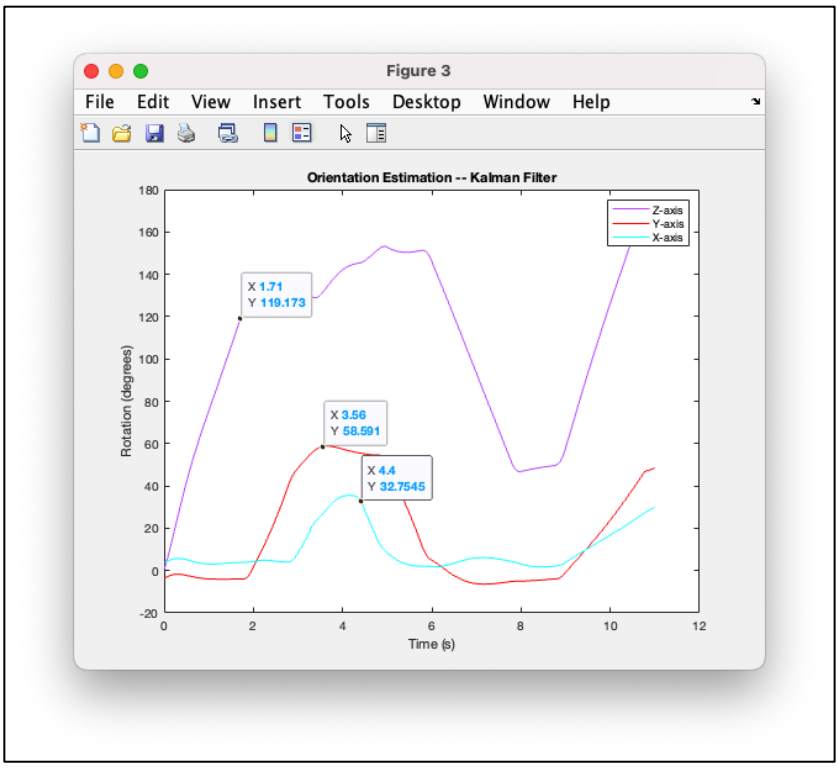


Figure 2.9.7.3: Kalman Filtered IMU6050 data

The implementation shows the Kalman Filter's benefits and drawbacks. The output of the Kalman Filter is smoother than the output of the default filter in the IMU Module, as seen in Figure 2.9.7.3. However, while the Kalman Filter is smoother than the default filter in the IMU Module, it has a higher latency. The general trend of the accelerometer data is followed by both filters. For yaw implementation, the gyroscope integration is used.

3. RESULTS & ANALYSIS

3.1 DETECTING AN ACCIDENT

Here, a specific amount of time is always given when an accident is detected by this system to notify the relevant parties, and that amount of time varies depending on the accident. Chapter 3.1.1 goes into further detail about this. The system's operational situations are shown in table 1 below.

Table 1: Operational Conditions

A	B	C	F
0	0	0	**0**
0	0	1	**0**
0	1	0	**1**
0	1	1	**1**
1	0	0	**1**
1	0	1	**1**
1	1	0	**1**
1	1	1	**1**

0	OFF
1	ON

A → Knock Sensor

B → IMU Module

C → Flame Sensor

F → Buzzer/SMS Alert

3.1.1 TIME DELAY

The delay time mentioned above is further discussed here. Use the following table to understand this more easily. In this table, the time varies according to the nature of the accident, i.e., the number of sensors that are triggered.

Knock Sensor (Collision)	IMU Module (Rollover)	Flame Sensor (Fire)	Time Delay
Yes	No	No	**15s**
No	Yes	No	**15s**
Yes	No	Yes	**10s**
No	Yes	Yes	**10s**
Yes	Yes	No	**05s**
Yes	Yes	Yes	**05s**

Table 2: Time Delay

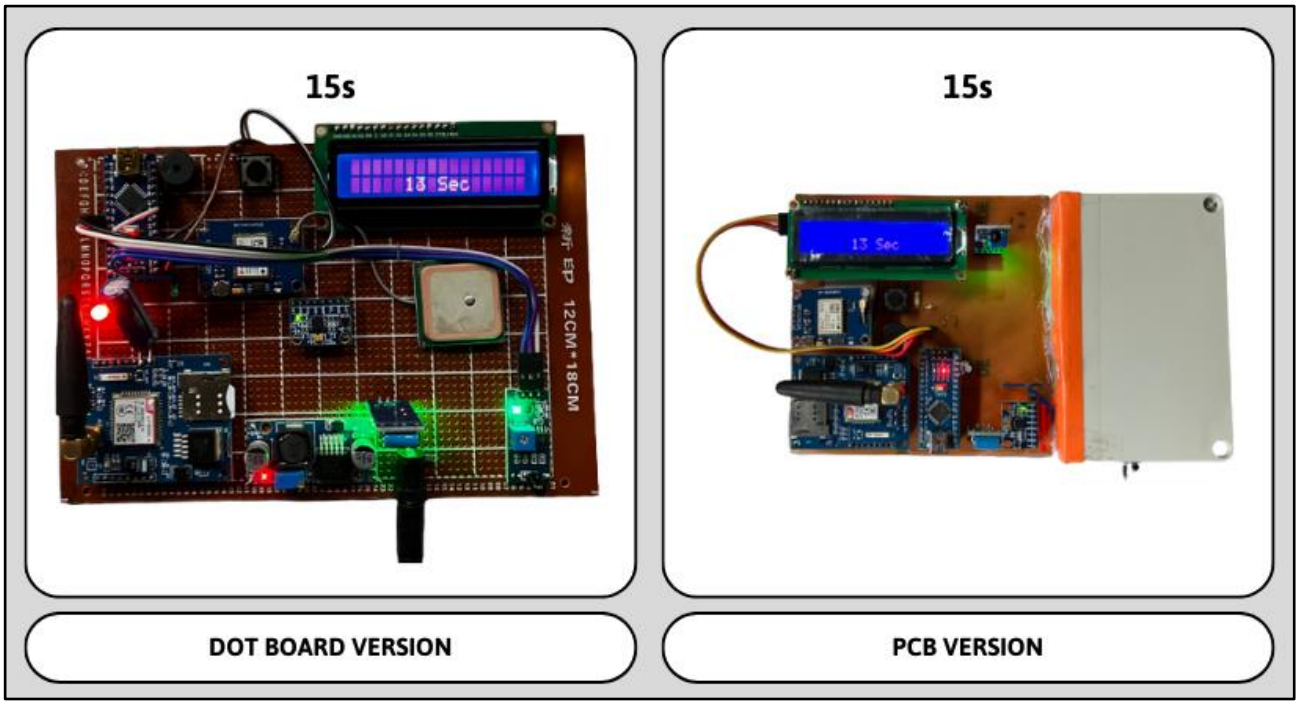


Figure 3.1.1.1: Time Delay (15s)

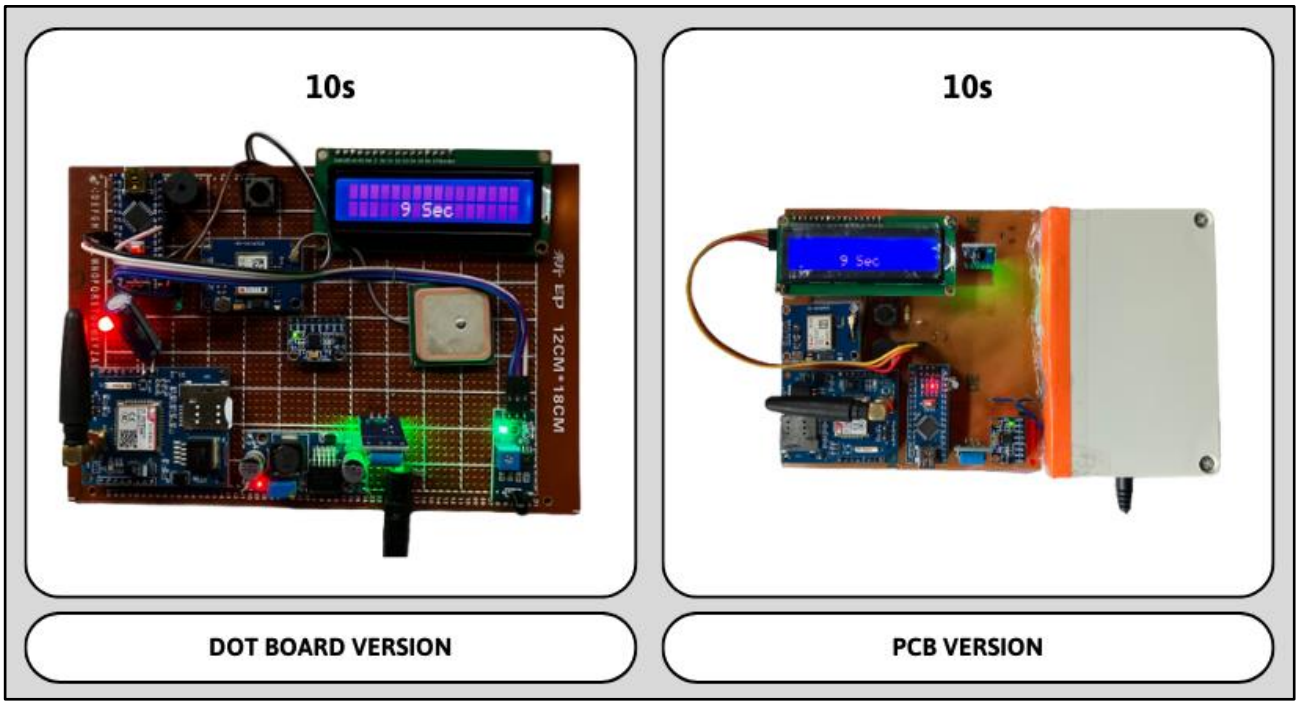


Figure 3.1.1.2: Time Delay (10s)

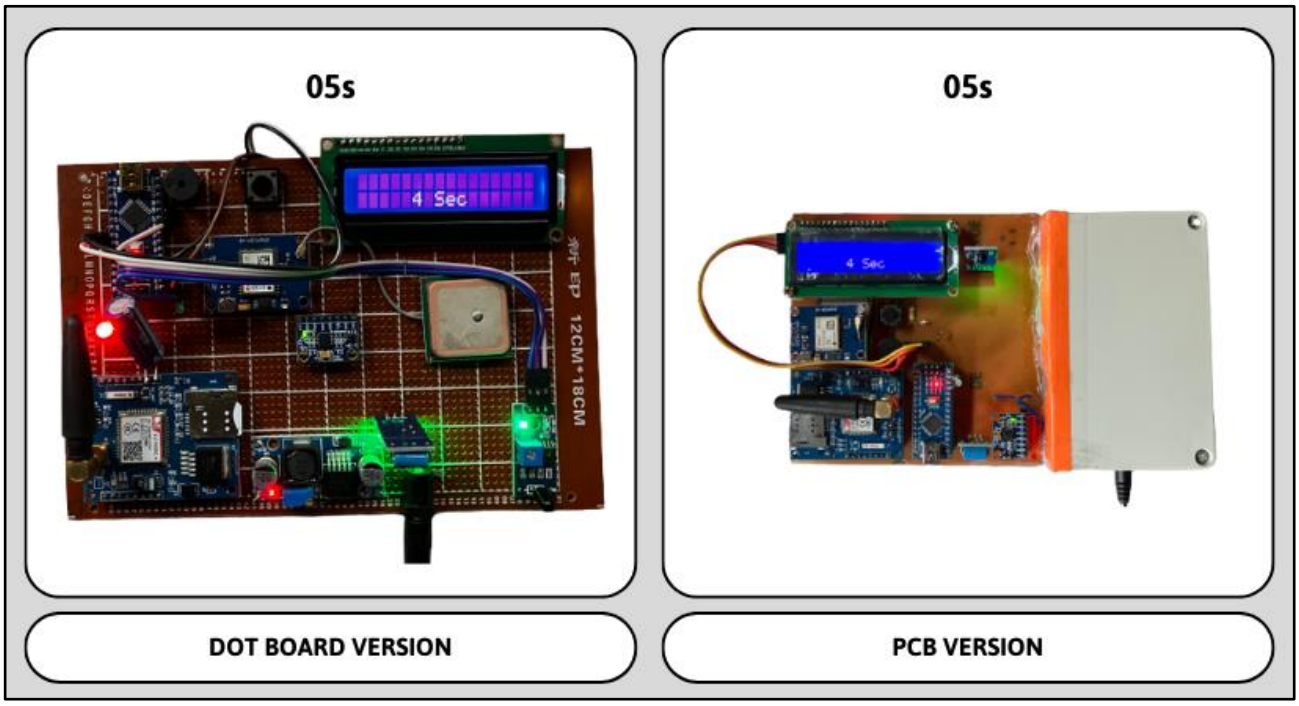


Figure 3.1.1.3: Time Delay (05s)

