University of Moratuwa

2022

Final Evaluation Report (IN 1900)

Level 1

Safety and Security System for Bicycle

Group 23

Faculty of Information Technology

Name of the supervisor: Mr.B.H.Sudantha

Dean/senior lecturer

Faculty of Information Technology

Name of the supervisor: Ms. Nipuni Chandimali

Lecturer

Faculty of Information Technology

Name of the instructor: Mr. Pasindu Udugahapattuwa

Assistance Lecturer

Faculty of Information Technology

Signature of the supervisors: 1. ……

2 ……

Submission date: 1th of October 2023

Group members

Rajapakse N.C.M.K.	**214165D**
Weerasinghe C.L.	214223F
Prilakshana T.P.G.D.	214160H
Paranavithana D.H.	214144M
Sathsara M.G.	214187V

TABLE OF CONTENTS

TABLES AND FIGURES

Tables

Figures

INTRODUCTION

There is a major fuel crisis in Sri Lanka nowadays. The prices of all kinds of fuel are skyrocketing. The government is failing to supply sufficient amounts of petrol and diesel. Hence, Sri Lankans are searching for more cost-effective transportation options. Out of them, A considerable amount of people have switched to foot bicycles. Generally riding bicycles can offer a wide range of benefits.

One of the most important benefits of cycling is improved public health. Regular cycling can improve cardiovascular health, increase muscle strength, and reduce the risk of chronic diseases such as obesity, diabetes, and heart disease. Cycling is also a sustainable mode of transportation that does not contribute to air pollution. This makes it an environmentally friendly alternative to driving, which is a major contributor to air pollution and climate change. In addition to the benefits to public health and the environment, Another important benefit of promoting cycling is reduced traffic congestion. By encouraging people to cycle instead of driving, there will be fewer cars on the road, leading to reduced traffic congestion and shorter commute times. This can also help reduce the amount of time people spend in traffic.

Overall, making bicycles more popular can lead to a healthier way of living and more sustainable transportation. Encouraging people to cycle instead of driving can help promote public health, reduce environmental impact, decrease traffic congestion, and provide economic benefits. Despite all those benefits bicycles are not widely used for transportation in Sri Lanka. While some people use bicycles for commuting or running errands, the vast majority of people believe that cycling is a leisure activity, rather than a practical mode of transportation.

One of the many challenges of cycling in Sri Lanka is the lack of safe cycling infrastructure. Many roads in Sri Lanka are narrow and crowded, and there are few designated bike lanes or other cycling infrastructure. This can make it dangerous for cyclists to share the road with other vehicles, particularly in urban areas. In addition, there is a lack of awareness among drivers about how to share the road safely with cyclists.

Despite these challenges, there is a growing interest in cycling in Sri Lanka. As the government and other organizations invest in cycling infrastructure and promote cycling as a healthy and sustainable mode of transportation, it is possible that cycling will become more popular in the country.

LITERATURE SURVEY

Introduction

Road accidents are common in Sri Lanka. Most of the time most damage occurs to the person with a smaller vehicle in a collision. People are more skeptical about riding bicycles or letting a loved one ride a bicycle through congested areas due to this reason. We are going to introduce an accident alert system so people can take action as soon as they get the Message.

Nowadays the value of a foot bicycle has increased drastically due to the ongoing fuel crisis. With that, the market price of a bicycle also has risen. Hence now bicycles are more prone to thievery, more than ever. Bicycle theft is a common problem in Sri Lanka nowadays, and it can be difficult for cyclists to protect their bikes from theft. Bicycles are very easy to get stolen in urban areas. One of the main reasons for this is a lack of adequate security Measures.

In addition to the lack of security measures, many Sri Lankan households do not have a secure place to store their bicycles. This means that bikes are often left outside or in areas that are easily accessible to thieves. This makes it easy for thieves to steal bikes, particularly at night when they are less likely to be seen. Our team saw this as the main problem that bicycle riders have to face.

In Sri Lanka, it is not uncommon for parents to be hesitant to allow their children to ride bicycles. One of the main reasons for this is safety concerns. Many roads in Sri Lanka are narrow and crowded, with little to no designated bike lanes or cycling infrastructure. This can make it dangerous for cyclists, particularly children who may not have the experience or skills to navigate traffic safely.

Methodology

The information upon which this survey is based was gathered using these methodologies. This is completed online by using a web-based survey program. An online search was conducted using resource websites and search engines to gather further information.

1. Through reading published reports about the challenges bicycle riders have to face.
2. Through searching the official websites of the manufacturing companies.
3. Through reading valuable research papers about bicycle riding and its impacts.
4. Searching official websites of online shopping centers to gather information about prices of the bicycles in Sri Lanka.

Conclusion

We came up with the following solutions to overcome the above-mentioned issues.

- A digital lock developed using a servo motor is used to ensure the security of the bike. With that users can lock and unlock the bicycle using the mobile application. Riders are encouraged to put this lock when the bicycle is parked for a longer time.

- There is a GPS tracker to locate the bike in case of an emergency. If there is an accident alert or in case of theft users can locate the bike using a text message.

- An accident alert system with a GSM module and accelerometer to measure the sudden change in acceleration. The accelerometer will be positioned on the lower half of the bicycle. down a push button and prevent the message from being delivered to the emergency contact. Whenever there is a sudden change in acceleration with angle change while riding the bicycle it will be recognized as an accident. It will ring the buzzer for up to one minute and then the alert message will be sent. If it is a minor accident, the rider can turn it off and prevent the message from sending.

- Hall sensor is used to develop a speed detector for the bike. The rider can utilize that feature to monitor their speed and adjust their pace as needed to maintain a safe and efficient commute.

- The whole system will be powered with a rechargeable battery. Users can see the remaining battery charging as a percentage through the App. A charger developed by the team will be provided and the user can use it to charge the battery with home electricity.

AIM AND OBJECTIVES

Aim:

The Aim of this initiative is to enhance the safety of bicycle riders and safeguard their bicycles, while making bicycle a preferred transportation method.

Objectives:

- **To establish worry-free bicycle transportation.**

This objective focuses on creating a worry-free experience for bicycle riders by implementing various safety and convenience features. It includes the integration of an accident alert system, which will promptly notify emergency contacts in case of an accident or emergency. Additionally, an electric lock will be installed to enhance bicycle security, allowing riders to lock and unlock their bicycles electronically. Furthermore, the inclusion of a speedometer will enable riders to monitor their speed, promoting responsible riding and better awareness of their riding conditions. Lastly, the incorporation of a GPS tracker will help locate the bicycle in case of theft or loss, providing peace of mind to riders and enhancing the overall safety of bicycle transportation.

- **To provide security measures for parked bicycles.**

This objective aims to enhance the security of parked bicycles. It involves the installation of an electric lock system, which allows riders to securely lock their bicycles when not in use. Additionally, the GPS tracker will play a crucial role in locating stolen or misplaced bicycles, serving as a deterrent to theft and increasing the chances of recovering stolen bicycles. These security measures will not only protect the investment of bicycle owners but also encourage more people to opt for bicycle commuting, knowing that their bicycles are safe and protected.

- **To promote bicycle usage as a preferred commuting option.**

This objective is centered around promoting bicycles as a preferred mode of transportation for daily commuting. The implementation of safety features such as the accident alert system and the inclusion of a speedometer will instill confidence in riders by offering a safer and more controlled riding experience. Furthermore, the added security measures, including the electric lock and GPS tracker, will reduce concerns about theft and bicycle loss, making bicycles a more appealing choice for daily commuting. By achieving this objective, the project aims to encourage more people to adopt bicycles as an eco-friendly, healthy, and convenient mode of transportation for their daily commutes.

