ABSTRACT

Gps Waypoint Navigated (GWN) vehicles are complicated and interesting work which develops competence in navigation planning. We had been working to navigate the vehicle from its current position to user defined destination automatically by avoiding obstacles. In the present prototype, we have worked with a land based rover, which has the exact same number of actuators. During the project, we have worked on challenges from GPS, because the resolution available in India is comparatively less due to restrictions put in place due to security reasons. It has an accuracy of 10 meters. We had calculated the bearing and heading from the GPS and digital compass. Getting heading from the compass possessed some difficulty because of magnetic influence from the vehicle chassis and motor flux. If we try to implement the model directly in a water-borne vehicle that would make the execution more complicated. So we have tried a model which has same working procedure as our project is considered. With this trial model we are trying to implement the desired functionality. After having the changes, if the requirement are satisfied then we can implement it directly to the main system and make it run.

ACKNOWLEDGEMENT

It is our proud privilege to epitomize our deepest sense of gratitude and indebtedness to our advisor, **Prof. Saroj Kumar Padhy** for his valuable guidance, keen and sustained interest, intuitive ideas and persistent endeavour. His inspiring assistance, laconic reciprocation and affectionate care enabled us to complete our work smoothly and successfully.

We extend our sincere thanks to **Prof. Santosh Kumar Panda, Department Project Coordinator and Prof. Rajesh Kumar Dash, B.Tech. Project Coordinator** for giving us the opportunity and motivating us to complete the project within stipulated period of time and providing a helping environment.

We would like to thank **Dr. Sukant K. Mohapatra(Chairman, NIST), Dr. Priyadarshi Tripathy (Principal, NIST) and Dr. Souren Misra (HoD, Department of Mechanical Engg.)** for having been the ultimate source of inspiration and moral support.

We would also like to extend our heartfelt gratitude to our parents and friends for their unflinching support and help.

H. DHARMENDRA GUPTA
ROLL NO. ME201812544

BORA PRATHAMESH
ROLL NO. ME201812047

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Navigation it is the process or activity of accurately ascertaining one's position and planning and following a route. An autonomous underwater vehicles (AUVs) is a robot that travels underwater without requiring input from an operator. Underwater robotics is making exploration cheaper and more accessible. Scientists can reach spots of the ocean that were previously untouchable. With their ability to withstand harsh environments.

There's no doubt the future of ocean exploration starts with underwater robotics. Technological developments and trends in sensors, processors and actuators for autonomous underwater vehicles (AUVs) have fostered new potential applications. This type of underwater vehicles has recently become an attractive alternative for underwater search and exploration since they are cheaper than manned vehicles. Over the past years, there have been abundant attempts to develop underwater vehicles to meet the challenge of exploration and extraction programs in the oceans. Recently, researchers have focused on the development of AUVs for long-term data collection in oceanography and coastal management.

CHAPTER 2
LITERTURE REVIEW

Autonomous underwater vehicles (AUVs) are considered as a substantial group of submerged systems known as “unmanned underwater vehicles (UUVs)”. UUVs are generally classified as AUV and remotely operated vehicle (ROV). ROVs are powered and operated from a surface control station by an umbilical cord or remote control. AUVs carry their independent onboard power supply. An AUV is a highly nonlinear robotic vessel, whose dynamic equation include square terms due to hydrodynamic damping factors. It can operate both above and beneath the ocean′s surface. The AUV propagates by changing its buoyancy in small steps, thereby converting the resultant vertical displacement to horizontal movement. This is accomplished by the interactivity between the surface control station and the water column. planning is finding the course of points across which AUV has to travel to reach the predefined destination from the starting location, whereas the time history of this journey of the AUV is referred to as trajectory planning. AUV navigation is a very important aspect of path planning. No external communication or “global positioning system (GPS)” signals are available in underwater environments. Thus, without information of direction and restricted power, it is very difficult for an AUV to navigate towards the desired target. Referring to the literature available on AUV navigation, one can distinguish three different problems that are “close- to-surface navigation, navigation in the mid-depth zone, close-to-bottom navigation”. In the path planning control (PPC) problem, an AUV has to traverse a convergent path without temporal constraints. Earlier works on PPC of wheeled robots solved two major issues reported as “path parameterization” and the selection of the termination point on the path. [2]

Time-coordinated path following (TC-PF)

A coordinated path-planning problem for a common group of under-actuated AUVs has been solved using “Lyapunov theory”. The proposed system is stable and is able to deal with the problem of temporary communication failures. The “time coordinated path following (TC-PF)” is a structure suggested to investigate the problem in

controlling multiple AUVs. It enables cooperative path planning subjected to space, time, and energy restrictions. Parallel formation designing and synchronization are constrained by two factors. The first factor is layout of communication network and the second factor is AUV.

Cooperative path planning in unpredictable environments

It has been proved by a number of researchers that the major issues of the cooperative path planning for multiple AUVs in the unpredictable underwater environment can be decomposed in three phases.[3] In the first phase, it has been considered that the cooperative path planning algorithm for multiple-AUVs are very hard to design using the existing methods due to the exponential increase in computation time with increase in the number of AUVs. Thus, new algorithms have to be designed to reduce computational complexity. Secondly, the calculation of cost in terms of time between two positions is quite troublesome as the ocean flows are typically continuous and time-varying vector fields. Finally, an evaluation function is required to approximate overall performance. Consequently, various methods and algorithms have been developed by researchers for solving the above mentioned problems. Here an attempt is made to review the available literature in this field to the best of our knowledge.[2]

Summary:

This survey presents a qualitative analysis of the impact of the marine environment on the path planning of AUVs. The underwater environment is characterized as predictable and unpredictable depending on path planning approximations. This paper summarizes the avail- able path planning algorithms employed for single and multiple AUVs with reference to predictable and unpredictable behavioral models of the environment. The issues involved in path planning of AUVs are discussed briefly. Merits and demerits of every method have been discussed briefly. Path costs are compared as low, moderate and high. Collision and obstacle avoidance are discussed as achieved, limited and poor based on whether the algorithm focused on these issues or not. Based on this study, we can conclude that the issues of unreliability have not been addressed much in the studied literature. Many assumptions are taken for AUV dynamics and operating environment, which are required to be critically analyzed for stability in real world scenarios. Thus, there is a need for formulating optimized

algorithms in the future that is computationally efficient and rugged for real time applications of AUVs.