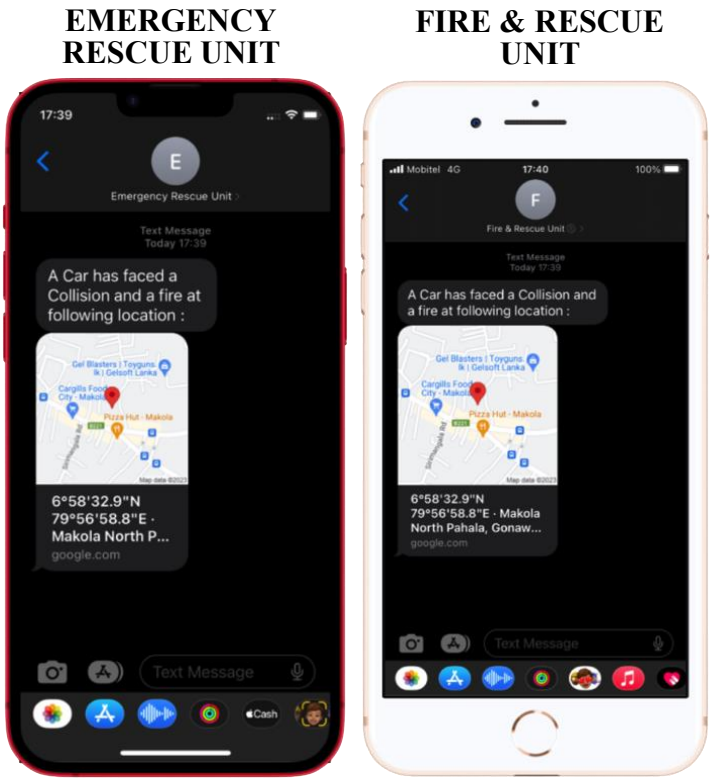
3.2 SEND MESSAGES USING GSM MODULE

3.2.1 NORMAL ALERT MESSAGES

Here, the accident's details are communicated to the relevant parties. This is where the accident and its location will be sent. The trigger sensor determines the type of accident. Another peculiarity here is whether the accident occurred due to the negligence of the driver, which is also sent along with this message. Under chapter 3.2.2, this will be covered in more detail.

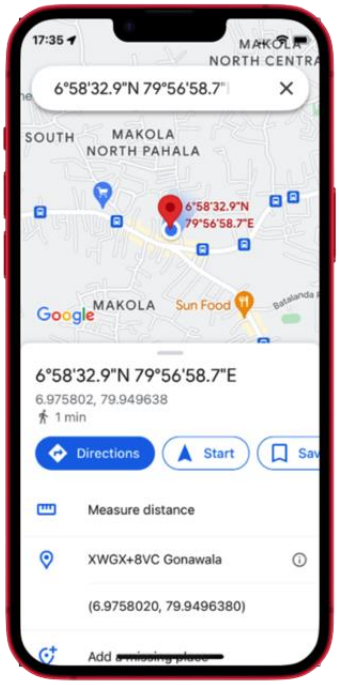
Collision Alert | Rollover Alert | Collision & Rollover Alert

Figure 3.2.1.1: Alert Messages [Emergency Rescue Unit]



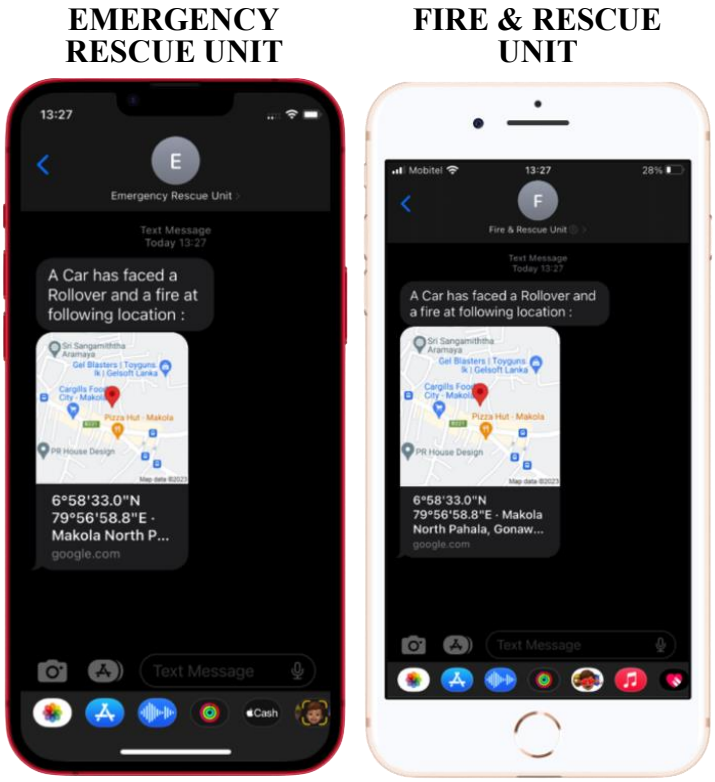
Collision & Fire Alert

Figure 3.2.1.2: Alert Messages [ERU & FRU-1]



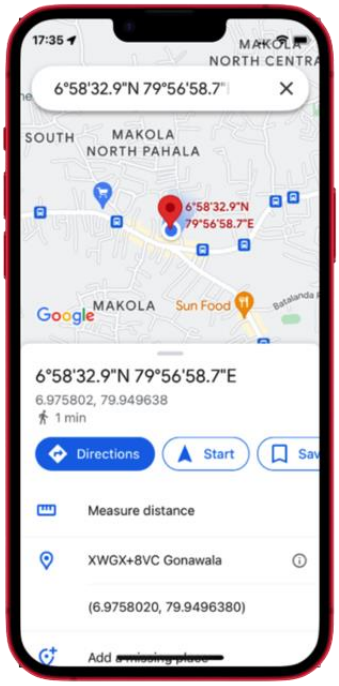
Google Maps

waze



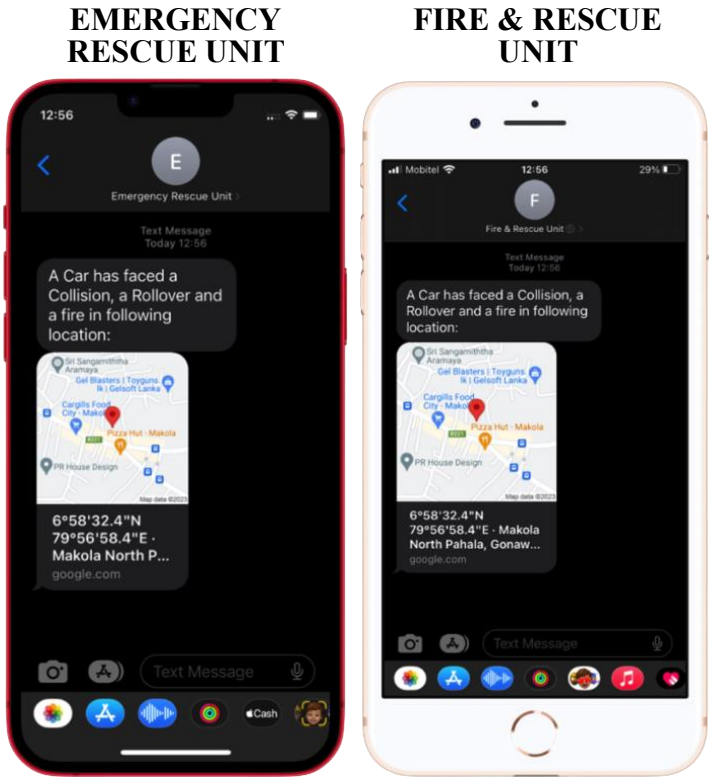
Rollover & Fire Alert

Figure 3.2.1.3: Alert Messages [ERU & FRU-2]



Google Maps

waze



Collision, Rollover & Fire Alert

Figure 3.2.1.4: Alert Messages [ERU & FRU-3]

3.2.2 DRIVER CARELESSNESS ALERT

Another use for this system is to evaluate whether or not the driver was at fault for the accident by looking at the order in which the sensors began to function after the accident. This is clearer by figure 3.2.2.1 below. This means that if the IMU Module is turned on and the car senses a skid or overturn on the road, the system will identify the turn as driver carelessness accident. Along with the message mentioned above, this gets sent.

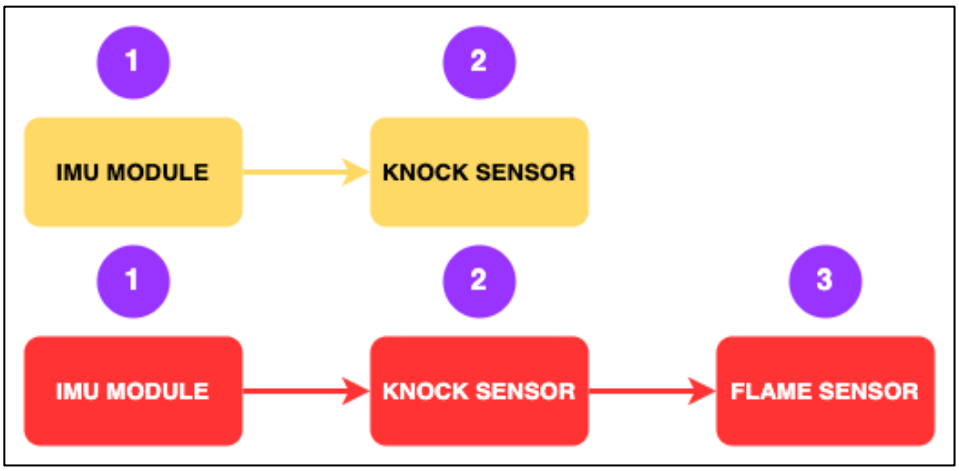


Figure 3.2.2.1: Sensor Order

The following diagrams show the driver carelessness message described above.

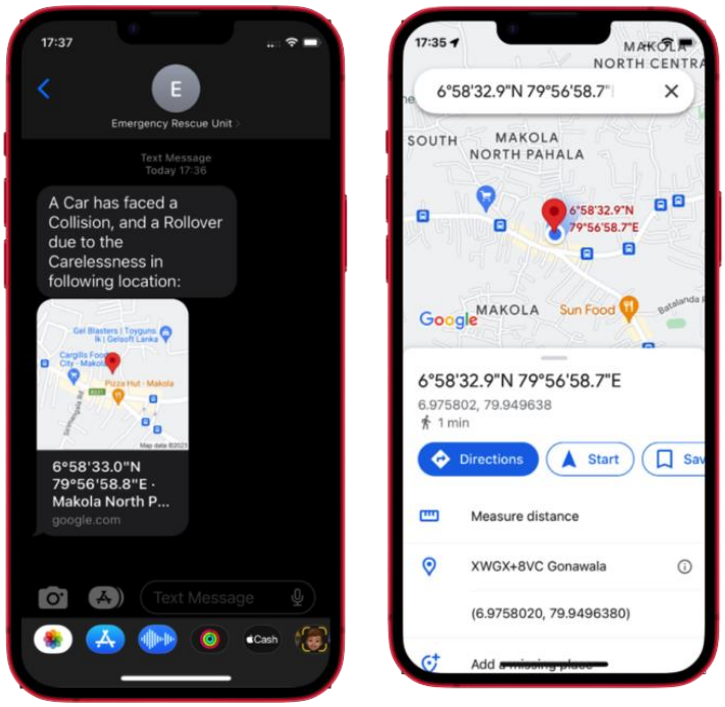


Figure 3.2.2.2: Driver Carelessness Alert [1]