ANALYSIS AND DESIGN

Block Diagram of the device

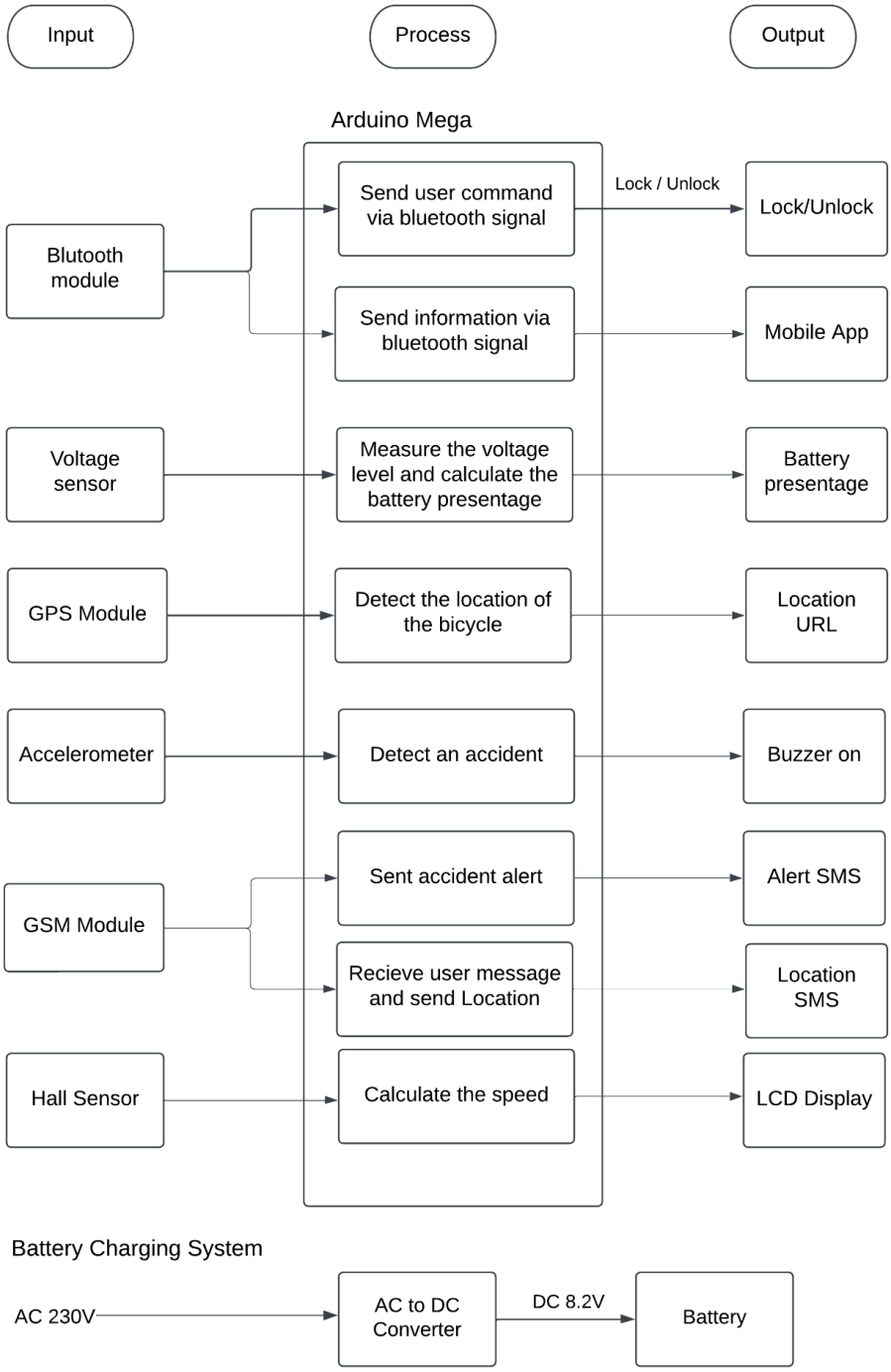

Diagram 4.1-Block diagram of the machine

Schematic diagram of the System

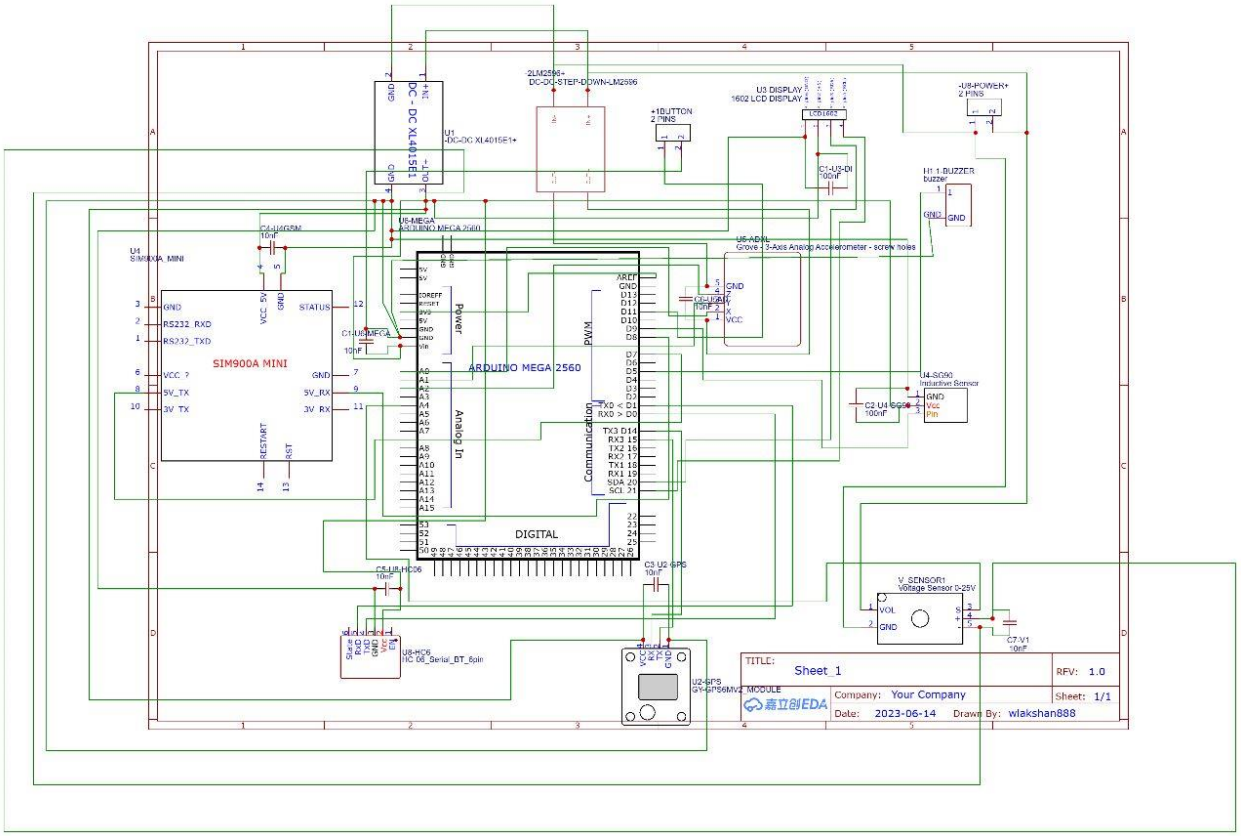

Diagram 4.2-Schematic diagram of the machine

PCB design of the System

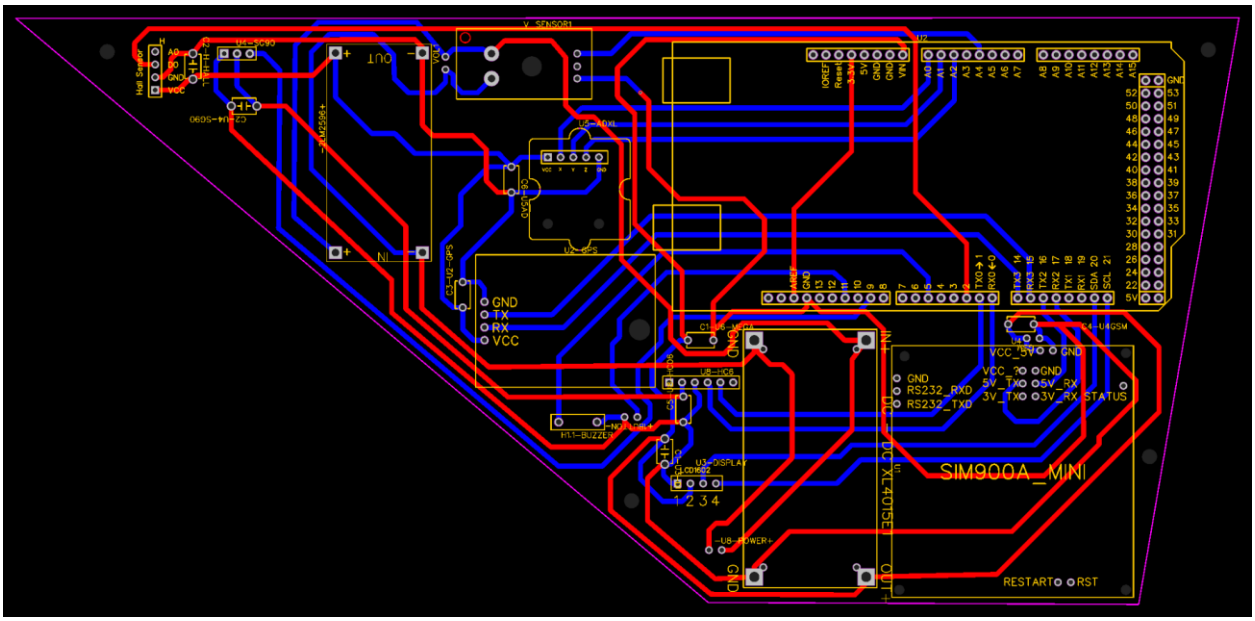

Diagram 4.3-Pcb design of the System

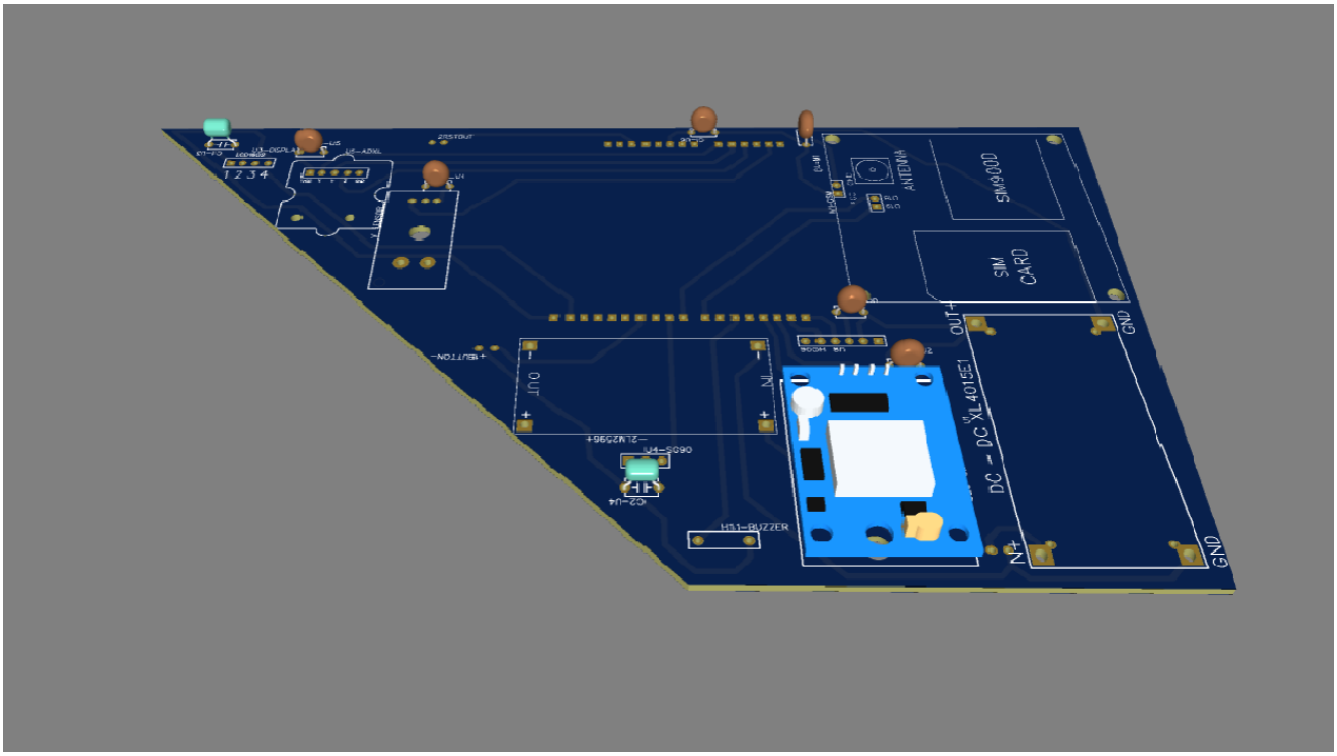

Diagram 4.4-3D view of PCB

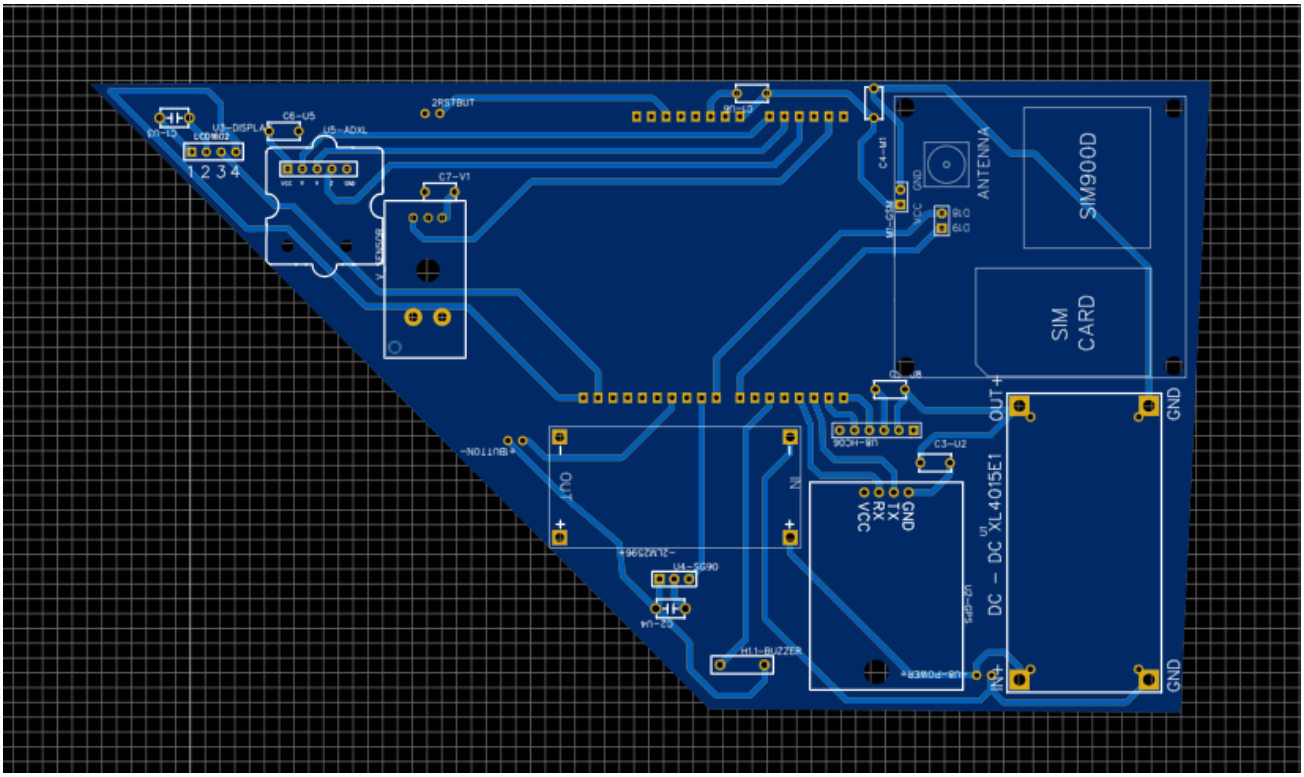

Diagram 4.5-2D view of PCB

FINAL PRODUCT

figure 4.6 final product

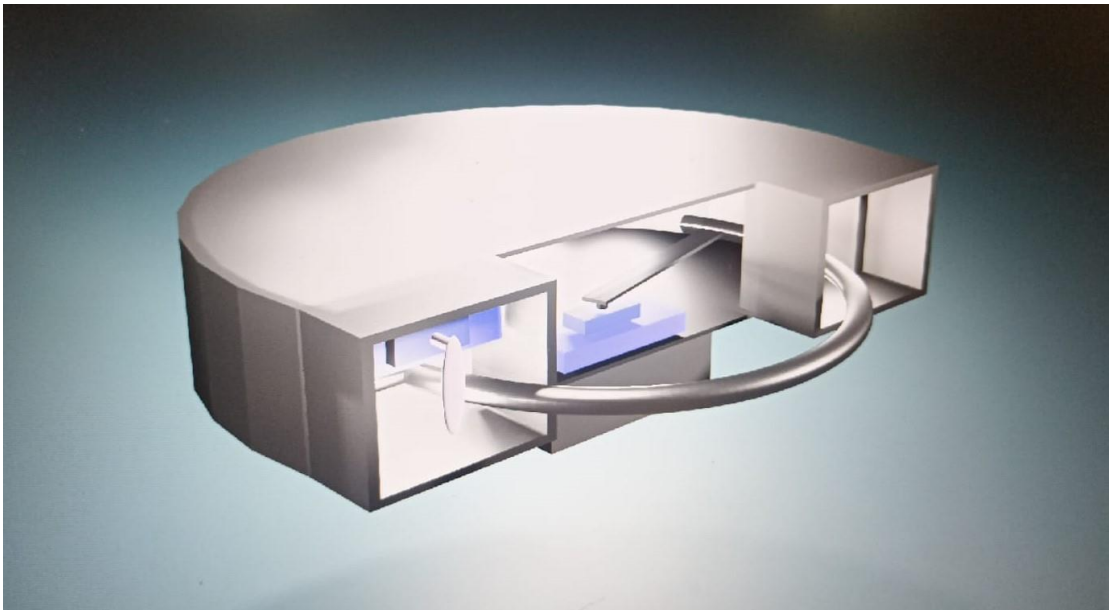

figure 4.7 Electric lock

TESTING AND IMPLEMENTATION

All members of our group diligently studied the assigned modules. Our project utilized the Arduino Mega microcontroller, and each member thoroughly familiarized themselves with its features. Team members also took responsibility for creating circuit diagrams and coding modules individually. Following these individual efforts, we consolidated the circuit diagram.

During the development process, we encountered several challenges, with the primary issue being related to GPS signal stability. Initially, we had intended to use the neo 6m module, but we eventually switched to the neo8m due to its superior accuracy and signal strength. To ensure the device's waterproofing, we applied glue as a protective measure.

We conducted numerous tests to determine the impact level on a bicycle that would accurately replicate an accident scenario. Based on these tests, we fine-tuned our accident detection algorithm. The system now triggers an accident alert only when both a significant impact and a fall are detected simultaneously. To prevent unnecessary alert messages, we incorporated a system with a buzzer and a push button.

In our design, two servo motors were employed to create an electric lock, with one responsible for turning the lever and the other for locking it in place. The accident alert system functions only when the lock is disengaged, while the location-finding feature remains operational as long as the batteries have sufficient power.

To simulate a multithreaded architecture, essential for simultaneous operation while the user rides the bicycle, we utilized the millis() function. We rigorously tested the system to ensure that all functions operated smoothly without interfering with one another.

For speed detection, we implemented a hall sensor, which records rotations every 3 seconds and displays the speed in kilometers per hour on the front display. Prior to coding this feature, we accurately measured the wheel's radius and conducted the necessary calculations.