CHAPTER 3
ARDUINO INTERFACING WITH DIFFERENT COMPONENTS

3.1 Digital Compass (HMC5883L)

The compass module consists of an HMC5883L three axis(x, y and z) magnetic field chip which consists of 3 magneto-resistive sensors arranged in axes perpendicular to each other. These type of sensors work by magneto-resistance property of a material. The ability to change its value of electrical resistance with respect to the applied external magnetic field in the axis directions. It can be also used to measure the magnetic field.[4]

Table 3.1 HMC5883L Module & Arduino Pin Connection

Arduino	**HMC5883L**
3.3V	VCC
GND	GND
Analog Input A4	SDA (Serial Data)
Analog Input A5	SCL (Serial Clock)

Note: DRDY is an optional connection so it is not used in the circuit.

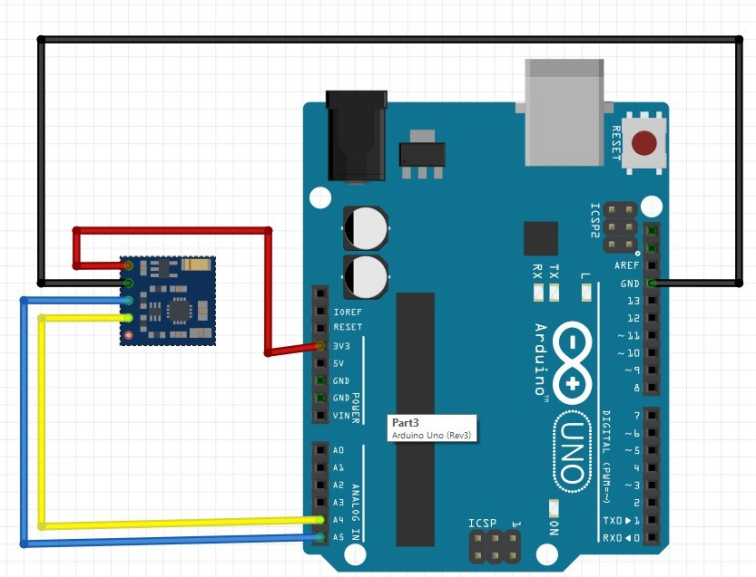

Figure 3.1 Wiring Diagram of Arduino & Digital Compass (HMC5883L)

3.2 Ultra Sonic Sensor (HC-SR04)

It can measure distance. It emits at 40,000 Hz which travels through the air and if there is an object or obstacle on its path it will rebounce back to the module. Considering the travel time and the speed of the sound you can calculate distance. Supply voltage of VCC is +5V. [5]

Formula:-

duration = pulseIn(echoPin, HIGH); //Reads the echopin, returns the sound wave travel time in microseconds.

distance = duration * 0.034 / 2; // Calculating the distance

Serial.print("Distance: "); //Displays the distance on Serial monitor

Serial.print(distance);

Table 3.2 HC-SR04 Module & Arduino Pin Connection

ARDUINO	HC-SR04
5V	VCC
GND	GND
Analog Input A0	ECHO
Analog Input A1	TRIG

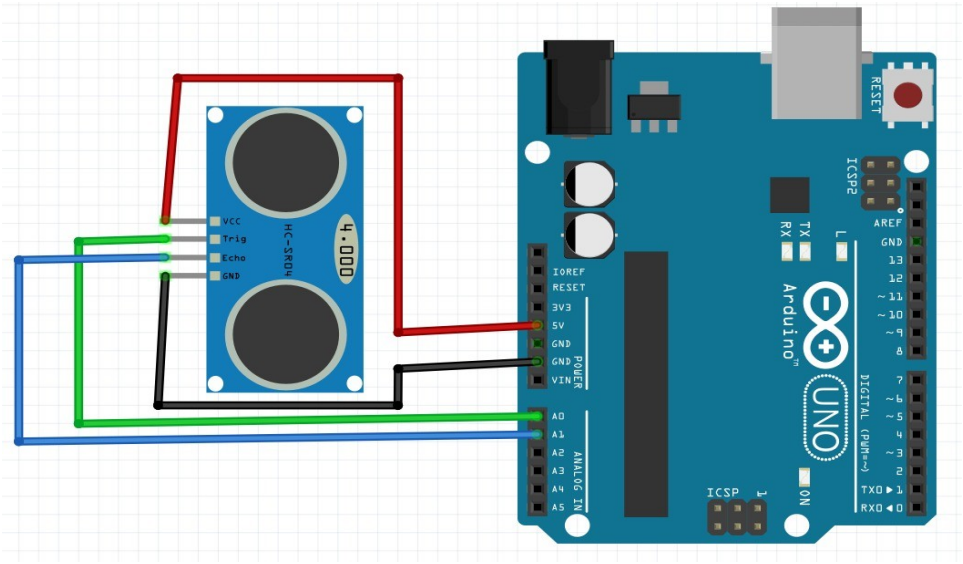

Figure 3.2 Wiring Diagram of Arduino & Ultrasonic Sensor (HC-SR04)

3.3 SIM808 GPS/GSM Modem

SIM808 GPS/GPRS/GSM Modem is an integrated quad-band GSM/GPRS and GPS navigation technology Arduino expansion shields. A credit card size only, according to the standard Arduino pin packaging, compatible with Arduino UNO, arduino Leonardo, arduino Mega and other arduino main board. Compared to the previous generation SIM908, SIM808 made some improvement on the performance and stability. In addition to the normal SMS and phone functions, the shield also supports MMS, DTMF, FTP and other functions. You can achieve the data acquisition, wireless data transceiver, IoT application and GPS orientating. The should integrates onboard microphone and headphone jack, saving your cost and making your project easily. it can also directly connect to the GSM and GPS antenna by an external antenna connector. [6]