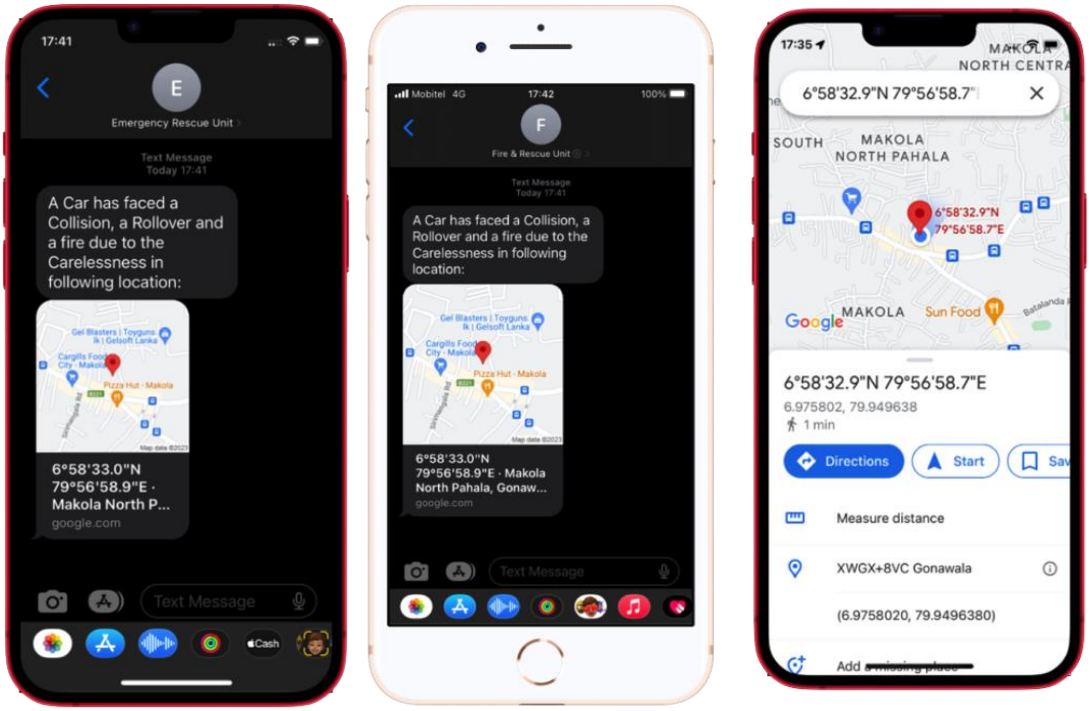


Figure 3.2.2.3: Driver Carelessness Alert [2]

3.3 EXPERIMENTAL RESULTS

The accident detection and alert system has been tested in fifteen different locations around Sri Lanka. This approach was evaluated in sloped, uneven, and curved roadways at various angles. The system recognized an accident if the degree of angles in the X or Y axis exceeded 310 and 340 degrees, respectively. The algorithm identified the danger of a crash if any vehicle collided with the model automobile (Robot Car Chassis Platform). Table 3 summarized the outcomes of the accident detection, exact position tracking, and notification message sending tests.

Evaluated system performance using the experimental data presented in Table 3. The system performance is separated into three sections: (1) Accident detection accuracy (nature of accident, driver carelessness), (2) Location tracking accuracy, (3) Notification sending accuracy.

1) Accident detection accuracy: The accident detection accuracy, or D_A, is defined as (1)

$$D_A = \frac{TP_A}{TP_A + FN_A} \times 100 - - - -[1]$$

Where TP_A and FN_A imply correct and inaccurate accident detection, respectively.

2) Location tracking accuracy: The accuracy of location tracking, or T_L, is defined as (2)

$$T_L = \frac{TP_L}{TP_L + FN_L} \times 100 - - - -[2]$$

TP_L and FN_L denote correct location tracking and incorrect location tracking, respectively.

3) Notification sending accuracy: The notification sending accuracy, or S_N, is defined as (3)

$$S_N = \frac{TP_N}{TP_N + FN_N} \times 100 - - - -[3]$$

where TP_N and FN_N denote successful and unsuccessful message transmission to the specified phone numbers, respectively.

This designed accident detection and notification system provides 100% accident detection accuracy, 87% specific location tracking accuracy, and 93% successful notification sending accuracy across emergency rescue service centers and fire & rescue units.

Number	Accident Detection Status	Location Tracking Status	Notification Sending Status
1	Yes	No	No
2	Yes	No	Yes
3	Yes	Yes	Yes
4	Yes	Yes	Yes
5	Yes	Yes	Yes
6	Yes	Yes	Yes
7	Yes	Yes	Yes
8	Yes	Yes	Yes
9	Yes	No	Yes
10	Yes	Yes	Yes
11	Yes	Yes	Yes
12	Yes	Yes	Yes
13	Yes	Yes	Yes
14	Yes	Yes	Yes
15	Yes	Yes	Yes

Table 3: Experimental Results

4. DISCUSSION

The project has received the required output, which is that soon after the accident, the Knock Sensor, IMU Module, and Flame Sensor will detect the accident and alert the microcontroller, which is placed in Arduino Nano, and if the victim needs all the services, he/she should not press the cancels button; otherwise, if the victim does not need any particular service, he/she can cancel the services by pressing the respective cancel switch within 05s to 15s. Depending on the victim's reaction, a message with a Google Maps location link will be sent to the specific service within 5 to 15 seconds. It is clear from the aforementioned findings that the proposed model has operated as anticipated and has produced positive outcomes.

In contrast to some other existing proposals, this advanced system extends a new, novel blend for Accident Detection (AD), Smart Reporting System (SRS), and driver carelessness alert that has distinctive secure systems and will work for the most exceptional types of situations to prevent car accidents, as shown in Figure 4.1.

PARAMETERS / SYSTEM	GPS	GSM	EMERGENCY RESCUE UNIT ALERT	FIRE RESCUE UNIT ALERT	ACCIDENT NATURE ALERT	SOS	SMS	LOCATION	ALARM	DRIVER CARELESSNESS ALERT	CANCEL SWITCH
Intelligent Accident-Detection And Ambulance-Rescue System [2]	✓	✗	✓	✗	✗	✗	✓	✗	✗	✗	✗
IoT approach to vehicle accident detection, reporting, and navigation [3]	✓	✓	✓	✗	✗	✗	✓	✓	✗	✗	✗
IMU based Accident Detection and Intimation System [4]	✓	✓	✓	✗	✗	✗	✓	✓	✗	✗	✗
An automated system for Accident Detection [5]	✗	✗	✓	✗	✗	✗	✗	✓	✗	✗	✗
SMART VEHICLE COLLISION DETECTION AND SOS SERVICE [6]	✓	✗	✓	✗	✗	✓	✗	✓	✗	✗	✗
Accident detection and reporting system using GPS, GPRS and GSM technology [9]	✓	✓	✓	✗	✗	✗	✓	✓	✓	✗	✗
Real-time crash detection analysis and emergency alert using smartphone [10]	✓	✓	✓	✗	✗	✗	✓	✓	✗	✗	✗
Intelligent accident detection and alert system for emergency medical assistance [11]	✗	✗	✓	✗	✗	✗	✓	✓	✗	✗	✗
Proposed system	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓

Table 4: Performance Metrics

5. CONCLUSION, LIMITATIONS & FUTURE SCOPE

5.1 CONCLUSION

In considering the goals of this manuscript, the designed system has provided a suitable solution to eliminate all potential issues, such as reporting about accidents, finding the nature of the accident (knock, collision, flame) and their locations, providing victims with immediate medical care, and having the option to cancel non-essential services for victims when those services are not required. Even though there are other ways to detect accidents in addition to reporting systems, there is still a significant issue, which is that even if the victim is only experiencing minor problems and does not require services like emergency rescue service center, fire & rescue unit, etc., he is still receiving them, which results in rising tension and time and fuel waste. To fix these problems This proposal appears to be the best option because it offers options of cancel switch for the failure of a wired system, helps to detect the accident through Global Positioning System (GPS), and sends a message along with driver carelessness alert to authorities to investigate the accident by Global System for Mobile module (GSM). The system has a buzzer to warn outsiders and the victim it is following. As a result, this system serves as a good idea for society by saving human lives.

The fact that this proposal addresses the vehicle Accident Detection (AD), Smart Reporting System (SRS), and driver carelessness alert sets it apart from other proposals, as well as the fact that it gives the victim control over canceling non - essential services with the aid of a switch and can resolve all automated system issues.

5.2 LIMITATIONS

Like every coin has two sides, technology offers advantages and disadvantages. This proposal has the following limitations:

5.2.1 SENSOR LIMITATIONS:

a) MPU-6050 6DOF [Appendix 1] –

- The gyroscope and accelerometer can only be sampled at 1-8kHz.

- The SD module's constraints will limit the sample rate to 500 Hz, which is sufficient for many applications.
- The gyroscope drifts, and the values reported are frequently incorrect. The accelerometer, on the other hand, produces a genuine result when the acceleration is progressive, but it suffers greatly from vibrations, returning incorrect angle values.

b) Vibration Sensor [Appendix 1] –

- You must manually adjust the detection level and only have a binary check to see if it is detected.
- Depending on the amount of motion you wish to detect, you'll need a fast loop to check whether motion is observed. This loop will need to be rapid in order to detect tiny durations of motion.

c) Flame Sensor [Appendix 1] –

- It supports a limited range and hence performs worse over longer distances.

5.2.2 OTHER COMPONENTS LIMITATIONS:

a) SIM800C [Appendix 1] –

- The disadvantage of GSM is that multiple users share the same bandwidth. When there are enough consumers, the transmission may suffer interference.
- GSM modules cannot convey large amounts of data generated by sensors; however, they are appropriate for applications requiring a minimal amount of data to be communicated.

b) GPS Module (NEO-6M) [Appendix 1] –

- It can take up to 2-3 hours to connect to the signal.
- Power loss.

5.2.3 OVERALL SYSTEM LIMITATIONS:

- A network at the accident site is critical to the project's success.

- In rare cases, system elements may be harmed only if the system's hardcore protective box is damaged.
- Because the accident is recognized using only the IMU module and knock sensors, the system has a poor level of accuracy.
- The Arduino Nano runs on a 5V battery. Even if the battery is depleted to 4.9V, it will malfunction and produce incorrect results.

5.3 FUTURE SCOPE

Further research can be conducted in the future to improve the algorithm and hardware components for this system and completely operationalize the system for accident avoidance. The following are a few suggestions for potential improvement areas.

- Vehicles can be equipped with rotating camera modules that can take pictures of their surroundings and send them to the appropriate services for further location clarification.

- In the future, we can integrate this system with a variety of sensors, including alcohol, sleepiness, heart rate, and other detectors.

- The suggested method can be improved by using AI and ML to identify accidents and assess the environment at the accident site and storing driving activities determine the accident is occurred by driver carelessness.

- Self-driving cars might use the proposed technology as well. This proposed solution can be used as a backup to protect human lives if a self-driving car is unable to prevent an accident.

6. REFERENCES

[01] Road safety facts (2022) Association for Safe International Road Travel. Available at: https://www.asirt.org/safe-travel/road-safety-facts/ (Accessed: January 10, 2023).

[02] B. Prachi, D. Kasturi, and C. Priyanka, "Intelligent Accident-Detection and Ambulance-Rescue System." [Online]. Available: https://core.ac.uk/download/pdf/24067793.pdf

[03] "An IOT approach to vehicle accident detection, reporting, and Navigation." [Online]. Available:https://www.researchgate.net/publication/311531453_An_IoT_approach_to_vehicle_accident_detection_reporting_and_navigation. (Accessed: 10-Jan-2023).

[4] P. Nath and A. Malepati, "IMU based Accident Detection and Intimation System," IEEE Xplore, May 01, 2018. https://ieeexplore.ieee.org/document/8465309 (accessed Jan. 10, 2023).

[05] A. Ali and M. Eid, "An automated system for Accident Detection," nyuscholars.nyu.edu, Jul. 06, 2015. https://nyuscholars.nyu.edu/en/publications/an-automated-system-for-accident-detection (accessed Jan. 10, 2023).

[06] M. S. Amin, M. A. S. Bhuiyan, M. B. I. Reaz, and S. S. Nasir, "GPS and Map matching based vehicle accident detection system," 2013 IEEE Student Conference on Research and Development, 2013 (Accessed May 05, 2022).

[07] A. Ali and M. Eid, "An automated system for accident detection," Researchgate.net. [Online]. Available: https://www.researchgate.net/publication/340400532_An_Approach_Towards_Intelligent_Accident_Detection_Location_Tracking_and_Notification_System. (Accessed: 10-May-2022).

[08] Elie Nasr, Elie Kfoury, David Khoury, "An IoT approach to vehicle accident detection, reporting and navigation," Researchgate.net. [Online]. Available:

https://www.researchgate.net/publication/311531453_An_IoT_approach_to_vehicle_accident_detection_reporting_and_navigation. (Accessed: 10-May-2022).

[09] Md. Syedul Amin, J. Jalil, and M. B. I. Reaz, "Accident detection and reporting system using GPS, GPRS and GSM technology," 2012 International Conference on Informatics, Electronics & Vision (ICIEV), May 2012, doi: 10.1109/iciev.2012.6317382.