Regarding the GSM module, we programmed it to respond to the "Location" message from any phone number. It checks whether the sender's number matches the authorized phone numbers stored in the module and sends the location as a Google Maps link only if a match is found.

TOTAL COST AND EXPENDITURE

Components	Quantities	Price per unit(Rs)	Total price(Rs)
Adxl335	1	460	460
GPS	1	3500	3500
GSM	1	2000	2000
Voltage sensor	1	110	110
Bluetooth module	1	1100	1100
Buzzer	1	40	40
Button	1	40	40
Switch	1	50	50
Plastic box	1	50	50
servo motors	2	600	1200
Batteries	4	1000	4000
Glass pieces	1	50	50
Metal sheet	1	800	800

Components	Quantities	Price per unit(Rs)	Total price(Rs)
Plug top	1	150	150
Capacitors	15	10	150
wires	5 m	80	400
Diodes	5	6	30
Resistors	5	6	30
PCB print	1	1700	1700
JST cables	10	25	250
Male female connectors	1	240	240
Pin headers	2 set	25	50
Arduino mega	1	5000	5000
Buck converters	2	400	800
Display	1	800	800
BMS	1	150	150

Components	Quantities	Price per unit(Rs)	Total price(Rs)
Battery casing	1	150	150
Hall sensor	1	150	150
Magnet	1	100	100
GPS antenna & connector	1	350	350
I2C	1	250	250
Other			500
Total			24,650

Table1. Table of expenditure

REFERENCES

- “1200+ Latest Electronics Engineering Projects Ideas” NevonProjects, nevonprojects.com/project-ideas/electronics-ideas/. Accessed 14 Feb. 2023.

- “AliExpress - Online Shopping for Popular Electronics, Fashion, Home & Garden, Toys & Sports, Automobiles and More products” AliExpress, www.aliexpress.com/?. Accessed 14 Feb. 2023.

- Arduino Tutorial 1: Setting Up and Programming the Arduino for Absolute Beginners.” YouTube, uploaded by Paul McWhorter, 31 May 2019, youtu.be/fJWR7dBuc18.

- “Contact Us – Scion Electronics” scionelectronics.com/contact-us/. Accessed 14 Feb. 2023.

- “DB0004 - Arduino UNO Normal Development Board with USB Cable” Arduino UNO Normal Development Board with USB Cabl, tronic.lk/product/arduino-uno-normal-development-board-with-usb-cable. Accessed 14 Feb. 2023.

- “Let's Make Arduino Sing with a Buzzer!!!!!!” projecthub.arduino.cc/. Accessed 14 Feb. 2023.

- “MIT App Inventor” appinventor.mit.edu/. Accessed 14 Feb. 2023.

APPENDIX A – INDIVIDUAL CONTRIBUTION
Individuals' contribution to the project

Rajapakse N.C.M.K. - 214165D

As the team leader, I'm more than happy to say that I'm very proud of what we have achieved together. I was responsible for the GPS(global positioning system) and GSM modules. The GY-NEO-8M module is an advanced GPS module based on uBlox m8N with an active antenna. It serves to detect the locations when required. SIM900A is a GSM (Global System for Mobile Communications) module that is used in projects that require communication over GSM networks. It is employed to automatically send emergency messages to designated contacts in the event of an accident detection or share location information when needed.