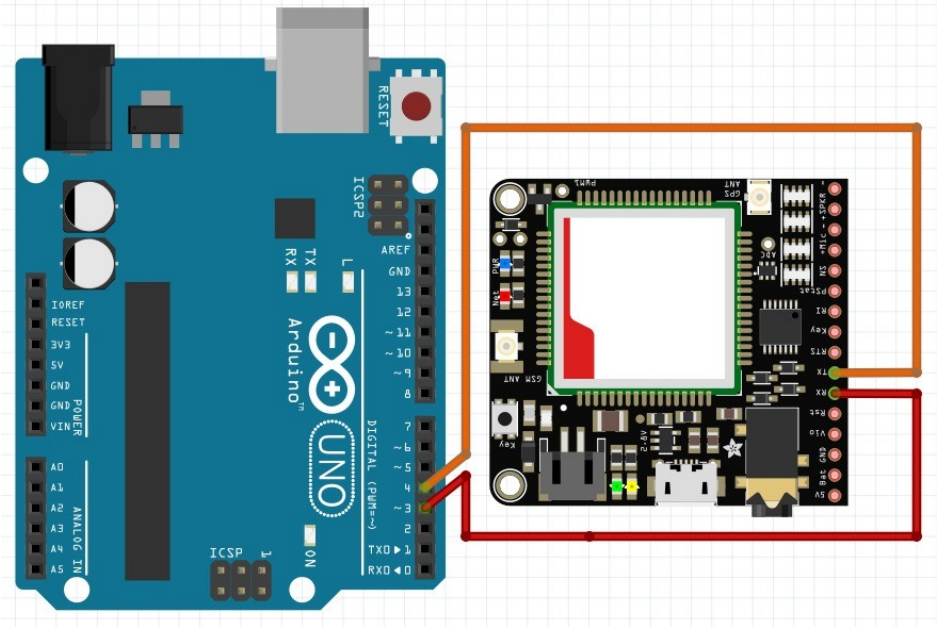

Figure 3.3 Wiring Diagram of Arduino & SIM808 (GPS)

Table 3.3 SIM808 Module & Arduino Pin Connection

ARDUINO	SIM808
12V	VCC
GND	GND
Digital Input D4	TX
Digital Input D3	RX

3.4 Battery

Battery act as a power supply is an important issue in a robot.

3.5 Motor Driver

Motor drivers acts as an interface between the motors and the control circuits. Motor require high amount of current whereas the controller circuit works on low current signals. So the function of motor drivers is to take a low-current control signal and then turn it into a higher-current signal that can drive a motor.

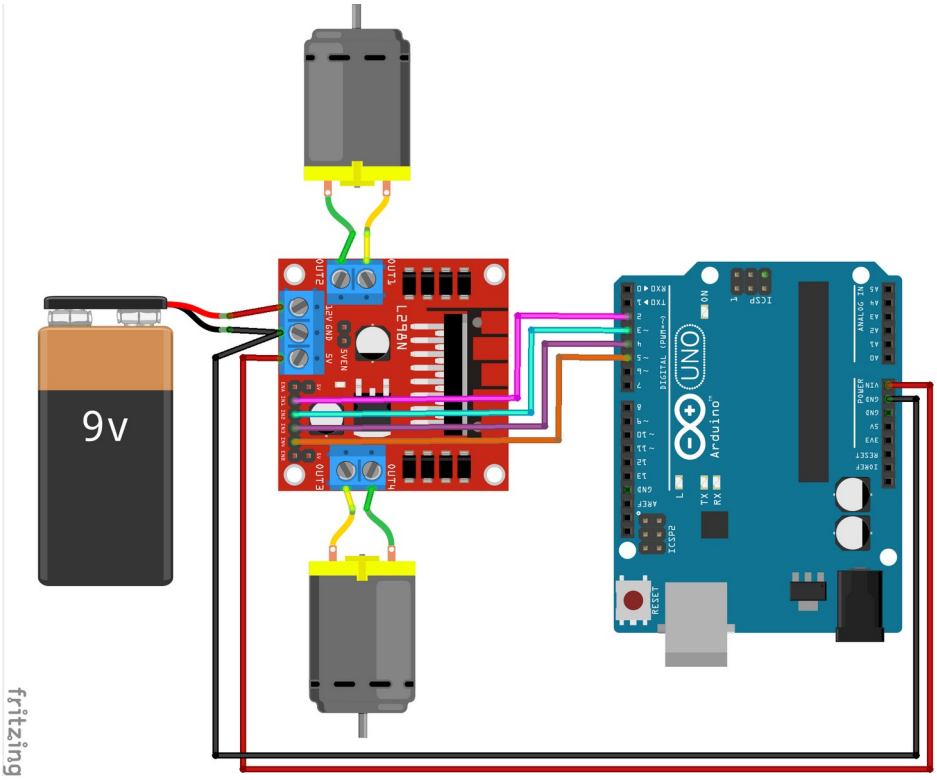

Figure 3.4 L239 Motor Driver

CHAPTER 4
ALGORITHM EXPLANATION

4.1 Arduino Interfacing With Ultrasonic Sensor

- **(**From **Fig 3.2 and Appendix A)** We will be inserting the code in arduino software and as well as will connecting the arduino UNO and ultrasonic sensor.
- Make sure the connection is completely and secured and perfectly connected to each other.
- We will compile to make sure there is no error it code and upload it to arduino.
- One of the eye will send the signal and the other will receive the signal. The formula mentioned above will calculated and the vehicle move forward until the required distance limit defined by the user.
- After reaching the limit the vehicle turn left or right and recalculate the distance. According to the result the vehicle will move forward.

4.2 Arduino Interfacing With Digital Compass (HMC5883L)

- **(**From **Fig 3.1 and Appendix C)** We will be inserting the above code in arduino software and as well as will connecting the arduino UNO and digital compass.
- Make sure the connection is completely and secured and perfectly connected to each other.
- We will compile to make sure there is no error it code and upload it to arduino.
- The result we will be getting the coordinate of x, y and z or we can also can get the specified direction i.e. "N OR E OR S OR W OR NE OR NW OR SE OR SE" by mentioning the angle also.

4.3 Arduino Interfacing With SIM808

- **(**From **Fig 3.3 and Appendix B)** We will be inserting the above code in arduino software and as well as will connecting the arduino UNO and digital compass.
- Make sure the connection is completely and secured and perfectly connected to each other.
- We will compile to make sure there is no error it code and upload it to arduino.
- As a result, it will be printing the co-ordinates of a particular point from the antenna i.e. (latitude, longitude).