[10] H. Sharma, R. K. Reddy, and A. Karthik, "S-CarCrash: Real-time crash detection analysis and emergency alert using smartphone," IEEE Xplore, Sep. 01, 2016. https://ieeexplore.ieee.org/document/7800181 (accessed Jan. 10, 2023).

[11] N. Kattukkaran, A. George, and T. P. M. Haridas, "Intelligent accident detection and alert system for emergency medical assistance," in 2017 International Conference on Computer Communication and Informatics (ICCCI), 2017, pp. 1–6 (Accessed: 10-May-2022).

[12] Adamu Umar, Shengbo Hu, Hongqiu Luo, "In-Vehicle Stereo Vision Systems with Improved Ant Colony Optimization Based Lane Detection: A Solution to Accidents Involving Large Goods Vehicles Due to Blind Spots," Scirp.org. [Online]. Available: https://www.scirp.org/journal/paperinformation.aspx?paperid=116031. (Accessed: 10-May- 2022).

[13] V. G. ALEJANDRA CECILIA, "DESIGN AND IMPLEMENTATION OF AN INERTIAL NAVIGATION SYSTEM USING KALMAN FILTER," 2017, Accessed: Jan. 15, 2023. [Online]. Available: https://repositorio.usm.cl/handle/11673/41282?show=full

[14] Naveen Prabu Palanisamy, "FILTERING OF IMU DATA USING KALMAN FILTER," Calstate.edu. [Online]. Available: https://scholarworks.calstate.edu/downloads/dv13zt241. (Accessed: 16-Jan-2023).

[15] G. Ferrer Mínguez, "Integración Kalman de sensores inerciales INS con GPS en un UAV," upcommons.upc.edu, Apr. 2009, Accessed: Jan. 18, 2023. [Online]. Available: https://upcommons.upc.edu/handle/2099.1/6930?locale-attribute=en

7. APPENDICES

7.1 Appendix 1: Description of Components

7.1.1 Arduino Nano

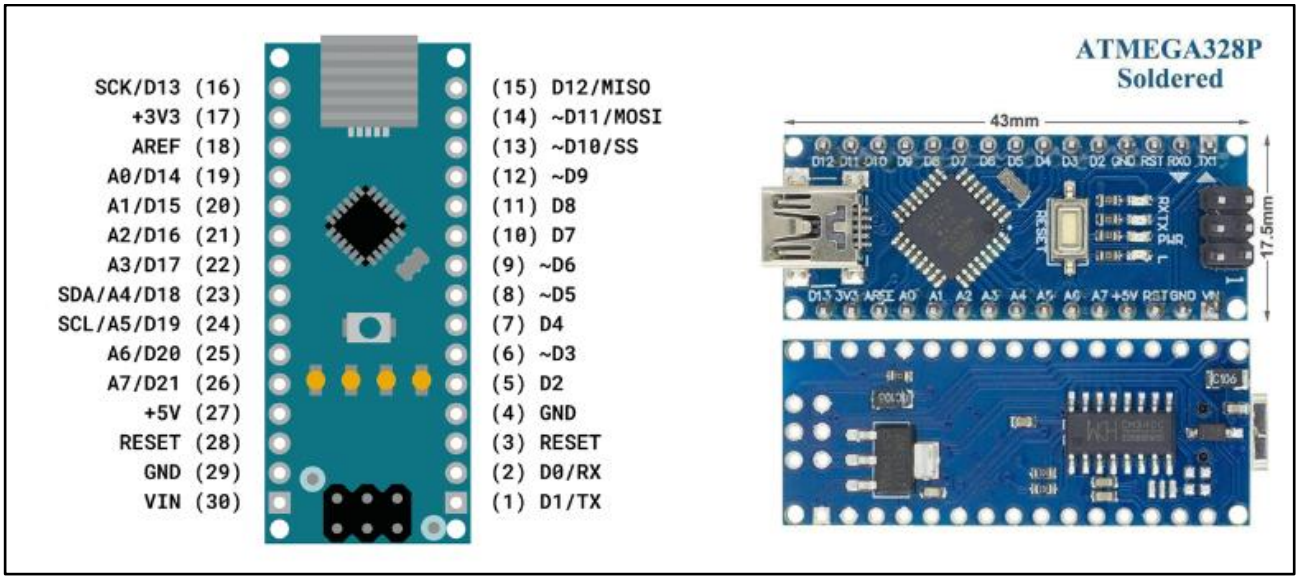


Figure 7.1.1.1: Arduino Nano

The ATmega 328p is an AVR microprocessor. It uses extremely little power and has an 8-bit CMOS architecture. The accident detection system was prototyped using an Arduino Nano, an ATmega 328p based development board. The I2C communication protocol, required for interacting with the IMU, is supported, and a serial connection is provided for receiving GPS and SIM800C data, making it the perfect choice for this application.

The microprocessor receives or transmits data from either the LCD screen, acceleration, and orientation using I2C. The microcontroller's digital I/O pins that are available are used for the sensor interface. The software serial port is used to connect the GPS and GSM. The accident detection algorithm is then used to all the data, and if an accident is found, the emergency services are notified right once of the location and specifics of the affected car.

7.1.2 SIM 800C

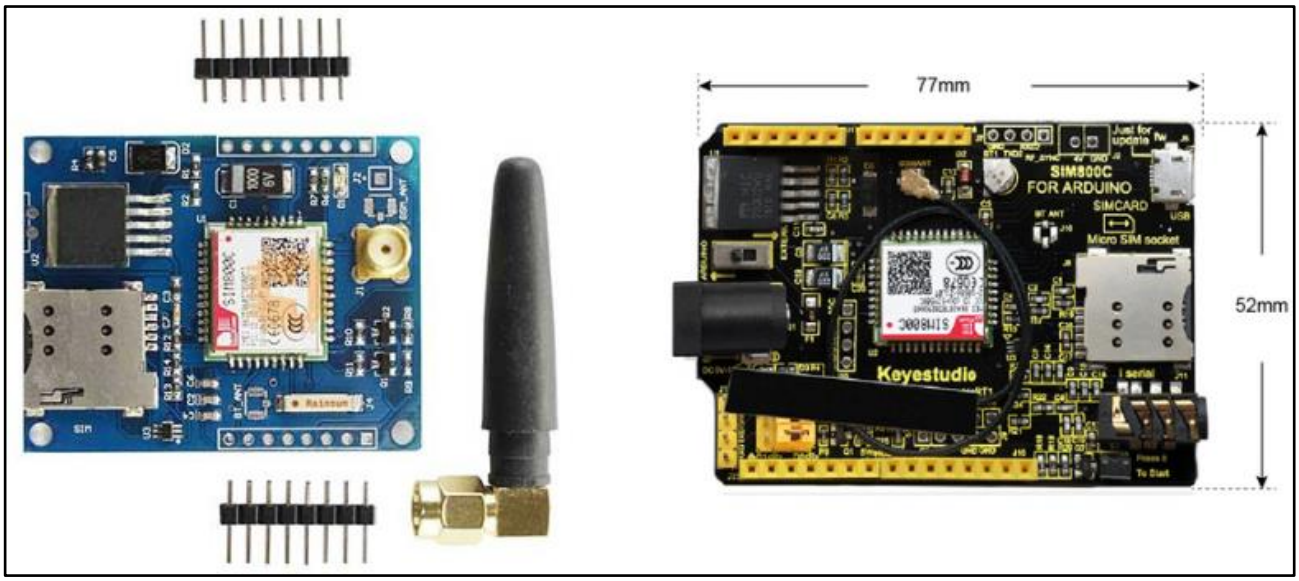


Figure 7.1.2.1: SIM 800C

The system needs a GPS position that will be used to alert emergency services (EMS) in the event of an accident. The GPS will also provide the vehicle velocity, which can be utilized to determine the potential severity of an accident. In the event of an accident, the EMS will be notified by GSM/GPRS.

In here this system use the SIM 800C Quad-Band GSM/GPRS module because it also has a GPS receiver for satellite navigation. Specific AT commands can be used to control this using the Arduino board over serial communication. The frequencies of EGSM 900MHz/DCS 1800MHz and GSM 850MHz/PCS 1900MHz are supported by the Quad-band GSM/GPRS engine. A 42 channel GPS receiver is also included with this shield. It provides decent navigation performance in urban environments at 1Hz update rate with 2.5m CEP and 160dBm tracking sensitivity.

7.1.3 GPS Module (NEO-6M)

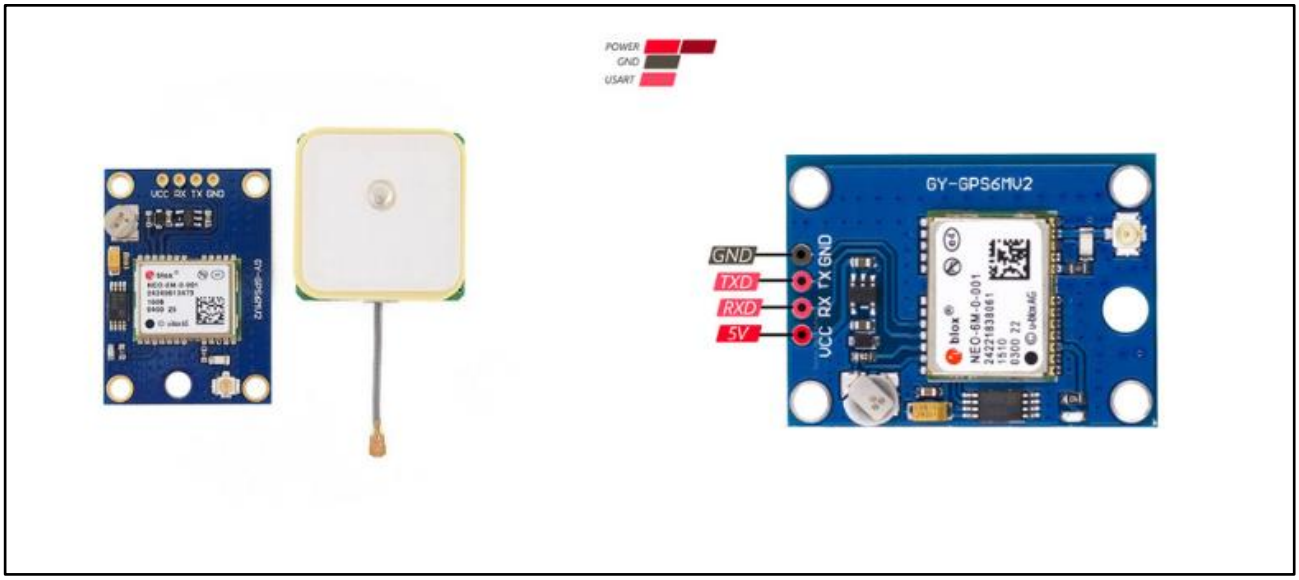


Figure 7.1.3.1: NEO – 6M GPS Module

The high-performance u-box 6 positioning engine is a feature of the NEO-6 module series, a family of standalone GPS receivers. In a tiny 16 x 12.2 x 2.4 mm dimension, these adaptable and reasonably priced receivers provide a wide range of connectivity choices. NEO-6 modules are the best choice for battery-operated portable devices with extremely tight budget and space restrictions thanks to their small architecture, power, and memory options. With a Time To-First Fix0 (TTFF) of less than 1 second, the 50-channel u-box 6 positioning engine is impressive. The dedicated acquisition engine can conduct enormous simultaneous time/frequency space searches with 2 million correlates, which makes it possible to find satellites right away.

Certain libraries must be imported to use a GPS Module with an Arduino Nano. They are a tiny GPS library and a software serial library. By simulating the capabilities of the hardwired RX and TX lines with software, the software serial library enables serial communication on the other digital pins of circuit boards. The tiny GPS library transforms global positioning information in NEMA format into convenient variables like Latitude, Longitude, Time, and other variables.