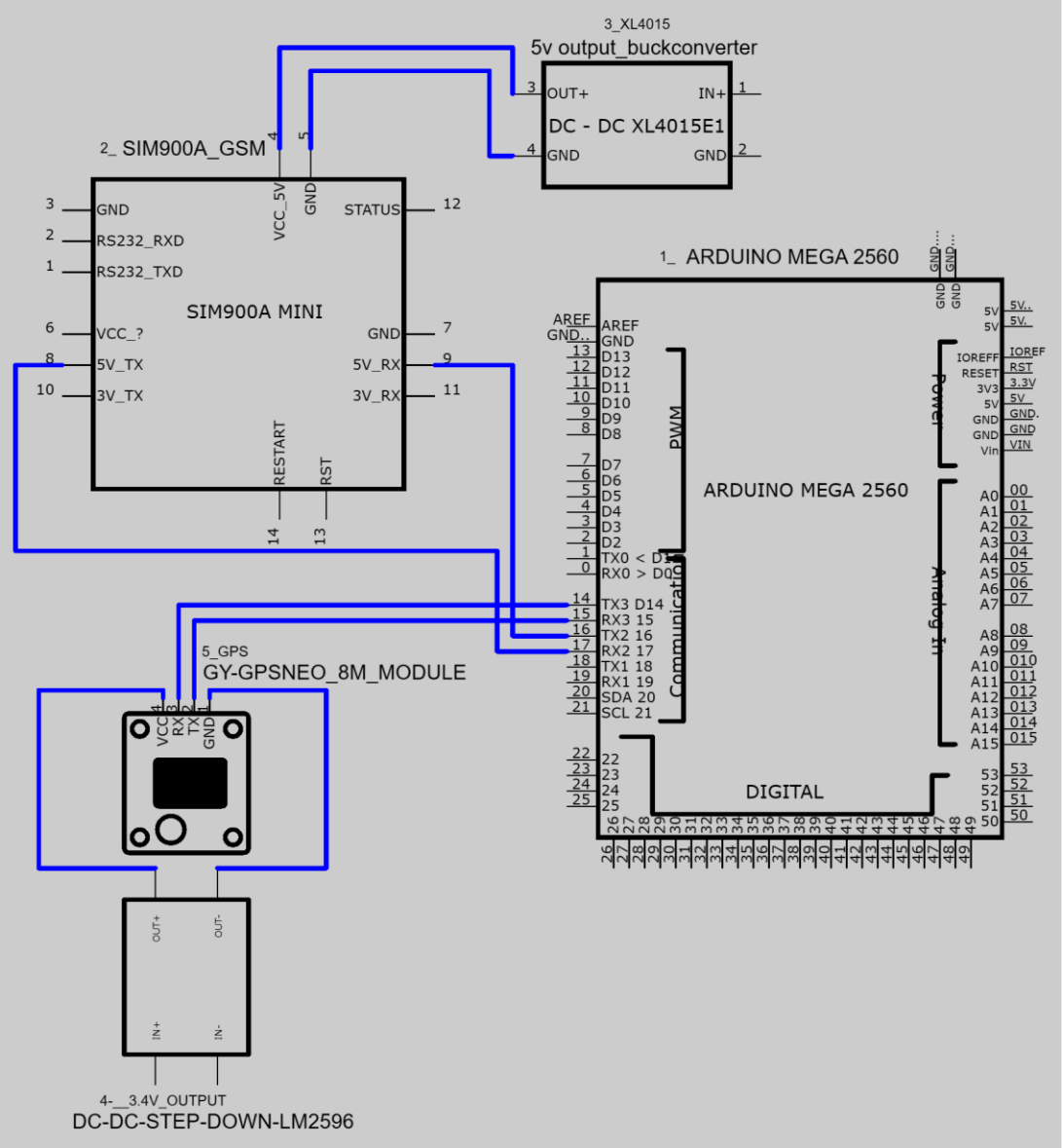

Diagram 4.i.1-Schematic diagram of location alert system

Code for GPS Module and GSM Module

```
#include <TinyGPS++.h>

TinyGPSPlus gps;  // Create a TinyGPS++ object
double longi;
double lati;
boolean GPS = true;
unsigned long previousMillis1 = 0;
String EmergencyContact = "+94766251587";
String Contact1 = "+94766251587";
String Contact2 = "+94764512428";
String url = "Location not available";
String phoneNum = "";

void setup() {

  // Begin serial communication with Arduino and SIM900A
  Serial2.begin(9600);
  // Begin serial communication with Arduino and SIM900A
  Serial3.begin(9600);
 Serial2.println("AT+CMGF=1"); // Change sim900A Serial to SIM900A
 delay(1000);
 Serial2.println("AT+CNMI=2,2,0,0,0"); // Configure module
}

void loop() {

 unsigned long currentMillis1 = millis();
 if (currentMillis1 - previousMillis1 > 58000) {
 getGPS();
 previousMillis1 = currentMillis1;
  }

 readMessege();
}

void getGPS() {
 boolean newData = false;
 for (unsigned long start = millis(); millis() - start < 2000;) {
 while (Serial3.available() > 0 && GPS) {
```

```
 if (gps.encode(Serial3.read())) {
 newData = true;
 break;
 }
 }
 }
if (newData) //If newData is true
 {
 lati = gps.location.lat();
 longi = gps.location.lng();
 newData = false;
 }

 if (gps.location.isValid()) {
 url = "https://www.google.com/maps/search/?api=1&query=" + String(lati, 6) + "," + String(longi, 6);

 } else {
 url = " Location: Not Available";
 }

 // If 5000 milliseconds pass and there are no characters coming in over the serial port
 if (millis() > 5000 && gps.charsProcessed() < 10) {
 url = " Location: Not Available";
 return;
 }
}
void readMessege() {
 if (Serial2.available()) {
 String response = Serial2.readString();
 Serial.println(response);
 //remove "/n"
 String message = response;
 message.replace("\n", "");
 String ATcommand = message.substring(0, response.indexOf(':') - 1);
 ATcommand.trim(); // Trim white spaces

 if (ATcommand == "+CMT") {
 // Find the index of the first double quote
 int firstDoubleQuoteIndex = message.indexOf('\"');
 // Extract the phone number
 phoneNum = message.substring(firstDoubleQuoteIndex + 1, message.indexOf('\"',
firstDoubleQuoteIndex + 1));
 int lastDoubleQuoteIndex = message.lastIndexOf('\"');
 // Extract the message
```

```
 String messageContent = message.substring(lastDoubleQuoteIndex + 1);
 messageContent.trim();
 //send the location
 if (phoneNum == Contact1 || phoneNum == Contact2) { //can add more phone numbers here
 if (messageContent.equalsIgnoreCase("Location")) {
 GPS = false;
 SendAlert(phoneNum, url);
 }
 }
 }
 }
}
void SendAlert(String phone, String link) {

 Serial.println("Initializing...");
 delay(1000);

 String message = link;

 Serial2.println("AT"); // Handshaking with SIM900
 updateSerial();

 Serial2.println("AT+CMGF=1"); // Configuring TEXT mode
 updateSerial();

 Serial2.print("AT+CMGS=\"" + phone + "\"\r");
 updateSerial();

 Serial2.print(message); // Text content
 updateSerial();
 Serial2.write(26);
 GPS = true;
}
void updateSerial() {
 long timeout = millis() + 1000; // Set a timeout of 1 second
 while (!Serial2.available() && millis() < timeout) {
 // Wait for data or timeout
 }

 while (Serial2.available()) {
 Serial.write(Serial2.read());
 }
}
```

```
void updateSerial() {
 delay(500);
 while (Serial.available()) {
 Serial2.write(Serial.read());
 }
 while (Serial2.available()) {
 Serial.write(Serial2.read());
 }
}
// Take samples and return the average
int ReadAxis(int axisPin) {
 long reading = 0;
 analogRead(axisPin);
 delay(1);
 for (int i = 0; i < sampleSize; i++) {
 reading += analogRead(axisPin);
 }
 return reading / sampleSize;
}


/*============================= Turn on and off buzzer====================*/
void turnOnBuzzer() {
 digitalWrite(BUZZER_PIN, HIGH); // turn on the buzzer
 buzzerOn = true;
}

void turnOffBuzzer() {
 digitalWrite(BUZZER_PIN, LOW); // turn off the buzzer
 buzzerOn = false;
}
```

Paranavithana D.H. - 214144M

I worked on creating the mobile app, made sure the Bluetooth module could send and get data, finished all the coding, and learned about servo motors and how to program them. We used a Bluetooth module to connect our software to hardware components. Our project involves creating a mobile app that lets users control a lock using Bluetooth. We used SG90 Servo Motors, which are small yet powerful motors that can turn up to 180 degrees.

To build the lock mechanism, we used two of these servo motors. We developed our mobile app using MIT App Inventor, a user-friendly tool for making Android apps. MIT App Inventor is a web-based platform initially created by Google and later adopted by MIT. It's known for its drag-and-drop interface, which makes app development easier.

Our mobile app not only controls the bicycle lock but also provides real-time updates on the battery status through Bluetooth.

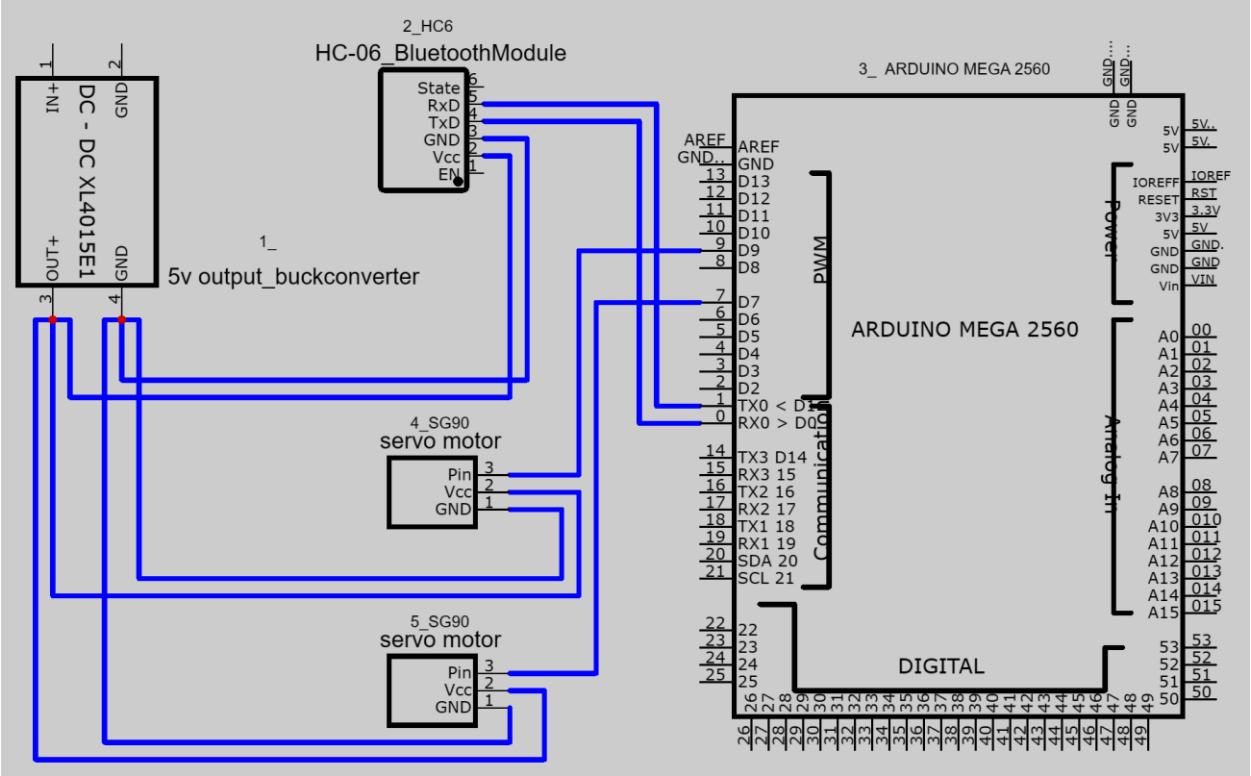

Diagram 4.ii.1-Schematic diagram of lock control by mobile app

Code for Bluetooth Module and Servo Motors

```
#include <Servo.h>

Servo pcc;
Servo pcc1;
char val;
unsigned long previousMillis4 = 0;
unsigned long previousMillis5 = 0;
boolean pccstate1 = false;
boolean pccstate2 = false;
boolean lockon = false;

void setup() {

  pcc.attach(9);   //servo
  pcc1.attach(7);  //servo2

}

void loop() {
  if (Serial.available()) {
    val = Serial.read();
    Serial.println(val);
 if (val == '1') {
 pcc.write(0);
 pccstate1 = true;
 previousMillis4 = millis();
 // Store the current time

 // pinMode(15, LOW);

 } else if (val == '0') {
 pcc1.write(30);
 pccstate2 = true;
 previousMillis5 = millis(); // Store the current time

 // pinMode(15, LOW);
 }
 }

 // Check the delay for servo1
 unsigned long currentMillis = millis();
 if (currentMillis - previousMillis4 > 2000 && pccstate1) {
```

```
 servo1();
 pccstate1 = false;
 lockon = true; ///lock eka wahila
 }

// Check the delay for servo2
 if (currentMillis - previousMillis5 > 2000 && pccstate2) {
 servo2();
 pccstate2 = false;
 lockon = false;
 }
}

void servo1() {
 pcc1.write(90);
}

void servo2() {
 pcc.write(180);
}
```

Code for Moblie App

Diagram 4.ii.2-Block diagram of the mobile app

Prilakshana T.P.G.D. - 214160H

I began by learning about the ADXL335 accelerometer, a sensor that measures acceleration in three directions. Then, I designed an algorithm to detect accidents and programmed it into our system.