4.4 Overall Code Explanation

- **(From APPENDIX D)**The user will define the pins to connect with motor driver, sim808, HMC5883l, and Ultrasonic Sensor.
- The above overall code will be written and uploaded to the arduino.
- There's a formula used to calculate the (A bearing is the direction from one place to another, measured in degrees of angle with respect to an accepted reference line)and (Heading is the direction the aircraft is pointing. The aircraft may be drifting a little or a lot due to a crosswind. Bearing is the angle in degrees (clockwise) between North and the direction to the destination). We have not used LORA module.

4.5 Architecture

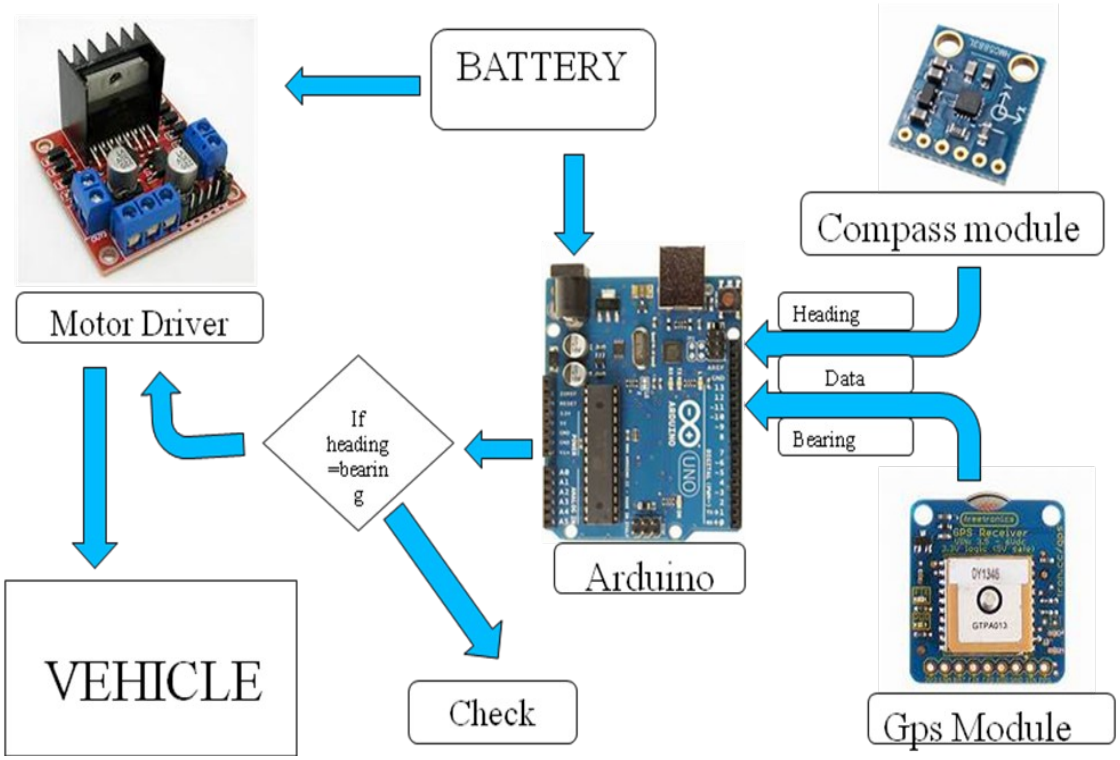

Figure 4.1 Mechanism of Navigation of Automated Vehicle

- Main source or brain of the vehicle is Arduino which is a prototyping board based on ATMEL ATmega microcontroller .
- Arduino collects the data from the GPS as Longitude and Latitude and from Compass as Direction. After receiving the data from both it will convert them into readable values (heading and bearing) and calculate the distance, store it in x and y variables.
- The Arduino will compare the heading and bearing values. This comparison result will make the decision to move forward or to check the left or right to get accurate values which satisfy the forward condition.
- **Fig 4.2** shows the schematic diagram of our model.
- **Fig 4.3 &Fig 4.4** shows the front view and top view of our model.
- **Figure 4.5** & **Figure 4.6** deals with the real chassis of underwater model.