7.1.4 Buck Converter (LM2596)

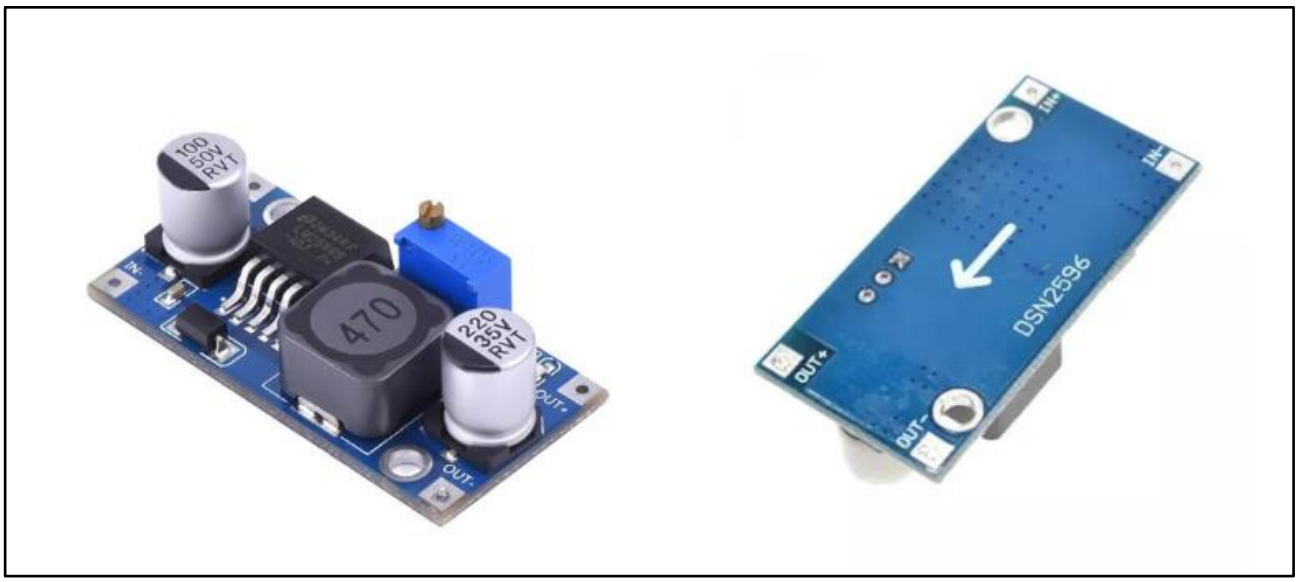


Figure 7.1.4.1: LM2596 Step Down Converter

The step-down (buck) switching regulators in the LM2596 series of regulators are monolithic integrated circuits that can drive a 3-A load with excellent line and load regulation. These devices come with 3.3 V, 5 V, 12 V fixed output voltages as well as an adjustable output version. These regulators are easy to operate, need a minimal number of external components, and have an internal frequency compensator and fixed-frequency oscillator.

Since all the system's devices require 5V, this is set to 5V. And the system is powered by a 7V power bank, which is converted to 5V through the buck converter.

Features:

- Wide input voltage ranges up to 40 V.
- Adjustable output voltage range 1.23 V − 37 V.
- Fixed-frequency internal oscillator @ 150 kHz.
- 150 kHz adjustable output voltage range.
- Guaranteed 3.0 A output load current.
- 80 A low power standby mode.

7.1.5 IMU Module (MPU6050)

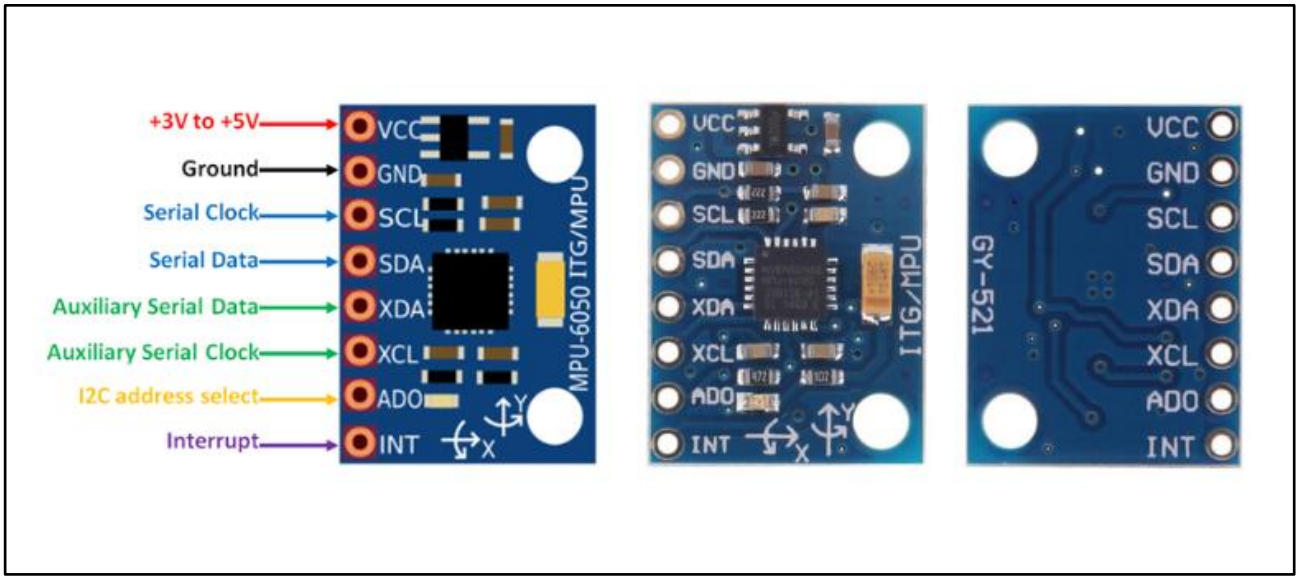


Figure 7.1.5.1: MPU6050 6-DoF Accelerometer and Gyro

A six-axis IMU sensor is the MPU 6050. It outputs six values: three from the gyroscope and three each from the accelerometer. Micro electromechanical systems technology is the foundation of the sensor MPU 6050. Self-balancing robots, unmanned drones, smartphones, etc. all use MPU 6050. IMU sensors determine the position of a sensor-attached item in three dimensions. Typically, these values are expressed as angles to aid with positioning.

It is a low-cost, low-power speed tracking device that can be installed in any device to monitor an object's movement speed. Gyros and accelerometers can both produce 2000°/sec second and 16g acceleration, respectively, which is more than enough to detect car accidents. The on-board AT mega 328p microprocessor handles the output processing.

7.1.6 Knock/Vibration Sensor

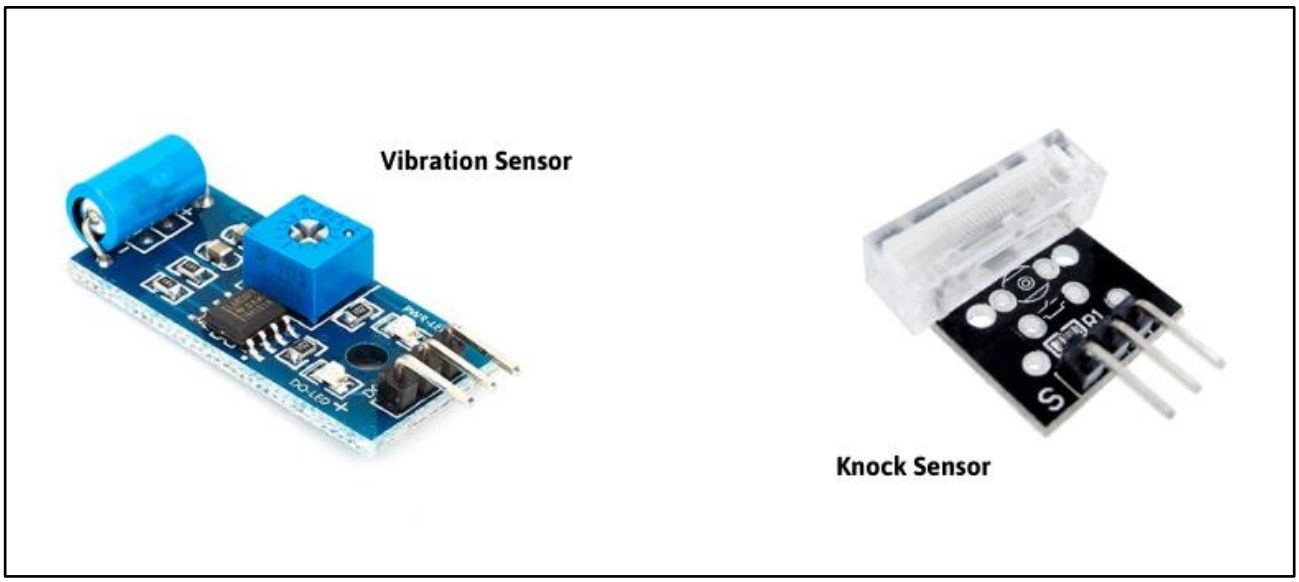


Figure 7.1.6.1: Knock/Vibration Sensor

This system works with both types of sensors and their functions are very similar. But in the hardware design of this system, the vibration sensor is used (due to the lack of goods, the knock sensor could not be used), so it is discussed here.

A vibration sensor is a tool that measures the strength and frequency of vibration in a certain machine, system, or piece of equipment. These metrics can be utilized to find an unexpected knock. The output from the sensor following impact is +5v and attached to the Arduino Nano's INT (pin 16). These sensors are fitted to the car's sides to identify whether an impact has happened there. The simplest possible connection between these sensors allows them to detect force impacts coming from any side of the car. For the paramedics to get at the scene as quickly as possible after an accident, this is frequently concerned with the protection of the system of the person operating the vehicle.

7.1.7 Flame Sensor

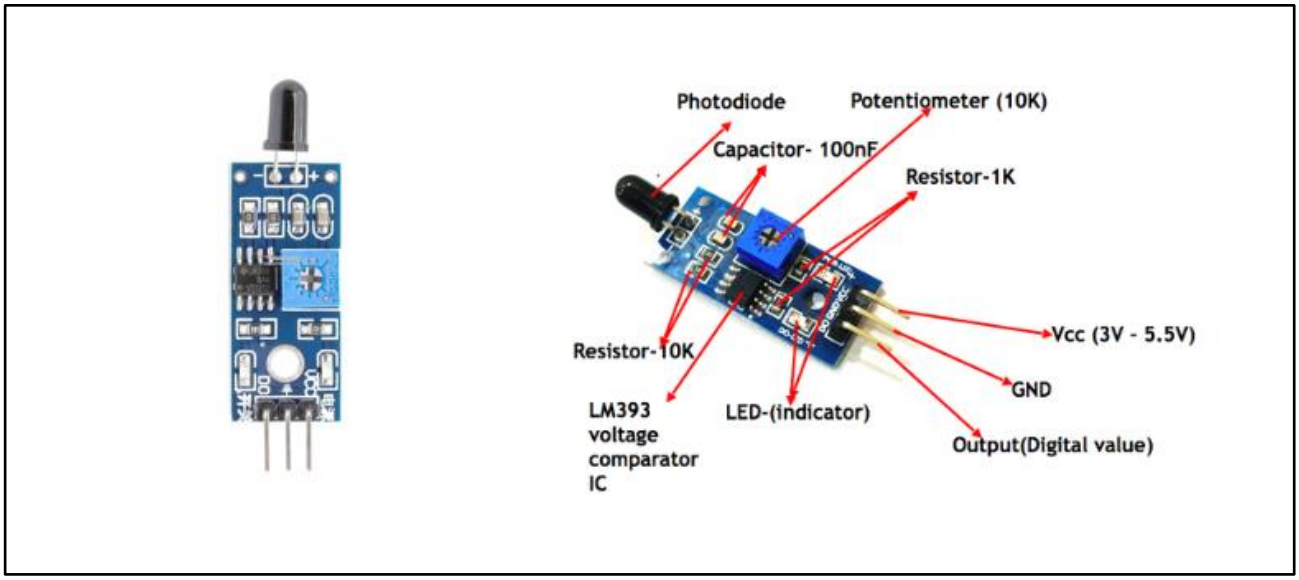


Figure 7.1.7.1: Flame Sensor

In this approach, sensors with characteristics like the Infrared Flame sensor, which has a high photosensitivity, a wide directional angle, and a quick response time, are utilized if the car catches fire after the accident. The detection range of this sensor is 20 cm (4.8 V) 100 cm, and it can detect flame sources with wavelengths between 760 and 1100 nm (1V). It can detect flame spectrum with a detection angle of roughly 60 degrees. The LM393 comparator chip ensures consistency in module readings. A signal was sent to the GSM module for notification once the module sensed readings for flame detection.