To create an accident alert system, I implemented a button and a buzzer. We used the ADXL335 accelerometer to sense impacts and tilts on the bicycle. When an accident occurs, the buzzer activates and rings continuously for 30 seconds. If it's a minor accident, individuals can press a button within those 30 seconds to deactivate the buzzer and stop the accident alert.

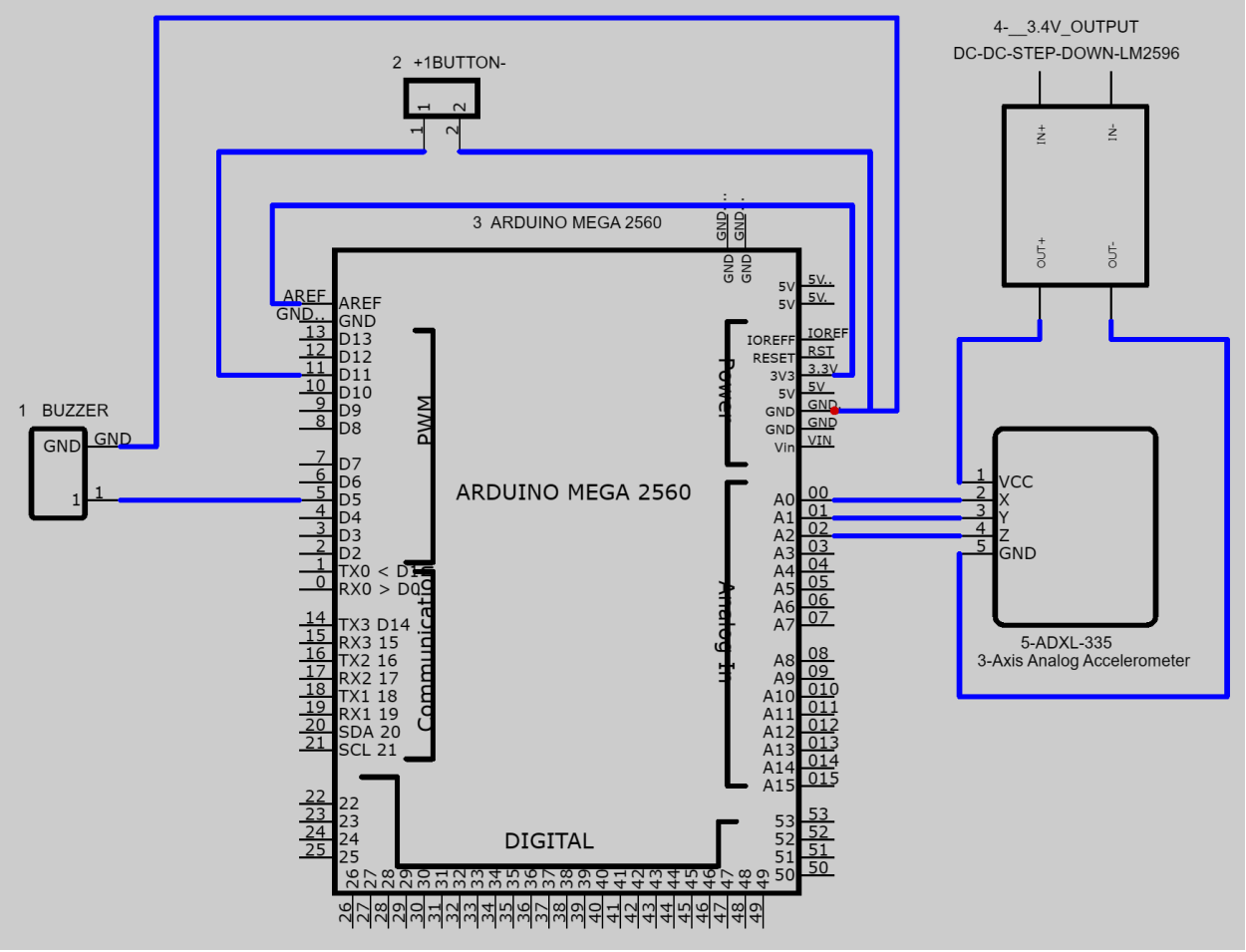

Diagram 4.iii.1-Schematic diagram of accident detect unit

Code for ADXL335 , Button and Buzzer

```
//********************************** FOR ACCIDENT *************************

const int xInput = A0;
const int yInput = A1;
const int zInput = A2;

int RawMin = 0;
int RawMax = 1023;
int count = 0;
const int sampleSize = 10;
byte updateflag;
byte updateflag1;
byte updateflag2;

//************************************ for buzzer ********************************
const int BUTTON_PIN = 11;  // Arduino pin connected to button's pin
const int BUZZER_PIN = 5;   // Arduino pin connected to Buzzer's pin
//*******************************************************************************

int xaxis = 0, yaxis = 0, zaxis = 0;
int deltx = 0, delty = 0, deltz = 0;
int vibration = 2, devibrate = 75;
int magnitude = 0;
int sensitivity = 150;
bool buttonPressed = false;  // flag to track if the button has been pressed
bool buzzerOn = false;       // flag to track if the buzzer is currently
//bool resetLoop = false;    // flag to track if the loop should be reset
bool codeExecuted = false;
boolean impactDetected = false;
//Used to run impact routine every 2mS.
unsigned long time1;
unsigned long impactTime;
unsigned long alertDelay = 30000;  //30 seconds

void setup() {

  analogReference(EXTERNAL);
  Serial.begin(9600);

  xaxis = analogRead(xInput);
  yaxis = analogRead(yInput);
  zaxis = analogRead(zInput);

}
void loop() {
accidentdetected();
```

```
}

void accidentdetected() {

  //call impact routine every 2mS
  if (micros() - time1 > 1999) Impact();  // 2000micro seconds = 2 ms
  fall();
# if (updateflag1 == 1 && updateflag2 == 1 ) {
 updateflag = 1;
 }
 if (updateflag1 == 1 && updateflag2 == 0) {
 updateflag = 0;
 }

 if (updateflag == 1 ) {
 updateflag = 0;
 updateflag2 = 0;
 updateflag1 = 0;
 if (lockon == false) {

 turnOnBuzzer();
 lcd.setCursor(0, 1);
 lcd.print("Accident");
 lcd.setCursor(9, 1);
 lcd.print(magnitude);

 impactDetected = true;
 impactTime = millis();
 }
 } //end of updateflag

 if (impactDetected == true) {
 if ((millis() - impactTime >= alertDelay) && digitalRead(BUZZER_PIN) == HIGH) {
 turnOffBuzzer();

GPS = false;
 SendAlert(EmergencyContact, "Accident Detected! " + url);
 impactDetected = false;
 impactTime = 0;
 }
 } //end of impactDetected delay

 if (digitalRead(BUTTON_PIN) == LOW) {
 buttonPressed = true;
 }
 if (buttonPressed == true) {
 turnOffBuzzer();
 buttonPressed = false;
```

```
 }
}

void Impact() {
  //-------------------------------------------------------------
  time1 = micros();  // resets time value
  //-------------------------------------------------------------
  int oldx = xaxis;  //store previous axis readings for comparison
  int oldy = yaxis;
  int oldz = zaxis;

  xaxis = analogRead(xInput);
  yaxis = analogRead(yInput);
  zaxis = analogRead(zInput);

   //-------------------------------------------------------------
  //loop counter prevents false triggering. Vibration resets if there is an impact. Don't detect new changes
until that "time" has passed.
  vibration--;
  if (vibration < 0)
     vibration = 0;

  if (vibration > 0)
       return;
  //-------------------------------------------------------------
  deltx = xaxis - oldx;
  delty = yaxis - oldy;
  deltz = zaxis - oldz;

  //Magnitude to calculate force of impact.
  magnitude = sqrt(sq(deltx) + sq(delty) + sq(deltz));

  if (magnitude >= sensitivity)  //impact detected
  {
    updateflag1 = 1;
    // reset anti-vibration counter
    vibration = devibrate;
  }

  else {
    magnitude = 0;
  }
}

Gvoid fall() {
  // read new state
  // int buttonState = digitalRead(BUTTON_PIN);
```

```
  // Read raw values
  int xRaw = ReadAxis(xInput);
  int yRaw = ReadAxis(yInput);
  int zRaw = ReadAxis(zInput);


  if ((yRaw > 500 && yRaw < 565) && ((zRaw > 400 && zRaw < 435) || (zRaw > 600 && zRaw < 650))) {
    if (!codeExecuted) {
      updateflag2 = 1; //fall detected
      codeExecuted = true;
    }
  } else if ((yRaw > 595 && yRaw < 6630) && (zRaw > 470 && zRaw < 550)) {
    codeExecuted = false;
    buttonPressed = false;
  }
}
```

Sathsara M.G. - 214187V

I started by learning about the hall sensor module and programming it to measure the bicycle's speed. Then, I studied the LCD screen and sensor module, and I programmed them as well. We used the hall sensor module to detect the rotation of the bicycle wheel and calculate the speed based on the frequency and duration of these rotations. The 3144 Hall switch, an integrated circuit, played a key role in detecting magnetic fields and converting them into digital voltage signals. For displaying the bicycle's speed, we incorporated a 16x2 LCD screen into our system.

As part of our project, I was also responsible for designing and constructing the bicycle frame and the electric lock.