4.6 Schematic Diagram and Physical Model

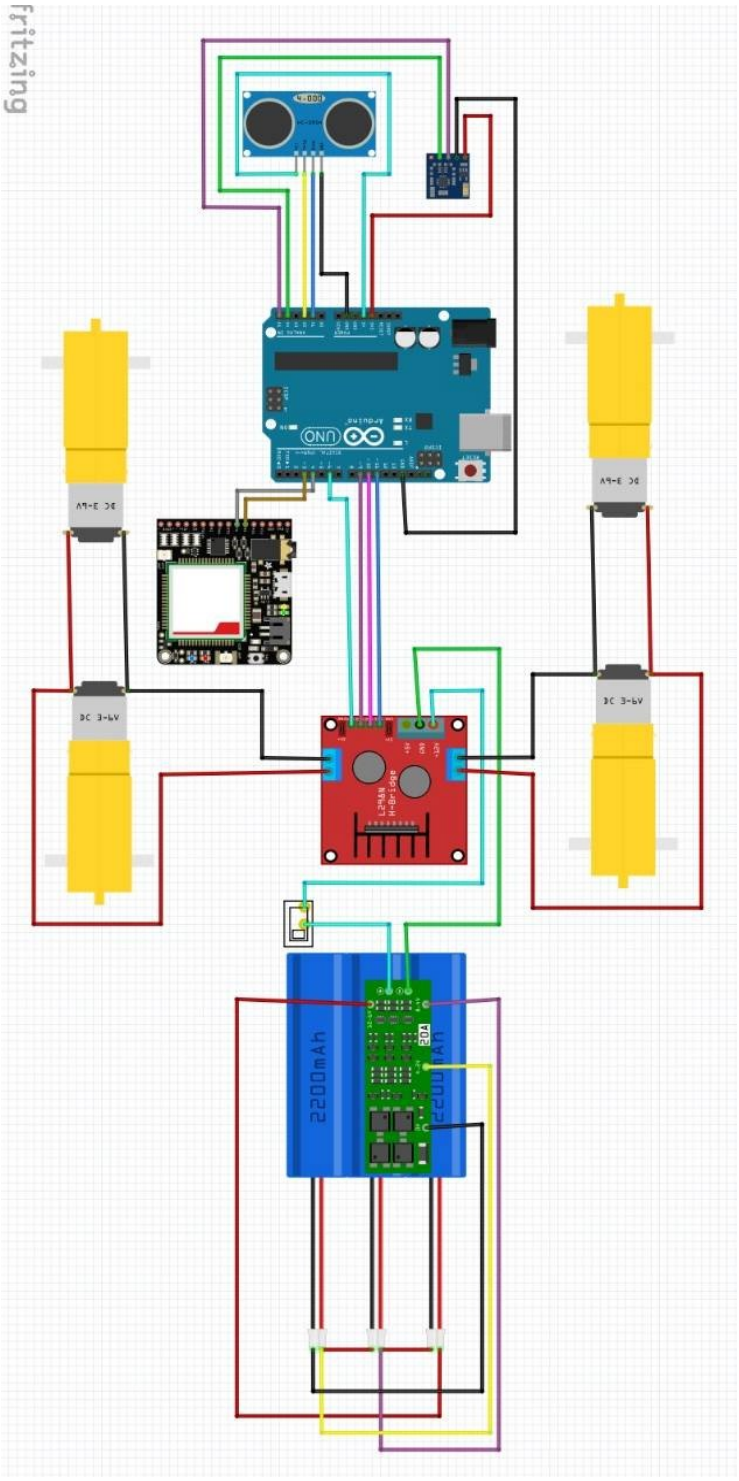

Figure 4.2 Schematic Diagram of Vehicle

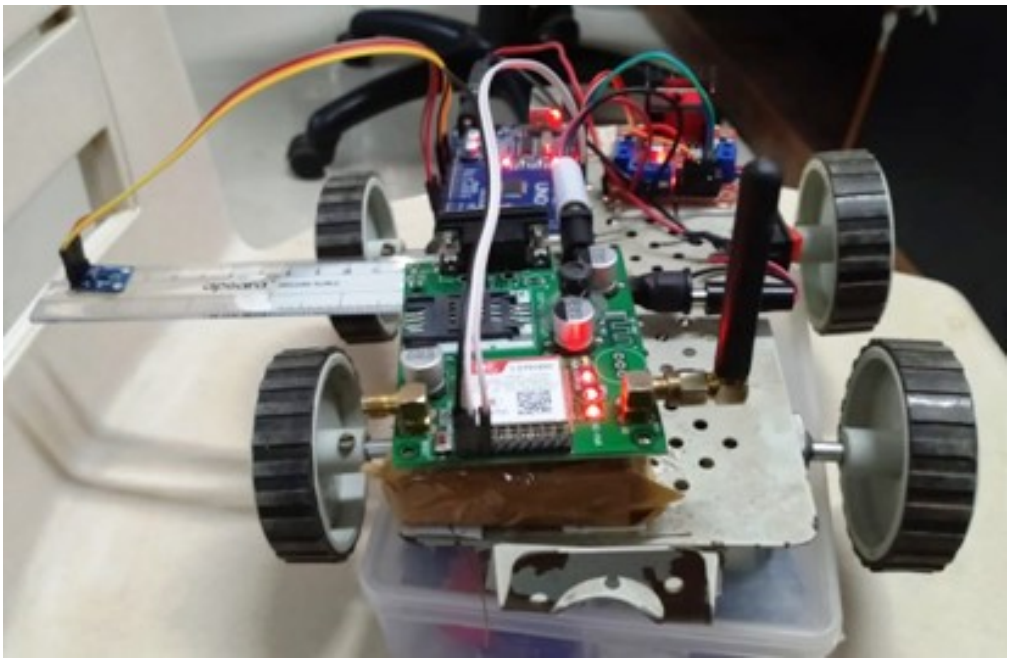

Figure 4.3 Front View

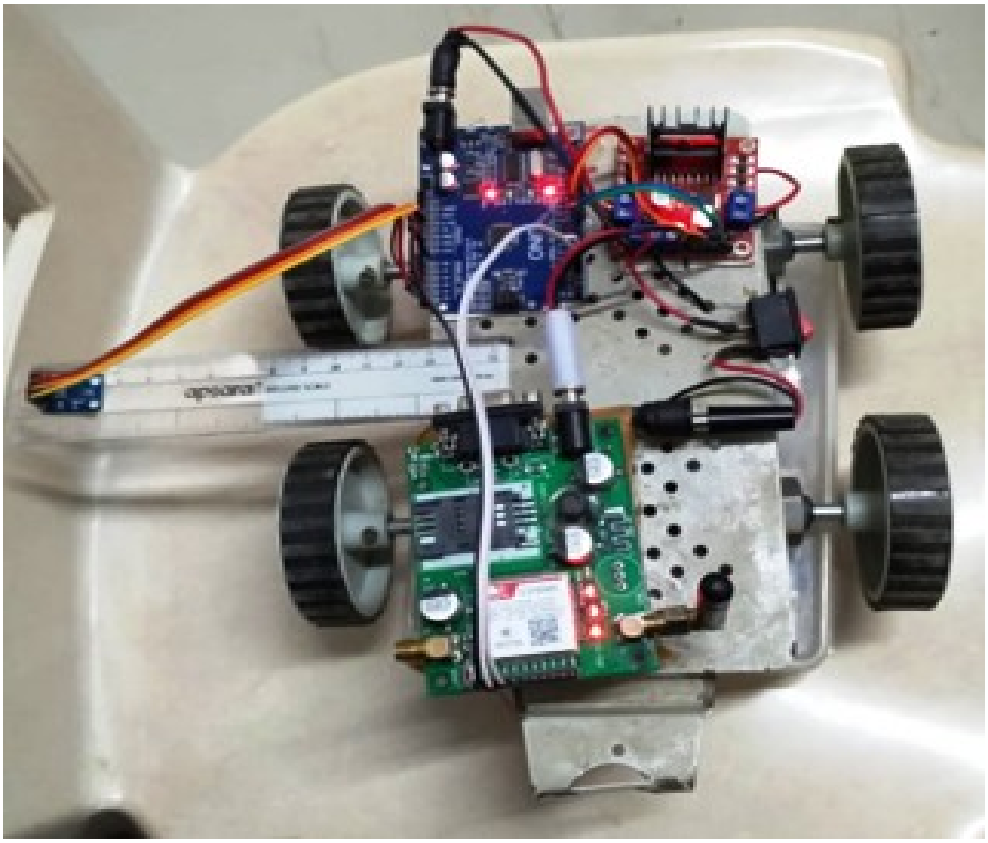

Figure 4.4 Top View

Figure 4.5 Real Model Front View

Figure 4.6 Real Model Top View

CHAPTER 5
CONCLUSION AND FUTURE SCOPE

5.1 Conclusion

There are new things introduced in our project. We have researched about it in official site of arduino and you tube videos done by them. Challenges always pops up in every type of work. In this project we have also faced the challenge like getting accurate compass direction, make the vehicle to go forward to the user defined location. Also we needed to control the ratio of heading and bearing for forwarding the vehicle.