7.1.8 LCD Display

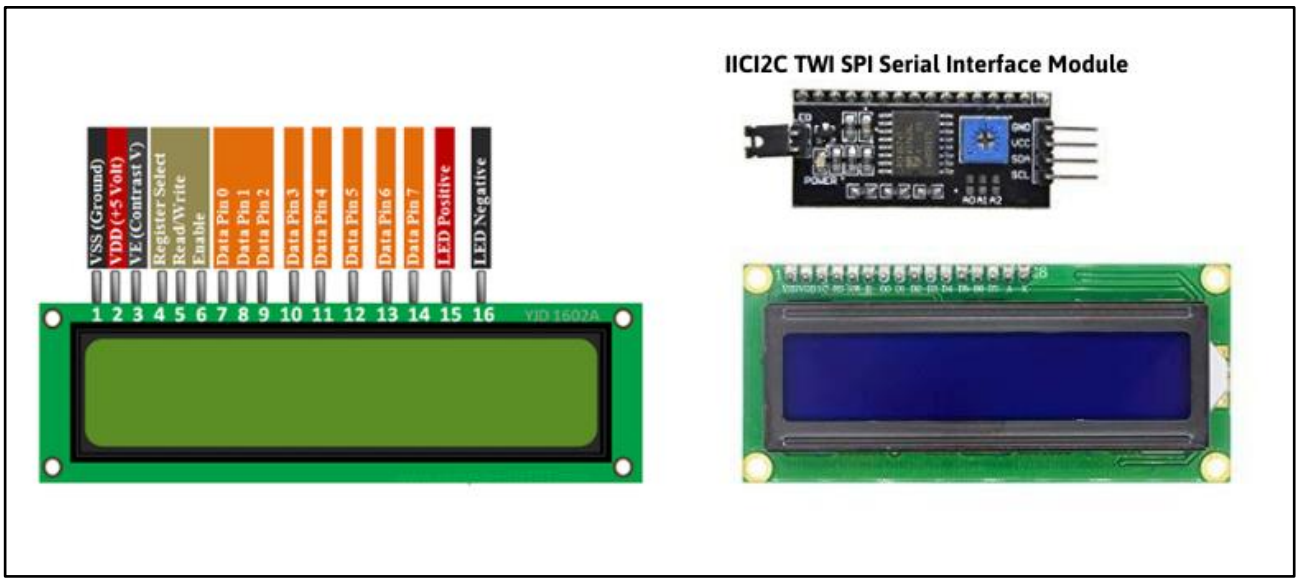


Figure 7.1.8.1: LCD 1602 Display

The 16x2 LCD display is used to show how the circuit is operating. In total, there are 16 pins. From left to right, they are numbered 1 to 16 (if you are reading from the backside). It's not particularly difficult to create custom LCD characters. It necessitates an understanding of LCD's custom generated random-access memory (CG-RAM) and the LCD chip controller. Hitachi HD4478 controller is typically found in LCDs. The primary tool for creating unique characters is CG-RAM. Once specified in the code, it stores the custom characters. The 64-byte CG-RAM capacity gives users the possibility to create eight characters at once. Eight bytes make up each character. In table 1, all the functionalities displayed on the LCD Screen are listed. The circuit diagram provides connections for LCD pins (Figure 2.5.1).

Status	**Message**
After system power on [1].	Accident Alert System
When the sensors are triggered [2].	Delay Time [15s to 0s (time depends on accident nature)]
After the delay time [3].	Accident Detected.!

Table 5: LCD Messages

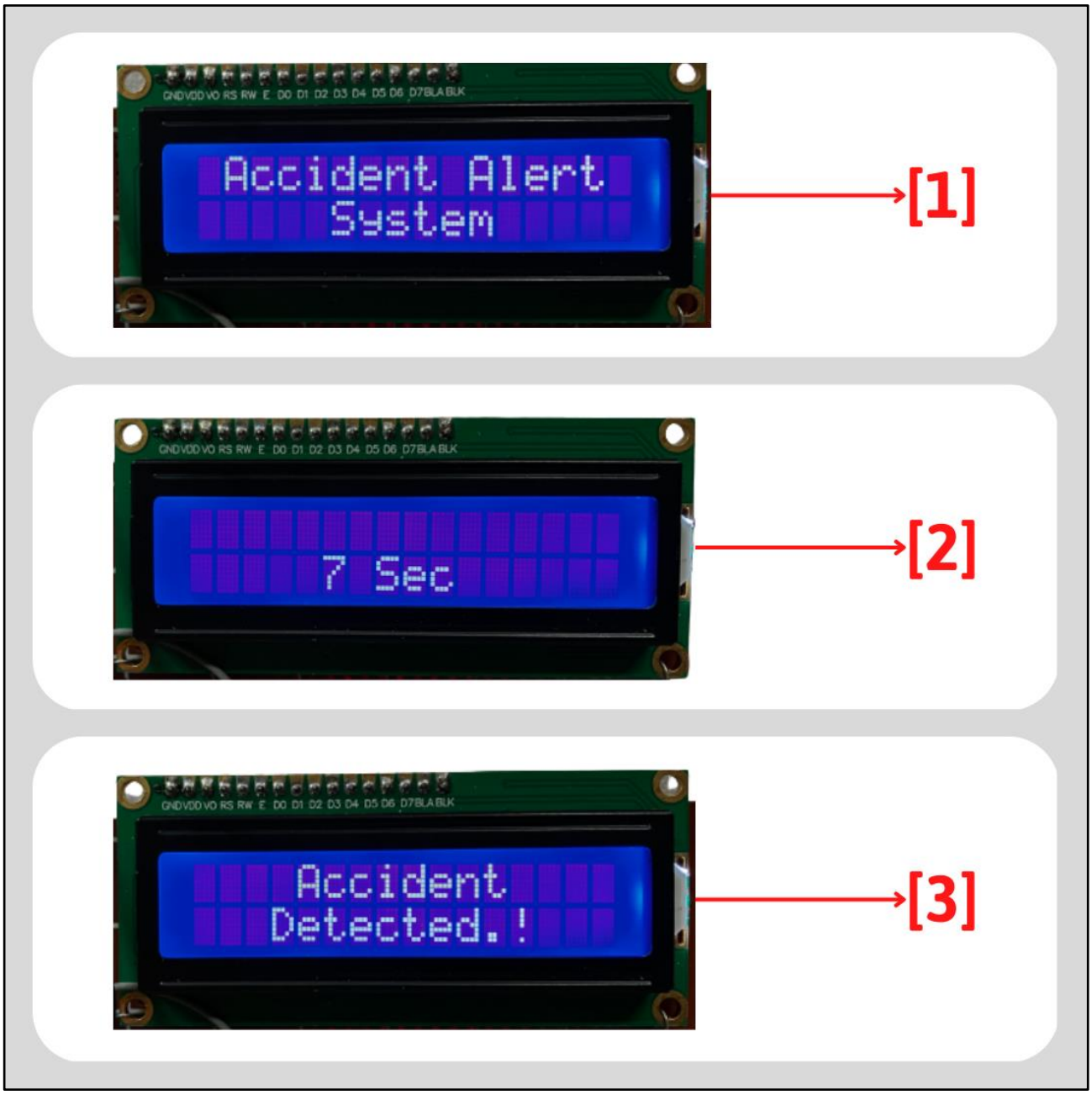


Figure 7.1.8.2: LCD Screen Outputs

7.2 Appendix 2: Summary of Sensors Pin Connection

Arduino Nano	MPU-6050 6DOF
5V	VCC
GND	GND
A4	SDA
A5	SCL

Table 6: IMU Module

Arduino Nano	NEO-6M GPS Module
5V	VCC
GND	GND
D2	RX
D3	TX

Table 7: GPS Module

Arduino Nano	SIM 800C
5V	VCC
GND	GND
D5	TXD
A1	RXD

Table 8: SIM800C GSM Module

Arduino Nano	Vibration Sensor
5V	VCC
GND	GND
A2	DO (Digital Output)

Table 9: Vibration Sensor

Arduino Nano	Flame Sensor
5V	VCC
GND	GND
D6	DO (Digital Output)

Table 10: Flame Sensor

Arduino Nano	LCD 1602 & IIC12C TWI SPI Module
5V	VCC
GND	GND
A4	SDA
A5	SCL

Table 11: LCD Display

Arduino Nano	Buzzer
GND	GND
D11	Positive

Table 12: Buzzer

Arduino Nano	Push Button
5V	Positive
D9	Negative

Table 13: Push Button

7.3 Appendix 3: PCB Design and Layout

Here, the PCB was designed according to the circuit diagram explained under chapter number 2.7. It can be seen in the diagrams below. Easy EDA and Fusion 360/EAGLE software were used to design this PCB.

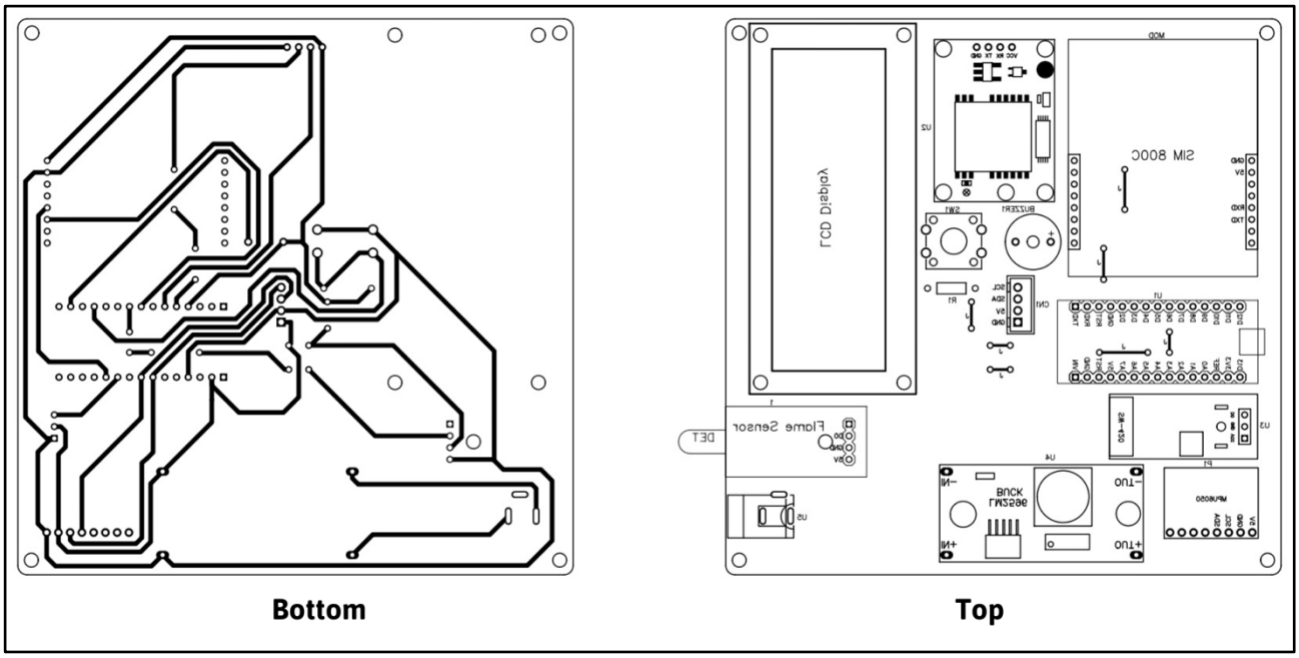


Figure 7.3.1: PCB Design

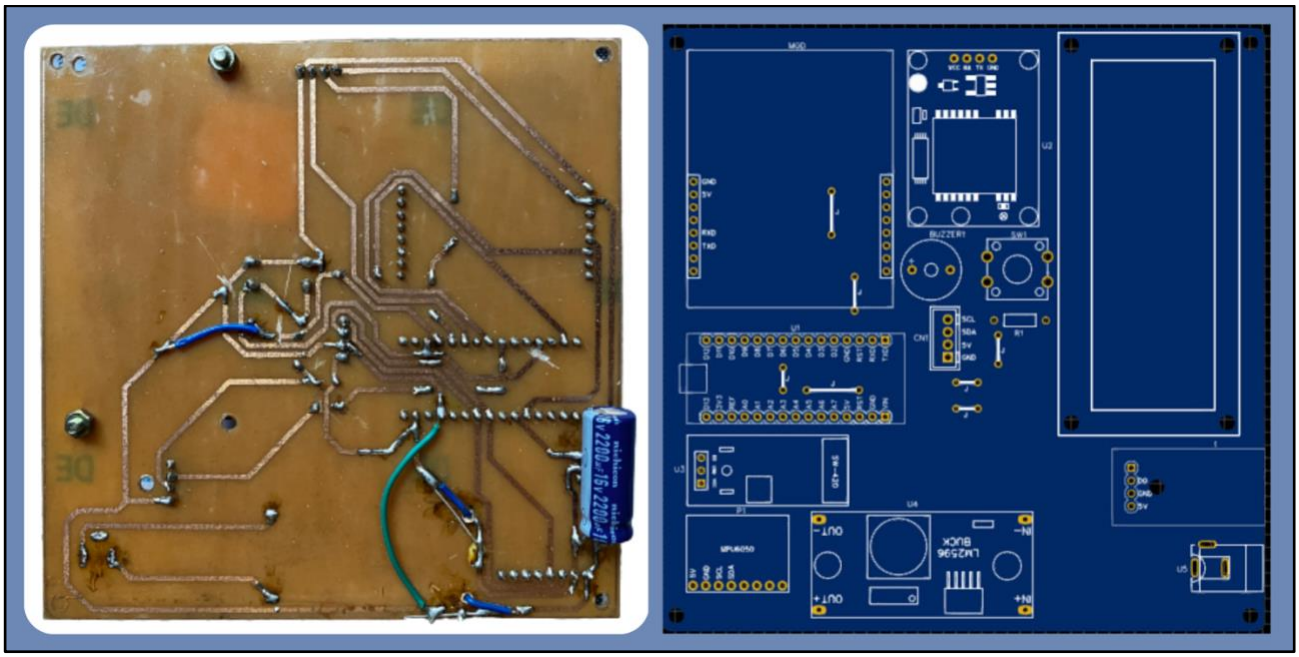


Figure 7.3.2: PCB Soldering

7.4 Appendix 4: Final Product

The diagram below shows the final configuration of the system. It consists of two final configurations as dot board and PCB. After all the devices discussed in the above chapters are assembled, the functional interface of the complete system appears in the following diagrams.