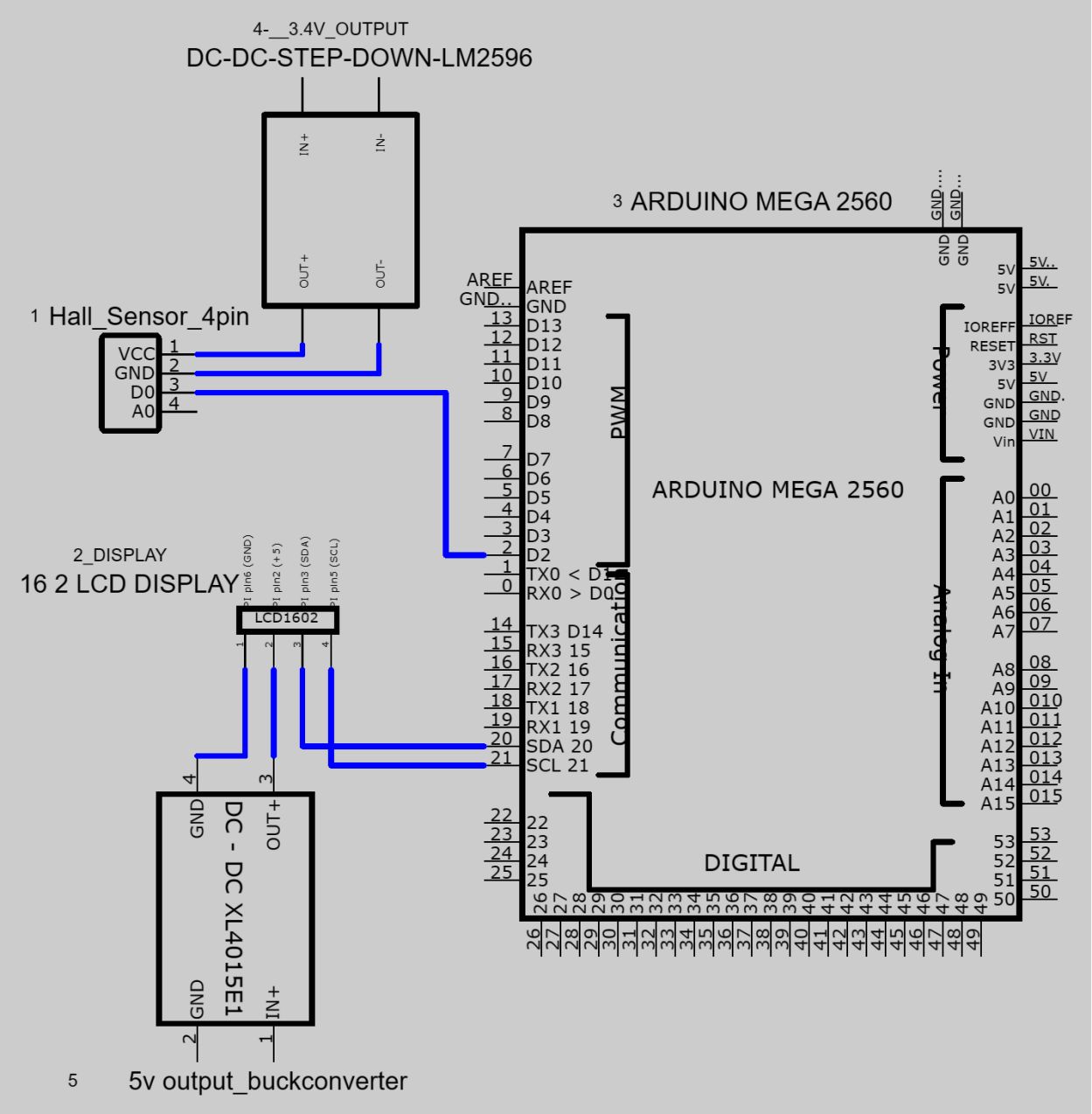

Diagram 4.iv.1-Schematic diagram of speedometer

Code for Hall Sensor and LCD Screen

```
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 16, 2); // I2C address 0x27, 16 column and 2 rows

// Constants
const int hallPin = 2;     // Hall sensor pin
const float wheelCircumference = 2 * 3.14 * 0.2; // Wheel circumference in meters (adjust as needed)
const unsigned long interval = 2000;   // Measurement interval in milliseconds

// Variables
volatile unsigned int hallCount = 0;   // Hall sensor count
unsigned long previousMillis = 0;
float speedKmph = 0.0;

void setup()
{
  lcd.init(); // initialize the lcd
  lcd.backlight();
  // Initialize serial communication
  Serial.begin(9600);

  // Attach interrupt to the Hall sensor pin
  attachInterrupt(digitalPinToInterrupt(hallPin), countPulse, RISING);
}

void loop()
{
 lcd.clear(); // clear display
 unsigned long currentMillis = millis();

 // Calculate the elapsed time
 unsigned long elapsedTime = currentMillis - previousMillis;

 // Check if the measurement interval has passed
 if (elapsedTime >= interval) {
 // Calculate speed in km/h
 float speedMps = (float)hallCount * wheelCircumference / elapsedTime; // Speed in meters per second
 speedKmph = speedMps * 3600; // Speed in km/h
 // Reset the hall sensor count and previous time
 hallCount = 0;
 previousMillis = currentMillis;
 }
 lcd.clear();
 lcd.print("speed:");
 lcd.print(speedKmph);
 lcd.print("kmph");
```

```
}
// Interrupt service routine to count pulses from the Hall sensor
void countPulse() {
 hallCount++;
}
```

Weerasinghe C.L. - 214223F

My role encompassed several pivotal components and tasks within the project:

PCB Design:
I took charge of the PCB design phase, utilizing EasyEDA software to meticulously plan the layout and interconnections of electronic elements on the board.
Frame Design and Construction: I was responsible for the design and physical construction of the device's frame, providing the essential structure and enclosure for our electronic components.
Charger Design (AC-DC Converter): I led the design of the charger, a critical aspect that involved converting a 230V AC input into a precise 8V-8.4V DC output. This charger was purpose-built for efficiently recharging lithium-ion batteries with the assistance of a 2S-BMS (Battery Management System).

Voltage Sensor and Programming:
My duties extended to comprehending and implementing the voltage sensor circuit, which relied on a voltage divider circuit featuring 30K and 7.5K resistors. Additionally, I programmed the system to accurately gauge voltage levels and calculate the remaining battery capacity as a percentage.

Power Supply Unit: I played a pivotal role in the design of the power supply unit, which entailed the utilization of two step-down buck converters, namely the LM2596 and XL4015. These converters were instrumental in reducing the battery voltage to provide suitable power levels for various components. Specifically, I adjusted the LM2596 to output 3.4V, while the XL4015 was calibrated to deliver 4.8V. Both converters operated within a designated input voltage range of 7.4V-8.4V, sourced from the 2P2S battery configuration.

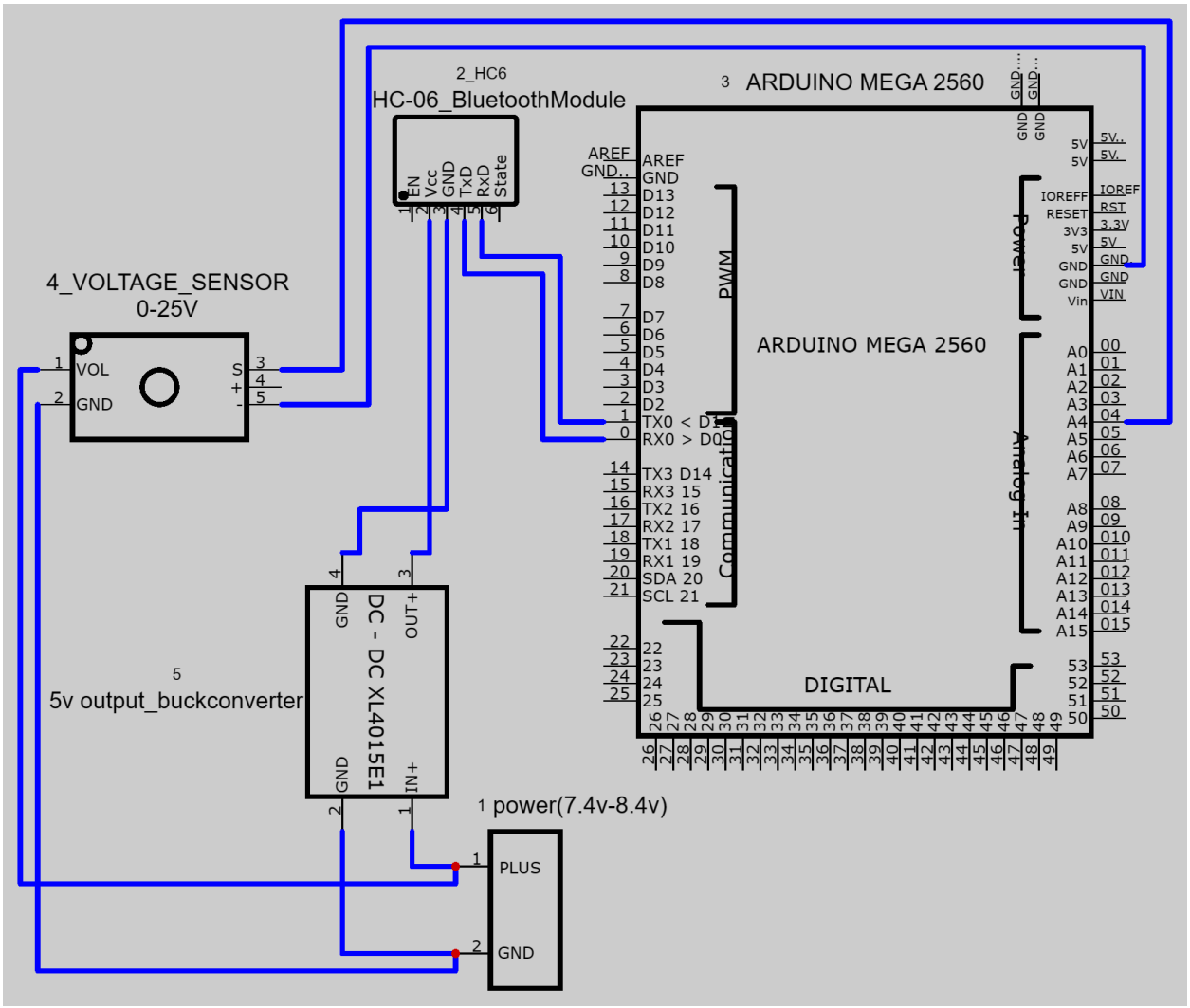

Diagram 4.v.1-Schematic diagram of battery percentage measure

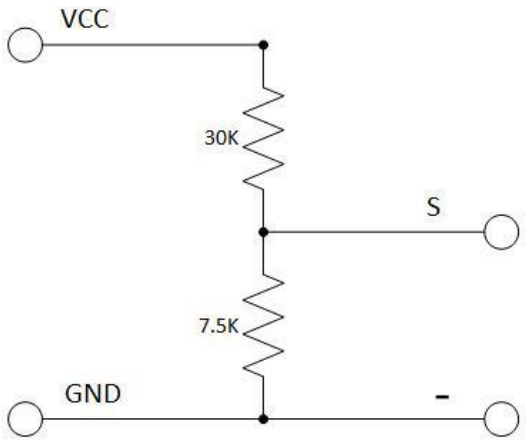

Diagram 4.v.2-Basic theory of voltage sensor

Code for Voltage sensor

```
int in = A4;
unsigned long previousMillis2 = 0;

void loop() {
  unsigned long currentMillis2 = millis();
  if (currentMillis2 - previousMillis2 > 2000) {
    voltage();
    previousMillis2 = currentMillis2;
  }
}

void voltage() {

  float valv = analogRead(in);
  float val2 = (valv / 1024) * 5 ;
  float val3 = (val2*37.5)/7.5
  float avg = ((val3 - 6.5) / 1.9) * 100;

  Serial.println(val3);
  Serial.print(avg);
  Serial.println("%");
}
```