5.2 Future Scope and Its Implementation

- It can be implemented in agriculture sectors to help farmers in cultivating process.
- Transportation service can be applicable inside the limited area .
- Also we can use Sim-Card services for instant control of the vehicle.
- We can use it for tour inside a specified coordinated location (such as different Universities for students as well as parents and guests also , tourists spots in different places across the world, etc).
- It can be used defense field like bomb plant or armed submarines.

REFERENCES

APPENDIX A

ARDUINO AND ULTRASONIC SENSOR

```
int Echo = A1;  // Echo(P2.0)
int Trig =A0;  //  Trig (P2.1)
int Front_Distance = 0;//
int Left_Distance = 0;
int Right_Distance = 0;
int Left_motor_go=8;     //(IN1)
int Left_motor_back=9;     //(IN2)
int Right_motor_go=10;    // (IN3)
int Right_motor_back=11;    // (IN4)
int distance;
void setup()
{
  Serial.begin(9600);
  pinMode(Left_motor_go,OUTPUT); // PIN 8 (PWM)
  pinMode(Left_motor_back,OUTPUT); // PIN 9 (PWM)
  pinMode(Right_motor_go,OUTPUT);// PIN 10 (PWM)
  pinMode(Right_motor_back,OUTPUT);// PIN 11 (PWM)
  pinMode(Echo, INPUT);    //
  pinMode(Trig, OUTPUT);   //
}
void run()     //
{
  digitalWrite(Right_motor_go,HIGH);  //
  digitalWrite(Right_motor_back,LOW);
  analogWrite(Right_motor_go,200);//PWM0~255
  analogWrite(Right_motor_back,0);
  digitalWrite(Left_motor_go,LOW);  //
  digitalWrite(Left_motor_back,HIGH);
  analogWrite(Left_motor_go,0);//PWM0~255
  analogWrite(Left_motor_back,200);
     //
}
void brake()  //
{
```

```
  digitalWrite(Right_motor_go,LOW);
  digitalWrite(Right_motor_back,LOW);
  digitalWrite(Left_motor_go,LOW);
  digitalWrite(Left_motor_back,LOW);
  delay(1000);//
}
void left()         //()
//void left()         //()
{
  digitalWrite(Right_motor_go,HIGH);  //
  digitalWrite(Right_motor_back,LOW);
  analogWrite(Right_motor_go,180);
  analogWrite(Right_motor_back,0);//PWM0~255
  digitalWrite(Left_motor_go,HIGH);   //
  digitalWrite(Left_motor_back,LOW);
  analogWrite(Left_motor_go,180);
  analogWrite(Left_motor_back,0);//PWM0~255
  delay(500);  //
}
void right()
//void right()        //()
{
  digitalWrite(Right_motor_go,LOW);   //
  digitalWrite(Right_motor_back,HIGH);
  analogWrite(Right_motor_go,0);
  analogWrite(Right_motor_back,200);//PWM0~255
  digitalWrite(Left_motor_go,LOW);//
  digitalWrite(Left_motor_back,HIGH);
  analogWrite(Left_motor_go,0);
  analogWrite(Left_motor_back,200);//PWM0~255
  delay(500);  //
}
void back()         //
{
   digitalWrite(Right_motor_go,LOW);  //
  digitalWrite(Right_motor_back,HIGH);
  analogWrite(Right_motor_go,0);
  analogWrite(Right_motor_back,160);//PWM0~255
  digitalWrite(Left_motor_go,HIGH);  //
```

```
  digitalWrite(Left_motor_back,LOW);
  analogWrite(Left_motor_go,150);
  analogWrite(Left_motor_back,0);//PWM0~255
  delay(500);     //
}
float Distance_test()   //
{
  digitalWrite(Trig, LOW);   // 2s
  delayMicroseconds(2);
  digitalWrite(Trig, HIGH);  // 10s10s
  delayMicroseconds(10);
  digitalWrite(Trig, LOW);    //
  float Fdistance = pulseIn(Echo, HIGH);  // ()
  Fdistance= Fdistance/58;       //58  Y=X*344/2
  // X= 2*Y/344 ==X=0.0058*Y ===/58
  //Serial.print("Distance:");      //
  //Serial.println(Fdistance);         //
  distance = Fdistance;
  return Fdistance;
}
void loop()
{
    Distance_test();
    if(distance<=20)
 {
  brake();
  delay(100);
  back();
  delay(200);
  brake();
  delay(200);
  left();
  delay(100);
  brake();
  delay(200);
  Distance_test();
  if(distance<=20)
  {
    back();
```

```
   delay(200);
   brake();
   delay(50);
   right();
   delay(50);
   brake();
   delay(50);
   right();
   Distance_test();
   if(distance<=20)
  {
   back();
   delay(200);
   brake();
   delay(50);
   right();
   delay(50);
   brake();
   delay(50);
   run();
  }else
  {
     run();
  }
  }else
  {
     run();
  }
 }else
 {
  run();
 }
 Distance_test();
}
```