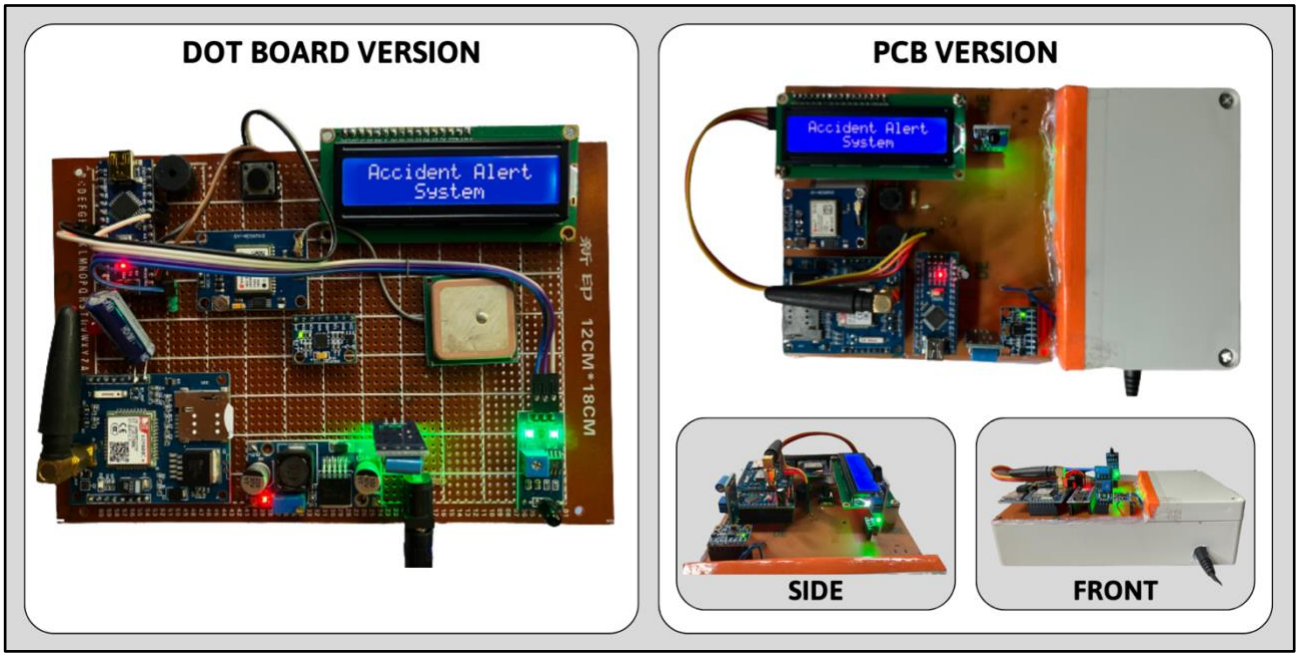


Figure 7.4.1: Final Product (PCB/Dot Board)

7.5 Appendix 5: Project Gantt Chart

In terms of the project schedule, the figure below shows a general idea of the project's main stages, followed by the starting and ending dates.

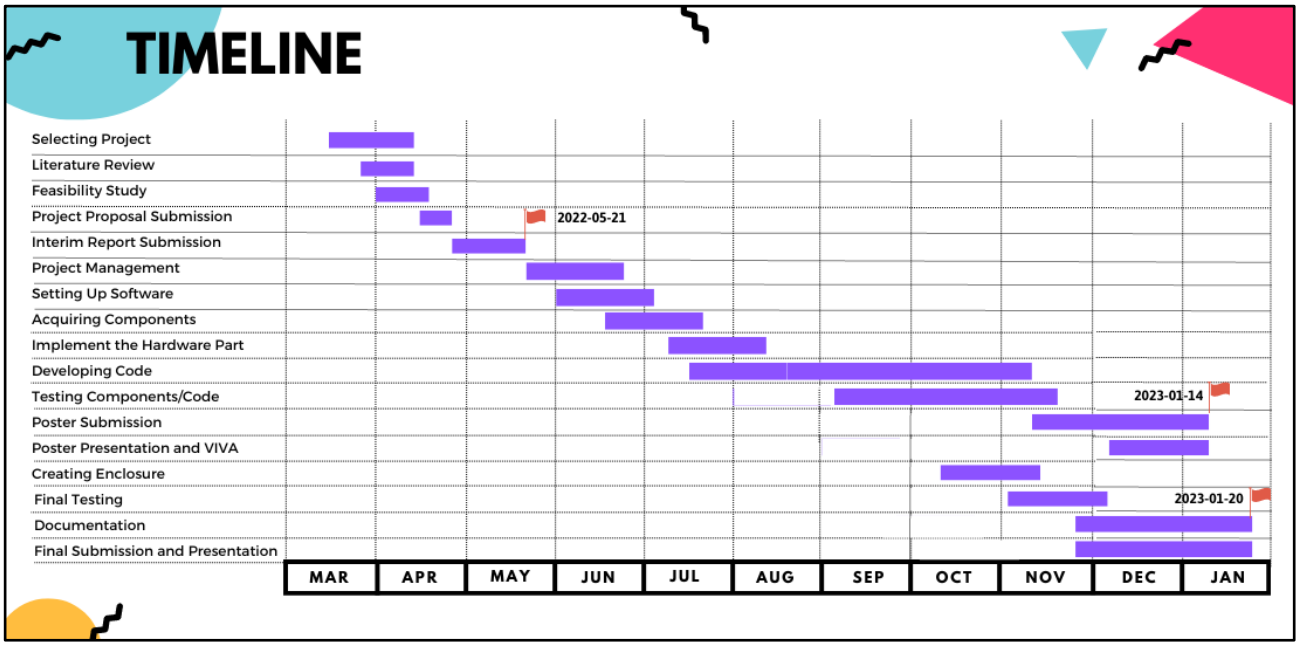


Figure 7.5.1: Project Timeline

7.6 Appendix 5: Bill of Materials

Table 14, 15, 16 below shows all the amount of money spent to create this system. Here it shows how the money is divided into two parts as dot board version and PCB version. These calculations are subject to the prices of the goods as of the date shown in the tables.

DOT BOARD VERSION - [2022-05-23]		
PRODUCT	**QUANTITY**	**PRICE**
Arduino Nano [ATmega328]	1	LKR 2,725
SIM800C GSM GPRS Module	1	LKR 4,160
GPS Module [NEO-6M]	1	LKR 1,790
LCD 16x02 Display	1	LKR 885
IIC12C TWI SPI Interface Module	1	LKR 325
Buzzer Alarm	1	LKR 240
Push Button	1	LKR 6
IR Infrared 3 Wire Flame Sensor	1	LKR 235
MPU-6050 6DOF	1	LKR 490
LM2596 Step Down Converter	1	LKR 345
1 Ohm 1/4W 1R 5% Carbon Film Resistor	10	LKR 15
LED 3mm Water Clear 20mA Red bulb	2	LKR 8
Solder Wire 0.8mm 100g *1 Roll	1	LKR 1,350
Male to Female Jumper Wire 40PCS	1	LKR 260
Male to Male Jumper Wire 40PCS	1	LKR 260
Circuit Board	1	LKR 50
Transport		LKR 450
Total expenses (LKR)		**LKR 13,594**
Total expenses (GBP)		€30.10

GBP 1 = LKR 451.67 (2023-01-18)

Table 14: Expenses for Dot Board Version

PCB VERSION - [2023-01-03]		
PRODUCT	QUANTITY	PRICE
Arduino Nano [ATmega328]	1	LKR 2,900
SIM800C GSM GPRS Module	1	LKR 4,160
GPS Module [NEO-6M]	1	LKR 1,850
LCD 16x02 Display	1	LKR 715
IIC12C TWI SPI Interface Module	1	LKR 460
Buzzer Alarm	1	LKR 240
Push Button	1	LKR 50
IR Infrared 3 Wire Flame Sensor	1	LKR 235
MPU-6050 6DOF	1	LKR 695
LM2596 Step Down Converter	1	LKR 345
3x30mm Bolt with Nut	4	LKR 40
30mm Plastic Spacer	4	LKR 40
Female Pin Header Breakable Strip 40 *1Pcs	5	LKR 175
200 x 120 x 55 Plastic Enclosure Project box	1	LKR 850
PCB Etching	1	LKR 1,270
Total expenses (LKR)		**LKR 14,025**
Total expenses (GBP)		€31.10

GBP 1 = LKR 451.67 (2023-01-18)

Table 15: Expenses for Dot Board Version

FULL TOTAL	
DOT BOARD VERSION	LKR 13,594
PCB VERSION	LKR 14,025
Total expenses	**LKR 27,619**
Total expenses	€61

GBP 1 = LKR 451.67 (2023-01-18)