AC DC Converter

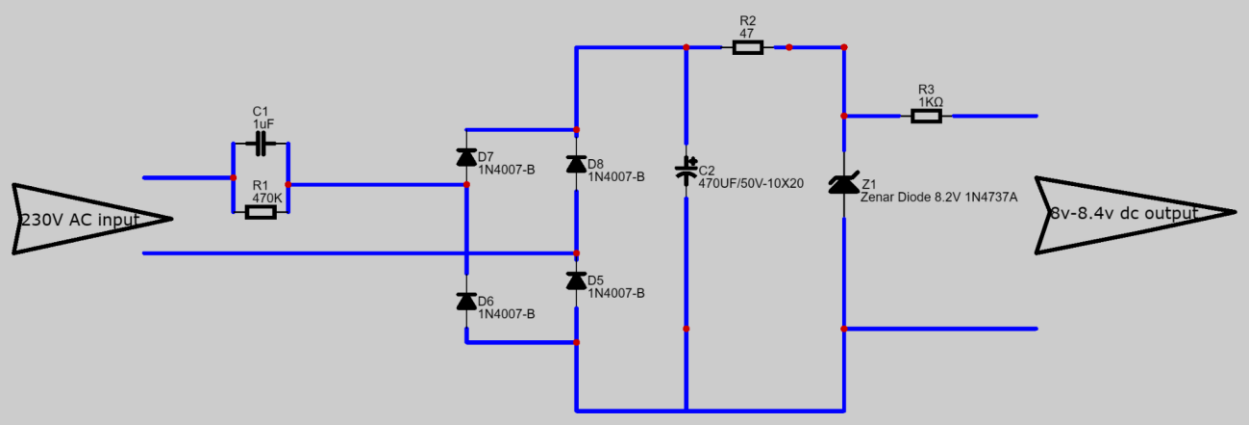

Diagram 4.v.3-Schematic diagram of AC-DC converter

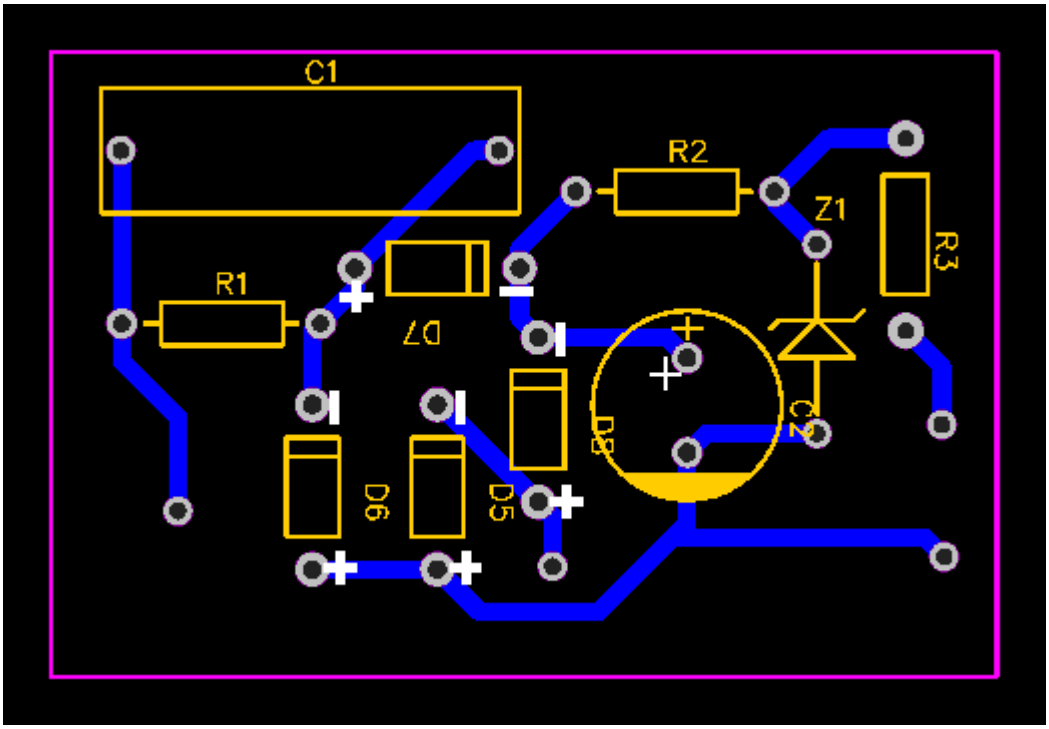

Diagram 4.v.4-pcb diagram of AC-DC Converter

Power Supply for Each Component

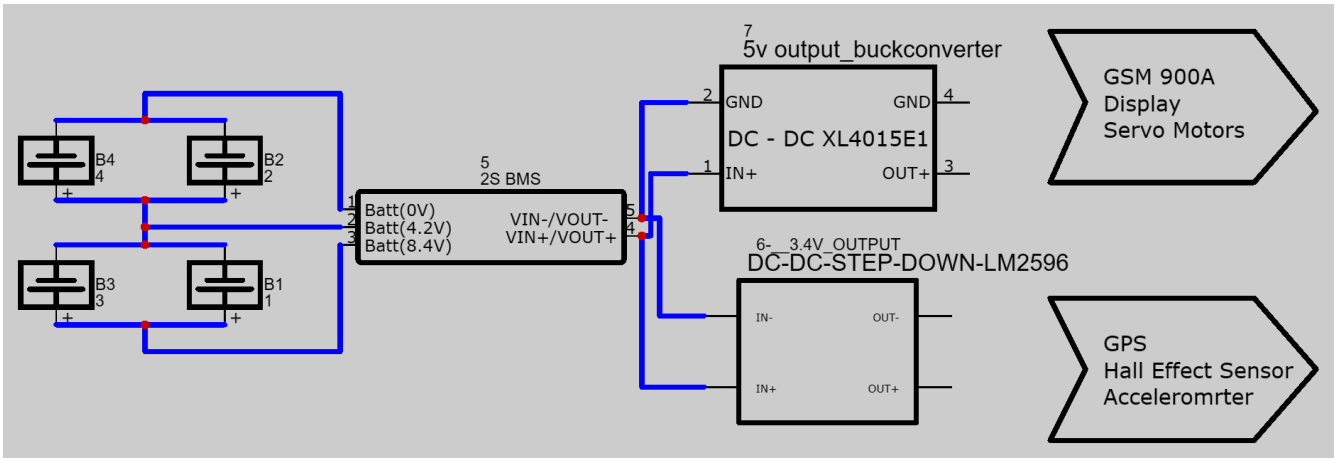

Diagram 4.v.5-Block diagram of the power supply

Appendix B – final code

```
#include <Arduino.h>
#include <LiquidCrystal_I2C.h>
#include <TinyGPS++.h>
#include <Servo.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);  // I2C address 0x27, 16 column and 2 rows


//************ FOR ACCIDENT ******************

const int xInput = A0;
const int yInput = A1;
const int zInput = A2;

int RawMin = 0;
int RawMax = 1023;
int count = 0;
const int sampleSize = 10;
byte updateflag;
byte updateflag1;
byte updateflag2;

//************ for buzzer **************
const int BUTTON_PIN = 11;  // Arduino pin connected to button's pin
const int BUZZER_PIN = 5;   // Arduino pin connected to Buzzer's pin
//*******************************

int xaxis = 0, yaxis = 0, zaxis = 0;
int deltx = 0, delty = 0, deltz = 0;
int vibration = 2, devibrate = 75;
int magnitude = 0;
int sensitivity = 150;
bool buttonPressed = false;  // flag to track if the button has been pressed
bool buzzerOn = false;       // flag to track if the buzzer is currently on
//bool resetLoop = false;    // flag to track if the loop should be reset
bool codeExecuted = false;
boolean impactDetected = false;
//Used to run impact routine every 2mS.
unsigned long time1;
unsigned long impactTime;
unsigned long alertDelay = 30000;  //30 seconds

//***********GPS*************
```

```
TinyGPSPlus gps;  // Create a TinyGPS++ object
double longi;
double lati;
boolean GPS = true;

unsigned long previousMillis1 = 0;

//====================================GSM=================================

String EmergencyContact = "+94766251587";
String Contact1 = "+94766251587";
String Contact2 = "+94707212979";
String url = "";
String phoneNum = "";

//====================================SERVO================================

Servo pcc;
Servo pcc1;
char val;
unsigned long previousMillis4 = 0;
unsigned long previousMillis5 = 0;
boolean pccstate1 = false;
boolean pccstate2 = false;
boolean lockon=false;

//====================================VOLTAGE==============================

int in = A4;
unsigned long previousMillis2 = 0;

//====================================FOR SPEED============================

// Constants
const int hallPin = 2;                    // Hall sensor pin
const float wheelCircumference = 2 * 3.14 * 0.2;  // Wheel circumference in meters (adjust as needed)
const unsigned long interval = 2000;          // Measurement interval in milliseconds

// Variables
volatile unsigned int hallCount = 0;  // Hall sensor count
unsigned long previousMillis = 0;
float speedKmph = 0.0;

void setup() {

  analogReference(EXTERNAL);
  Serial.begin(9600);
```

```
  // Begin serial communication with Arduino and SIM900
  Serial2.begin(9600);

  // Start the software serial port at the GPS's default baud
  Serial3.begin(9600);

  lcd.init();  // initialize the lcd
  lcd.backlight();
  // Attach interrupt to the Hall sensor pin
  attachInterrupt(digitalPinToInterrupt(hallPin), countPulse, RISING);

  pcc.attach(9);   //servo
  pcc1.attach(7);  //servo2
  pcc1.write(30);
  pcc.write(180);
  pinMode(in, INPUT);

  Serial2.println("AT+CMGF=1");  // Change sim900A  Serial to SIM900A
  delay(1000);
  Serial2.println("AT+CNMI=2,2,0,0,0");  // Configure module


  //********* ADXL335 ***************
  xaxis = analogRead(xInput);
  yaxis = analogRead(yInput);
  zaxis = analogRead(zInput);
  //********** For buzzer ***************
  pinMode(BUTTON_PIN, INPUT_PULLUP);  // set Arduino pin to input pull-up mode
  pinMode(BUZZER_PIN, OUTPUT);        // set Arduino pin to output mode
  digitalWrite(BUZZER_PIN, LOW);
  //****************************
  lcd.begin(0x27, 16, 2);
}

void loop() {

  lcd.init();

  unsigned long currentMillis1 = millis();
  if (currentMillis1 - previousMillis1 > 58000) {
    getGPS();
    previousMillis1 = currentMillis1;
  }

  accidentdetected();

  servo();
```

```
  //servo1();


  unsigned long currentMillis2 = millis();
  if (currentMillis2 - previousMillis2 > 2000) {
    voltage();
    previousMillis2 = currentMillis2;
  }

  speed();
  readMessege();
}


void accidentdetected() {


  //call impact routine every 2mS
  if (micros() - time1 > 1999) Impact();  // 2000micro seconds = 2 ms
  fall();

  if (updateflag1 == 1 && updateflag2 == 1 ) {
    updateflag = 1;
  }
  if (updateflag1 == 1 && updateflag2 == 0) {
    updateflag = 0;
  }

  if (updateflag == 1 ) {
    updateflag = 0;
    updateflag2 = 0;
    updateflag1 = 0;
    if(lockon==false){

    turnOnBuzzer();
    Serial.println("Accident Detected");
     Serial.println(magnitude);
      lcd.setCursor(0,1);
      lcd.print("Accident");
      lcd.setCursor(9,1);
      lcd.print(magnitude);

    impactDetected = true;
    impactTime = millis();
    }
  }  //end of updateflag

  if (impactDetected == true) {
```

```
    if ((millis() - impactTime >= alertDelay) && digitalRead(BUZZER_PIN) == HIGH) {
      turnOffBuzzer();
      GPS = false;
      SendAlert(EmergencyContact, "Accident Detected! " + url);
      Serial.println("Sent Alert");
      impactDetected = false;
      impactTime = 0;
    }
  }  //end of impactDetected delay

  if (digitalRead(BUTTON_PIN) == LOW) {
    buttonPressed = true;
  }
  if (buttonPressed == true) {
    turnOffBuzzer();
    Serial.println("");

    buttonPressed = false;
    // impactDetected = false;
  }
}

void Impact() {
  //---------------------------------------------------------------
  time1 = micros();  // resets time value
  //---------------------------------------------------------------
  int oldx = xaxis;  //store previous axis readings for comparison
  int oldy = yaxis;
  int oldz = zaxis;

  xaxis = analogRead(xInput);
  yaxis = analogRead(yInput);
  zaxis = analogRead(zInput);

  //---------------------------------------------------------------
  //loop counter prevents false triggering. Vibration resets if there is an impact. Don't detect new changes until
 that "time" has passed.
  vibration--;
  //Serial.print("Vibration = "); Serial.println(vibration);
  if (vibration < 0) vibration = 0;
  //Serial.println("Vibration Reset!");

  if (vibration > 0) return;
  //---------------------------------------------------------------
  deltx = xaxis - oldx;
  delty = yaxis - oldy;
  deltz = zaxis - oldz;
```

```
  //Magnitude to calculate force of impact.
  magnitude = sqrt(sq(deltx) + sq(delty) + sq(deltz));

  if (magnitude >= sensitivity)  //impact detected
  {
    updateflag1 = 1;
    // reset anti-vibration counter
    vibration = devibrate;
  }

  else {
    //if (magnitude > 15)
    //Serial.println(magnitude);
    //reset magnitude of impact to 0
    magnitude = 0;
  }
}

void fall() {
  // read new state
  // int buttonState = digitalRead(BUTTON_PIN);

  // Read raw values
  int xRaw = ReadAxis(xInput);
  int yRaw = ReadAxis(yInput);
  int zRaw = ReadAxis(zInput);

  // Convert raw values to 'milli-Gs"
  long xScaled = map(xRaw, RawMin, RawMax, -3000, 3000);
  long yScaled = map(yRaw, RawMin, RawMax, -3000, 3000);
  long zScaled = map(zRaw, RawMin, RawMax, -3000, 3000);

  // Re-scale to fractional Gs
  float xAccel = xScaled / 1000.0;
  float yAccel = yScaled / 1000.0;
  float zAccel = zScaled / 1000.0;

  /*Serial.print("X, Y, Z  :: ");
    Serial.print(xRaw);
    lcd.setCursor(0,0);
    lcd.print(xRaw);
    Serial.print(", ");
    Serial.print(yRaw);
    Serial.print(", ");
    lcd.setCursor(5,0);
    lcd.print(yRaw);
    Serial.print(zRaw);
    lcd.setCursor(10,0);