APPENDIX B

ARDUINO AND SIM808 GPS

```
#include <SoftwareSerial.h>
#define WAYPOINT_THRESHHOLD 5
SoftwareSerial gps(3, 2);  //---RX=D3 and TX=D2
String lat1="19.199725", lon1="84.745451";
String lat2, lon2;
int LED_PIN = 10;
String data[5];
int i = 0;
char c;
void setup() {
  Serial.begin(9600);
  gps.begin(9600);
  Serial.println("wait for 5 sec");
  delay(5000);
  gps.println("AT+CGNSPWR=1");
  delay(2000);
  gps.println("AT+CGNSSEQ=\"RMC\"");
  delay(2000);
  pinMode(LED_PIN, OUTPUT);
}
void loop() {
    int i=0;
    lon2 = "";
    lat2= "";
      for(i=0; i<5; i++)
        data[i]="";
    i=0;
    Serial.println("----Command given----");
```

```
    gps.flush();
    long int time = millis();
    gps.println("AT+CGNSINF");
    while ((time+1000 ) > millis()){
    while(gps.available()){
       //Serial.println("test point 33");
       c = gps.read();
       if (c != ',') {
         data[i] +=c
         delay(20);
        }
        else
        i++;
       }
       if(i==5)
       break;
    }
    lat2 = data[3];
    lon2 = data[4];
    Serial.println(lat2);
    Serial.println(lon2);
    Serial.println("----Response Print-1--");
    delay(1000);
    distance(lat1.toFloat(),lon1.toFloat(),lat2.toFloat(),lon2.toFloat());
}
 double distance(double lat1, double lon1, double lat2, double lon2)
  {
    // Conversion factor from degrees to radians (pi/180)
      const double toRadian = 0.01745329251;
    // First coordinate (Radians)
    double lat1_r = lat1 * toRadian;
    double lon1_r = lon1 * toRadian;
    // Second coordinate (Radians)
    double lat2_r = lat2 * toRadian;
    double lon2_r = lon2 * toRadian;
    // Delta coordinates
    double deltaLat_r = (lat2 - lat1) * toRadian;
    double deltaLon_r = (lon2 - lon1) * toRadian;
    // Distance
```

```
   double a = sin(deltaLat_r/2)*sin(deltaLat_r/2) + cos(lat1_r) * cos(lat2_r) *
sin(deltaLon_r/2) * sin(deltaLon_r/2);
   double c = 2 * atan2(sqrt(a), sqrt(1-a));
   double distance = 6371 * c * 1000;
   Serial.println("distance to target:");
   Serial.println(distance);
   return distance;
 if(distance<5)
   {
      digitalWrite(LED_PIN, HIGH);
   }
     else
       digitalWrite(LED_PIN, LOW);
 }
```

APPENDIX C

ARDUINO INTERFACING WITH HMC5883L

```
#include <QMC5883LCompass.h>
QMC5883LCompass compass;
void setup() {
  Serial.begin(9600);
  compass.init();
}
void loop() {
  int x, y, z;
  // Read compass values
  compass.read();
  // Return XYZ readings
  x = compass.getX();
  y = compass.getY();
  z = compass.getZ();
  Serial.print("X: ");
  Serial.print(x);
  Serial.print(" Y: ");
  Serial.print(y);
  Serial.print(" Z: ");
  Serial.print(z);
  Serial.println();
  delay(250);
}
```