Table 16: Total Expenses

7.7 Appendix 6: Arduino Code

Below is the working code of the complete system. This compiles with Arduino IDE 2.0.3.

```
//Including Libraries
#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
#include <Wire.h>
#include <SoftwareSerial.h>
#include <TinyGPS++.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 16, 2); // set the LCD address to 0x27 for a 16 chars and 2 line display
TinyGPSPlus gps;      // The TinyGPS++ object
SoftwareSerial ss(3, 4);// The serial connection to the GPS device
double latitude;
double longitude;

SoftwareSerial serialSIM800(5, 15);// The serial connection to the GSM device
//double latitude;

Adafruit_MPU6050 mpu;

//Declaring sensor Pins
byte buzzer = 11;
byte led = 15;
byte  shock_Sensor = 16;
byte  flame_Sensor = 6;

//Declaring status variables
byte  shock_Sensor_State = 0;
byte  flame_Sensor_State = 1;
byte  is_Car_Flipped = 0;
byte  is_Shoked = 0;
byte  is_Flame = 1;
byte is_Special_Scenario = 0;
byte is_Careless = 0;


void setup() {
 // Declaring I/O Devices
 pinMode(shock_Sensor, INPUT);
 pinMode(buzzer, OUTPUT);
```

```
 pinMode(led, OUTPUT);
 digitalWrite(led, LOW);

 serialSIM800.begin(9600); //Start software Serial for GSM
 ss.begin(9600);//Start software Serial for GPS

 while (!Serial);

 // Try to initialize!
 if (!mpu.begin()) {
   while (1) {
     delay(10);
   }
 }

 // set accelerometer range to +-8G
 mpu.setAccelerometerRange(MPU6050_RANGE_8_G);

 // set gyro range to +- 500 deg/s
 mpu.setGyroRange(MPU6050_RANGE_500_DEG);

 // set filter bandwidth to 21 Hz
 mpu.setFilterBandwidth(MPU6050_BAND_21_HZ);
 delay(100);

 lcd.init();// Initialize LCD

 // Print a message to the LCD.
 lcd.backlight();
 lcd.setCursor(1, 0);
 lcd.print("Accident Alert");
 lcd.setCursor(5, 1);
 lcd.print("System");
}

void loop() {

 read_Shock();
 if (is_Shoked == 1) {
   tone(buzzer, 1000); // Send 1KHz sound signal...
   digitalWrite(led, HIGH); // Turn ON LED
   delay(800);
   noTone(buzzer);     // Stop sound...
   is_Careless = 0;
```

```
unsigned long start = millis();
int shock_Increment = 0;
do {

  lcd.clear();
  lcd.setCursor(5, 1);
  int fifteen_Countdown = (15000 - (millis() - start)) / 1000;
  lcd.print(fifteen_Countdown);
  lcd.print(" Sec");
  delay(50);

  shock_Increment++;

  read_Gyro();
  read_Flame();

  if (is_Car_Flipped == 1) {
    byte  a = 0;
    do {
      a++;
      delay(50);

      lcd.clear();
      lcd.setCursor(5, 1);
      lcd.print((5000 - (a * 50)) / 1000);
      lcd.print(" Sec");

      read_Flame();
      if (is_Flame == 1) {
        is_Special_Scenario = 1;
      }

    } while (a < 100);
    // run accident execution

    if (is_Special_Scenario == 1) {
      C_R_F();
      is_Special_Scenario = 0;
    }
    else {
      C_R();
    }
    //C_R();
```

```
    }

    if (is_Flame == 1) {
      byte  b = 0;

      do {
        b++;
        delay(50);

        lcd.clear();
        lcd.setCursor(5, 1);
        lcd.print((10000 - (b * 50)) / 1000);
        lcd.print(" Sec");

        read_Gyro();
        if (is_Car_Flipped == 1) {
          byte  a = 0;
          do {
            a++;
            delay(50);

            lcd.clear();
            lcd.setCursor(5, 1);
            lcd.print((5000 - (a * 50)) / 1000);
            lcd.print(" Sec");
          } while (a < 100);
          // run accident execution
          C_R_F();
        }

      } while (b < 200);
      // run * accident execution
      C_F();
    }



  } while (millis() - start < 15000);
  //run possible accident execution
  C();
}
```

```
read_Gyro();
if (is_Car_Flipped == 1) {
 tone(buzzer, 1000); // Send 1KHz sound signal...
 digitalWrite(led, HIGH);
 delay(800);
 noTone(buzzer);    // Stop sound...
 is_Careless = 1;
 int flip_Increment = 0;
 unsigned long start = millis();
 do {

  lcd.clear();
  lcd.setCursor(5, 1);
  int fifteen_Countdown = (15000 - (millis() - start)) / 1000;
  lcd.print(fifteen_Countdown);
  lcd.print(" Sec");
  delay(50);//50

  flip_Increment++;

  read_Shock();
  read_Flame();

  if (is_Shoked == 1) {
   byte  a = 0;
   do {
    a++;
    delay(50);

    lcd.clear();
    lcd.setCursor(5, 1);
    lcd.print((5000 - (a * 50)) / 1000);
    lcd.print(" Sec");
    //  Serial.println("Inside 5 sec count down");

    read_Flame();
    if (is_Flame == 1) {
     is_Special_Scenario = 1;
    }

   } while (a < 100);
   if (is_Special_Scenario == 1) {
    C_R_F();
    is_Special_Scenario = 0;
```

```
    }
    else {
      C_R();
    }

  }


  if (is_Flame == 1) {
    byte  b = 0;

    do {
      b++;
      delay(50);

      lcd.clear();
      lcd.setCursor(5, 1);
      lcd.print((10000 - (b * 50)) / 1000);
      lcd.print(" Sec");


      read_Shock();
      if (is_Shoked == 1) {
        byte  a = 0;
        do {
          a++;
          delay(50);

          lcd.clear();
          lcd.setCursor(5, 1);
          lcd.print((5000 - (a * 50)) / 1000);
          lcd.print(" Sec");

        } while (a < 100);
        C_R_F();
      }

    } while (b < 200);
    R_F();
    // run * accident execution
  }

} while (millis() - start < 15000);
//run possible accident execution
```

```
   R();

  }


  delay(100);
}

void read_Shock() {
  shock_Sensor_State = digitalRead(shock_Sensor);
  if (shock_Sensor_State == 1) {
    is_Shoked = 1;
  }
  else {
    is_Shoked = 0;
  }
}

void read_Flame() {
  flame_Sensor_State = digitalRead(flame_Sensor);
  if (flame_Sensor_State == 0) {
    is_Flame = 1;
  }
  else {
    is_Flame = 0;
  }
}

void read_Gyro() {
  /* Get new sensor events with the readings */
  sensors_event_t a, g, temp;
  mpu.getEvent(&a, &g, &temp);

  if ((a.acceleration.y) < -3) {
    //  Serial.println("Right Flip");
    is_Car_Flipped = 1;
  }
  else if ((a.acceleration.y) > 3) {
    //  Serial.println("Left Flip");
    is_Car_Flipped = 1;
  }
  else if ((a.acceleration.x) < -3) {
    // Serial.println("Back Flip");
    is_Car_Flipped = 1;
```

```
 }
 else if ((a.acceleration.x) > 3) {
  // Serial.println("Front Flip");
  is_Car_Flipped = 1;
 }

 else {
  is_Car_Flipped = 0;
 }
}

void find_GPS() {
 unsigned long start = millis();
 do
 {
  while (ss.available())
   gps.encode(ss.read());

 } while (millis() - start < 2000);



 if (gps.location.isUpdated()) {

  latitude = (gps.location.lat());
  longitude = (gps.location.lng());

 }

 Serial.println(latitude, 6);
 Serial.println(longitude, 6);
}

void C_R_F() {
 show_Accident_Detected_LCD();

 find_GPS();

 for (int i = 0; i < 2; i++) {
  while (!Serial);

  //Being serial communication with Arduino and SIM800
  serialSIM800.begin(9600);
  delay(1000);
```

```
  //Set SMS format to ASCII
  serialSIM800.write("AT+CMGF=1\r\n");
  delay(1000);
  if (i == 0) {
   //Send new SMS command and message number
   serialSIM800.write("AT+CMGS=\"0768217964\"\r\n");
////////////////////////////////////////////////// Emergency Unit Number
  }
  else {
   serialSIM800.write("AT+CMGS=\"0705608924\"\r\n");
////////////////////////////////////////////////// Fire Brigade
  }
  delay(1000);


  //Send SMS content
  if (is_Careless == 1) {
   serialSIM800.write("A Car has faced a Collision, a Rollover and a fire due to the Carelessness in
following location: ");
  }
  else {
   serialSIM800.write("A Car has faced a Collision, a Rollover and a fire in following location: ");
  }

  serialSIM800.write("https://www.google.com/maps/place/");
  delay(1000);
  serialSIM800.print(latitude, 6); //Send SMS content
  serialSIM800.print(","); //Send SMS content
  serialSIM800.print(longitude, 6); //Send SMS content


  serialSIM800.write((char)26);
  delay(1000);
  delay(5000);
 }
 asm volatile ("  jmp 0"); // Reset Arduino

}

void C_R() {
 show_Accident_Detected_LCD();
 find_GPS();
```

```
  while (!Serial);

  //Being serial communication with Arduino and SIM800
  serialSIM800.begin(9600);
  delay(1000);

  //Set SMS format to ASCII
  serialSIM800.write("AT+CMGF=1\r\n");
  delay(1000);

  //Send new SMS command and message number
  serialSIM800.write("AT+CMGS=\"0768217964\"\r\n");
////////////////////////////////////////////////////////// Emergency Unit Number
  delay(1000);


  //  //Send SMS content
  if (is_Careless == 1) {
    serialSIM800.write("A Car has faced a Collision, and a Rollover due to the Carelessness in
following location: ");
  }

  else {
    serialSIM800.write("A Car has faced a Collision, and a Rollover in following location: ");
  }

  serialSIM800.write("https://www.google.com/maps/place/");
  delay(1000);
  serialSIM800.print(latitude, 6); //Send SMS content
  serialSIM800.print(","");    //Send SMS content
  serialSIM800.print(longitude, 6); //Send SMS content

  serialSIM800.write((char)26);
  delay(1000);

  delay(100);       // delay in between reads for stability
  asm volatile ("  jmp 0");
}

void C_F() {
  show_Accident_Detected_LCD();
  find_GPS();

  for (int i = 0; i < 2; i++) {
```

```
  while (!Serial);

  //Being serial communication with Arduino and SIM800
  serialSIM800.begin(9600);
  delay(1000);

  //Set SMS format to ASCII
  serialSIM800.write("AT+CMGF=1\r\n");
  delay(1000);

  //Send new SMS command and message number
  if (i == 0) {
    serialSIM800.write("AT+CMGS=\"0768217964\"\r\n");
////////////////////////////////////////////////////// Emergency Unit
  }
  else {
    serialSIM800.write("AT+CMGS=\"0705608924\"\r\n");
////////////////////////////////////////////////////// Fire Brigade
  }
  delay(1000);

  //Send SMS content
  serialSIM800.write("A Car has faced a Collision and a fire at following location : ");
  serialSIM800.write("https://www.google.com/maps/place/");
  delay(1000);
  serialSIM800.print(latitude, 6); //Send SMS content
  serialSIM800.print(","); //Send SMS content
  serialSIM800.print(longitude, 6); //Send SMS content

  serialSIM800.write((char)26);
  delay(1000);

  // Serial.println("SMS Sent!");
  //  delay(100);       // delay in between reads for stability
  delay(5000);
 }

 asm volatile ("  jmp 0");// Reset Arduino
}

void C() {
 show_Accident_Detected_LCD();
 find_GPS();
```

```
  while (!Serial);

  //Being serial communication with Arduino and SIM800
  serialSIM800.begin(9600);
  delay(1000);

  //Set SMS format to ASCII
  serialSIM800.write("AT+CMGF=1\r\n");
  delay(1000);

  //Send new SMS command and message number
  serialSIM800.write("AT+CMGS=\"0768217964\"\r\n");
////////////////////////////////////////////////////////// Emergency Unit Number
  delay(1000);


  //Send SMS content
  serialSIM800.write("A Car has faced a Collision in following location: ");
  serialSIM800.write("https://www.google.com/maps/place/");
  delay(1000);
  serialSIM800.print(latitude, 6); //Send SMS content
  serialSIM800.print(","); //Send SMS content
  serialSIM800.print(longitude, 6); //Send SMS content

  //Send Ctrl+Z / ESC to denote SMS message is complete
  serialSIM800.write((char)26);
  delay(1000);

  // Serial.println("SMS Sent!");
  delay(100);       // delay in between reads for stability
  asm volatile ("  jmp 0");// Reset Arduino
}



void R_F() {
  show_Accident_Detected_LCD();
  find_GPS();


  for (int i = 0; i < 2; i++) {
    while (!Serial);
```

```
  //Being serial communication with Arduino and SIM800
  serialSIM800.begin(9600);
  delay(1000);

  //Set SMS format to ASCII
  serialSIM800.write("AT+CMGF=1\r\n");
  delay(1000);

  //Send new SMS command and message number
  if (i == 0) {
    serialSIM800.write("AT+CMGS=\"0768217964\"\r\n");
////////////////////////////////////////////////////// Emergency Unit Number
  }
  else {
    serialSIM800.write("AT+CMGS=\"0705608924\"\r\n");
////////////////////////////////////////////////////////// Fire Brigade
  }
  delay(1000);

  //Send SMS content
  serialSIM800.write("A Car has faced a Rollover and a fire at following location : ");
  serialSIM800.write("https://www.google.com/maps/place/");
  delay(1000);
  serialSIM800.print(latitude, 6); //Send SMS content
  serialSIM800.print(","); //Send SMS content
  serialSIM800.print(longitude, 6); //Send SMS content

  //Send Ctrl+Z / ESC to denote SMS message is complete
  serialSIM800.write((char)26);
  delay(1000);
  delay(5000);

 }

 asm volatile ("  jmp 0");// Reset Arduino
}

void R() {
 show_Accident_Detected_LCD();
 find_GPS();

 while (!Serial);

 //Being serial communication with Arduino and SIM800
```

```
  serialSIM800.begin(9600);
  delay(1000);

  //Set SMS format to ASCII
  serialSIM800.write("AT+CMGF=1\r\n");
  delay(1000);

  //Send new SMS command and message number
  serialSIM800.write("AT+CMGS=\"0768217964\"\r\n");
////////////////////////////////////////////////////////// Emergency Unit Number
  delay(1000);

  //Send SMS content
  serialSIM800.write("A Car has faced a Rollover at following location : ");
  serialSIM800.write("https://www.google.com/maps/place/");
  delay(1000);
  serialSIM800.print(latitude, 6); //Send SMS content
  serialSIM800.print(",");     //Send SMS content
  serialSIM800.print(longitude, 6); //Send SMS content

  //Send Ctrl+Z / ESC to denote SMS message is complete
  serialSIM800.write((char)26);
  delay(1000);

  // Serial.println("SMS Sent!");
  delay(100);       // delay in between reads for stability

  asm volatile ("  jmp 0");// Reset Arduino
}

void show_Accident_Detected_LCD() {
  lcd.clear();
  lcd.setCursor(4, 0);
  lcd.print("Accident");
  lcd.setCursor(3, 1);
  lcd.print("Detected.!");
}
```