```

```
    lcd.print(zRaw);
    Serial.print(" :: ");
    Serial.print(xAccel,0);
    Serial.print("G, ");
    Serial.print(yAccel,0);
    Serial.print("G, ");
    Serial.print(zAccel,0);
    Serial.println("G");*/

  //if ((xRaw > 470 && xRaw < 550) && (zRaw > 400 && zRaw < 607))
  // if ((xRaw >500 && xRaw <550 )&&(zRaw >410&& zRaw <650 ) )//Accident detected
  //if ((yRaw > 490 && xRaw < 535 ) && ((zRaw > 410 && zRaw < 425 ) || (zRaw > 620 && zRaw < 635) ))
//Accident detected
  if ((yRaw > 500 && yRaw < 565) && ((zRaw > 400 && zRaw < 435) || (zRaw > 600 && zRaw < 650))) {
    if (!codeExecuted) {
      updateflag2 = 1;
      Serial.println("Fall detected");
      codeExecuted = true;
    }
  } else if ((yRaw > 595 && yRaw < 6630) && (zRaw > 470 && zRaw < 550)) {
    codeExecuted = false;
    buttonPressed = false;
  }
}

// Take samples and return the average
int ReadAxis(int axisPin) {
  long reading = 0;
  analogRead(axisPin);
  delay(1);
  for (int i = 0; i < sampleSize; i++) {
    reading += analogRead(axisPin);
  }
  return reading / sampleSize;
}


/============================  Turn on and off buzzer ============================/
void turnOnBuzzer() {
  digitalWrite(BUZZER_PIN, HIGH);  // turn on the buzzer
  buzzerOn = true;
}

void turnOffBuzzer() {
  digitalWrite(BUZZER_PIN, LOW);  // turn off the buzzer
  buzzerOn = false;
}
```

```
/============================  FOR GPS  ======================================/

void getGPS() {
  boolean newData = false;
  for (unsigned long start = millis(); millis() - start < 2000;) {
    while (Serial3.available() > 0 && GPS) {
      //Serial.println("Insode the while loop");
      if (gps.encode(Serial3.read())) {
        newData = true;
        break;
      }
    }  //end of while loop
  }

  if (newData)  //If newData is true
  {
    lati = gps.location.lat();
    longi = gps.location.lng();
    Serial.print("\n\nLatitude: ");
    Serial.println(lati, 6);
    Serial.print("Longitude: ");
    Serial.println(longi, 6);
    newData = false;
  } else {
    Serial.println("No GPS data is available");
  }

  if (gps.location.isValid()) {
    Serial.println("========Valid Location=======");
    Serial.print("Latitude: ");
    Serial.println(lati, 6);
    Serial.print("Longitude: ");
    Serial.println(longi, 6);
    Serial.print("Altitude: ");
    Serial.println(gps.altitude.meters());
    //delay(3000);
    url = "https://www.google.com/maps/search/?api=1&query=" + String(lati, 6) + "," + String(longi, 6);

  } else {
    Serial.println("Location: Not Available");
    url = " Location: Not Available";
    //delay(3000);
  }
  //readMessege();

  // If 5000 milliseconds pass and there are no characters coming in
  // over the software serial port, show a "No GPS detected" error
  if (millis() > 5000 && gps.charsProcessed() < 10) {
```

```
    Serial.println("No GPS detected");
    url = " Location: Not Available";
    return;
   }
  Serial2.println("AT+CMGF=1");  // Change sim900A  Serial to SIM900A
  delay(1000);
  Serial2.println("AT+CNMI=2,2,0,0,0");  // Configure module

}

/============================  FOR SEND LOCATION  ================================/


void readMessege() {
  if (Serial2.available()) {
    String response = Serial2.readString();
    Serial.println(response);
    //remove "/n"
    String message = response;
    message.replace("\n", "");
    String ATcommand = message.substring(0, response.indexOf(':') - 1);
    ATcommand.trim();  // Trim white spaces

    if (ATcommand == "+CMT") {
      // Find the index of the first double quote
      int firstDoubleQuoteIndex = message.indexOf('\"');
      // Extract the phone number
      phoneNum = message.substring(firstDoubleQuoteIndex + 1, message.indexOf('\"', firstDoubleQuoteIndex + 1));
      int lastDoubleQuoteIndex = message.lastIndexOf('\"');
      // Extract the message
      String messageContent = message.substring(lastDoubleQuoteIndex + 1);
      messageContent.trim();
      Serial.println("Message: " + messageContent);
      Serial.println("phone number: " + phoneNum);
      //send the location
      if (phoneNum == Contact1 || phoneNum == Contact2) {  //can add more phone numbers here
        if (messageContent.equalsIgnoreCase("Location")) {
          //FoundLocation = false;
          //getGPS(phoneNum);
          GPS = false;
          SendAlert(phoneNum, url);
        }
      }
    }  //end of if ATcommand
  }
}

/============================  FOR SEND ALERT  ================================/
```

```
void SendAlert(String phone, String link) {

  Serial.println("Initializing...");
  delay(1000);

  String message = link;

  Serial2.println("AT");  // Handshaking with SIM900
  updateSerial();

  Serial2.println("AT+CMGF=1");  // Configuring TEXT mode
  updateSerial();

  Serial2.print("AT+CMGS=\"" + phone + "\"\r");
  updateSerial();

  Serial2.print(message);  // Text content
  updateSerial();
  Serial2.write(26);
  GPS = true;
}


void updateSerial() {
  delay(500);
  while (Serial.available()) {
    Serial2.write(Serial.read());  // Forward what Serial received to Software Serial Port
  }
  while (Serial2.available()) {
    Serial.write(Serial2.read());  // Forward what Software Serial received to Serial Port
  }
}


//================================for
servo================================================

void servo() {
  if (Serial.available()) {
    val = Serial.read();
    Serial.println(val);

    if (val == '1') {
      pcc.write(0);
      pccstate1 = true;
      previousMillis4 = millis();
       // Store the current time
```

```
      // pinMode(15, LOW);

    } else if (val == '0') {
      pcc1.write(30);
      pccstate2 = true;
      previousMillis5 = millis();  // Store the current time

      // pinMode(15, LOW);
    }
  }

  // Check the delay for servo1
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis4 > 2000 && pccstate1) {
    servo1();
    pccstate1 = false;
    lockon=true; ///lock eka wahila

  }

  // Check the delay for servo2
  if (currentMillis - previousMillis5 > 2000 && pccstate2) {
    servo2();
    pccstate2 = false;
    lockon=false;
  }
}

void servo1() {
  pcc1.write(90);
}

void servo2() {
  pcc.write(180);
}



//================================== for voltage ============================//

void voltage() {

  float valv = analogRead(in);
  float val2 = (valv / 1024) * 25 * 0.35;
  float avg = ((val2 - 6.5) / 1.5) * 100;

  Serial.println(valv);
```

```
  Serial.println(val2);
  Serial.println(avg);


  //delay(1000);
}

//================================== FOR SPEED ==================================

void speed() {
  lcd.clear();  // clear display
  lcd.setCursor(0, 0);         // move cursor to   (0, 0)
  //lcd.print("Arduino");       // print message at (0, 0)
  //delay(500);                 // display the above for two seconds
  unsigned long currentMillis = millis();

  // Calculate the elapsed time
  unsigned long elapsedTime = currentMillis - previousMillis;

  // Check if the measurement interval has passed
  if (elapsedTime >= interval) {
    // Calculate speed in km/h
    float speedMps = (float)hallCount * wheelCircumference / elapsedTime;  // Speed in meters per second
    speedKmph = speedMps * 3600;                                           // Speed in km/h

    // Display the speed
    Serial.print("Speed: ");
    Serial.print(speedKmph);
    Serial.println(" km/h");

    // Reset the hall sensor count and previous time
    hallCount = 0;
    previousMillis = currentMillis;
  }
  lcd.clear();
  lcd.print("speed:");
  lcd.print(speedKmph);
  lcd.print("kmph");
}
// Interrupt service routine to count pulses from the Hall sensor
void countPulse() {
  hallCount++;
}
```