APPENDIX D

OVERALL CODE (ARDUINO, SIM808, HMC5883L)

```
#include <SoftwareSerial.h>
#include <MechaQMC5883.h>
#include <Wire.h>
#include<Math.h>
#define PI 3.1415
MechaQMC5883 qmc;
          //for GPS
SoftwareSerial gps(4, 3);  //---RX=D4 and TX=D5
String latd="19.199728";
String logd="84.745465";
String latc, logc;
         //for ULTRASONIC
int Echo = A1;                  // Echo(P2.0)
int Trig =A0;                  //  Trig (P2.1)
int distance1;
         //for MOTOR
const int rp=10;
const int rn=11;
const int lp=6 ;
const int ln=9;
        //for CALCULATION
String data[5];
int i = 0;
char c;
float bearing;
float heading;
float d;
float finalv;
int l=0;
int k=0;
int distance;
float MSG[2]={};
void setup()
{
  Serial.begin(9600);
```

```
  Serial.begin(9600);
  gps.begin(9600);
  Serial.println("wait for 3 sec");
  delay(3000);
  gps.println("AT+CGNSPWR=1");
  delay(2000);
  gps.println("AT+CGNSSEQ=\"RMC\"");
  delay(2000);
  pinMode(rp, OUTPUT);
  pinMode(rn, OUTPUT);
  pinMode(lp, OUTPUT);
  pinMode(ln, OUTPUT);
  pinMode(Echo, INPUT);    //
  pinMode(Trig, OUTPUT);   //
  qmc.init();
  Wire.begin();
}

void loop() {
  k=k+1;
  headingcal();
  delay(100);
  gpsdata();
  delay(100);
  steering();
  delay(100);
  Serial.println("k =");
  Serial.println(k);
  if(k==250)
  {
    message();
    delay(200);
    k=0;
  }
  float x,y,deltalog,deltalat,latc1,logc1,latd1,logd1;
  latc1=latc.toFloat()* 0.01745;
  logc1=logc.toFloat()* 0.01745;
  latd1=latd.toFloat()* 0.01745;
  logd1=logd.toFloat()* 0.01745;
```

```
  deltalog= logd.toFloat()-logc.toFloat();
  deltalat=latd.toFloat() -latc.toFloat();
float  deltalog1=deltalog* 0.01745;
float  deltalat1=deltalog* 0.01745;
  x=cos(latd.toFloat())*sin(deltalog);
  y=(cos(latc.toFloat())*sin(latd.toFloat()))-
(sin(latc.toFloat())*cos(latd.toFloat())*cos(deltalog));

  bearing=(atan2(x,y))*(180/3.14);
  bearing= ( ((int)bearing + 360) % 360 );
  Serial.print("bearing:");
  Serial.print( "\n");
  Serial.println(bearing);
  float a,c;
  a=(sin(deltalat1/2))*(sin(deltalat1/2)) +
((cos(latc1))*(cos(latd1))*((sin(deltalog1/2))*(sin(deltalog1/2))));
  c=2*(atan2(sqrt(a),sqrt(1-a)));
  d=6371*c*1000;
  Serial.println("distance ");
  Serial.println(d);
   Serial.println("metre");
   c = gps.read();
        if (c != ',')
                {

                 if(bearing>=0&&bearing<180)
                 {
                   if(heading>=0&&heading<180)
                   {
                      if(heading<=bearing)
                     {
                      right();
                      delay(300);
                      stop1();
                     }
                     else if(heading>=bearing)
                     {
                      left();
                      delay(300);
```

```
                    stop1();
                   }
                    if(finalv>=0.92&&finalv<=1.08)
                   {
                      Ultrasonic();
                      if(distance1<=50)
                      {
                       stop1();
                       delay(50);
                       back();
                       delay(100);
                       left();
                       delay(500);
                       stop1();
                       Ultrasonic();
                       if(distance1<=50)
                       {
                          back();
                          delay(200);
                          stop1();
                          delay(300);
                          right();
                          delay(500);
                          stop1();
                          delay(500);
                          right();
                          Ultrasonic();
                          if(distance1<=50)
                              {
                                back();
                                delay(200);
                                stop1();
                                delay(300);
                                right();
                                delay(1000);
                                stop1();
                                delay(50);
//                                forward();
```

```
                }
                  else{
                  forward();
                      }
          }
               else{
                    forward();
                  }

         }else
         {
          forward();
         }

         Serial.println("forward");

     }
  }else
       {left();
       delay(300);
       stop1();}
if(d>=0&&d<=5)
       {
        stop1();
       }
}

    if(bearing>=180&&bearing<=360)
{
 if(heading>=180&&heading<=360)
 {
    if(heading<=bearing)
   {
    right();
    delay(300);
    stop1();
   }
   if(heading>=bearing)
   {
```

```
left();
delay(300);
stop1();
}
if(finalv>=0.92&&finalv<=1.08)
{
   Ultrasonic();if(distance1<=20)
   {
    stop1();
    delay(50);
    back();
    delay(100);
    left();
    delay(500);
    stop1();
    if(distance1<=20)
    {
      back();
      delay(200);
      stop1();
      delay(300);
      right();
      delay(500);
      stop1();
      delay(500);
      right();
      if(distance1<=20)
         {
           back();
           delay(200);
           stop1();
           delay(300);
           right();
           delay(500);
           stop1();
           delay(50);
           forward();

          }
```

```
                                else{
                                forward();
                                    }
                            }
                                 else{
                                  forward();
                                 }

                            }else
                            {
                             forward();
                            }

                            Serial.println("forward");
                        }
                    } else
                          {right();
                          delay(300);
                          stop1();}
                    if(d>=0&&d<=5)
                          {
                           stop1();
                          }
                    }
               else if(logd==logc && latc==latd)
                        {
                           stop1();
                           delay(10000);
                        }
                  }

 gpsdata();



}

void gpsdata()
{
```

```
int i=0;
logc = "";
latc= "";
  for(i=0; i<=5; i++)
    data[i]="";
   i=0;
Serial.println("----Command given----");
gps.flush();
long int time = millis();
gps.println("AT+CGNSINF");

while ((time+1000 ) > millis())
{
     while(gps.available())
      {
   //Serial.println("test point 33");
        c = gps.read();
        if (c != ',')
       {
         data[i] +=c;
         delay(20);
       }
        else
        i++;
    }
   if(i==5)
   break;
}



latc = data[3];
logc = data[4];



Serial.println(latc);
Serial.println(logc);
Serial.println("----Response Print-1--");
delay(500);
```

```
}

void headingcal()
{
 int x,y,z;
 qmc.read(&x,&y,&z);
 delay(100);

 heading=atan2(y,x)*180/3.14;//atan2(x,y)
 heading= ( ((int)heading + 360) % 360 );
 Serial.print("heading: ");
 Serial.print("\n");
 Serial.println(heading);
  if (heading == 0 || heading == 360 )
        {
            Serial.println("N");
        }
   if (heading > 0 && heading < 90 )
        {
            Serial.print("NE");
            Serial.print(heading);
            //Serial.println("E");
        }
   if (heading == 90)
        {
             Serial.println("E");
        }
   if (heading > 90 && heading < 180 )
        {
             Serial.print("SE");
             Serial.print(heading);
             //Serial.println("E");
        }
   if (heading == 180)
        {
             Serial.println("S");
        }
   if (heading > 180 && heading < 270 ) {
             Serial.print("SW");
```

```
                Serial.print(heading);
                //Serial.println("W");
          }
    if (heading == 270)
          {
                Serial.println("W");
          }
    if (heading > 270 && heading < 360 )
          {
                Serial.print("NW");
                Serial.print(heading);
               // Serial.println("W");
          }
}
void message()
{
    gps.print("AT+CMGF=1\r");
    delay(2000);
    gps.print("AT+CMGS=\"9348707870\"\r");
    delay(2000);
    gps.print("\r");
    delay(2000);
    gps.println("Dear Bora and Gupta your vehicle is here");
    gps.print("http://maps.google.com/maps?q=loc:");
    gps.print(latc);
    gps.print(",");
    gps.print(logc);
    delay(2000);
    gps.println((char)26);
    delay(2000);
}

void steering()
{
  finalv=heading/bearing;
  Serial.print("finalv: ");
  Serial.println(finalv);

}
```

```
void Ultrasonic()
{
 digitalWrite(Trig, LOW);   // 2s
 delayMicroseconds(2);

 digitalWrite(Trig, HIGH);  // 10s10s
 delayMicroseconds(10);
 digitalWrite(Trig, LOW);    //
 float Fdistance = pulseIn(Echo, HIGH);  // ()
 Fdistance= Fdistance/58;       //58  Y=X*344/2
 // X= 2*Y/344 ==X=0.0058*Y ===/58
 //Serial.print("Distance:");      //
 //Serial.println(Fdistance);         //
 distance1 = Fdistance;
 Serial.println("ud=");
 Serial.println(distance1);
 return Fdistance;
}

void forward()
{

 analogWrite(rp,240);
 analogWrite(rn,0);
 analogWrite(lp,220);
 analogWrite(ln,0);
 Serial.print("F");

}

void right()



{
 analogWrite(rp,0);
 analogWrite(rn,130);
 analogWrite(lp,130);
 analogWrite(ln,0);
 Serial.print("R");
```

```
}

void left()
{
 analogWrite(rp,130);
 analogWrite(rn,0);
 analogWrite(lp,0);
 analogWrite(ln,130);
 Serial.print("L");

}

void back()
{
 analogWrite(rp,0);
 analogWrite(rn,110);
 analogWrite(lp,0);
 analogWrite(ln,110);
 Serial.print("B");
}
void stop1()
{
  analogWrite(rp,0);
 analogWrite(rn,0);
 analogWrite(lp,0);
 analogWrite(ln,0);
 Serial.print("stop");
 delay(100);
}
```