DLC PROJECT REPORT

Design and Implementation of Mapping Robot Using Ultrasonic Sensor

Course Title: Digital Logic & Circuits

Course Code: EL101

Course Incharge: Vikas Upadhyaya

Academic Year: 2015-16(Semester III)

Submitted By:

Vadlamudi Vamsheeth U101115FCS274(S7)

A Mano Vishnu U101115FEC012(S7)

Velineni Manish Sai U101115FCS175(S7)

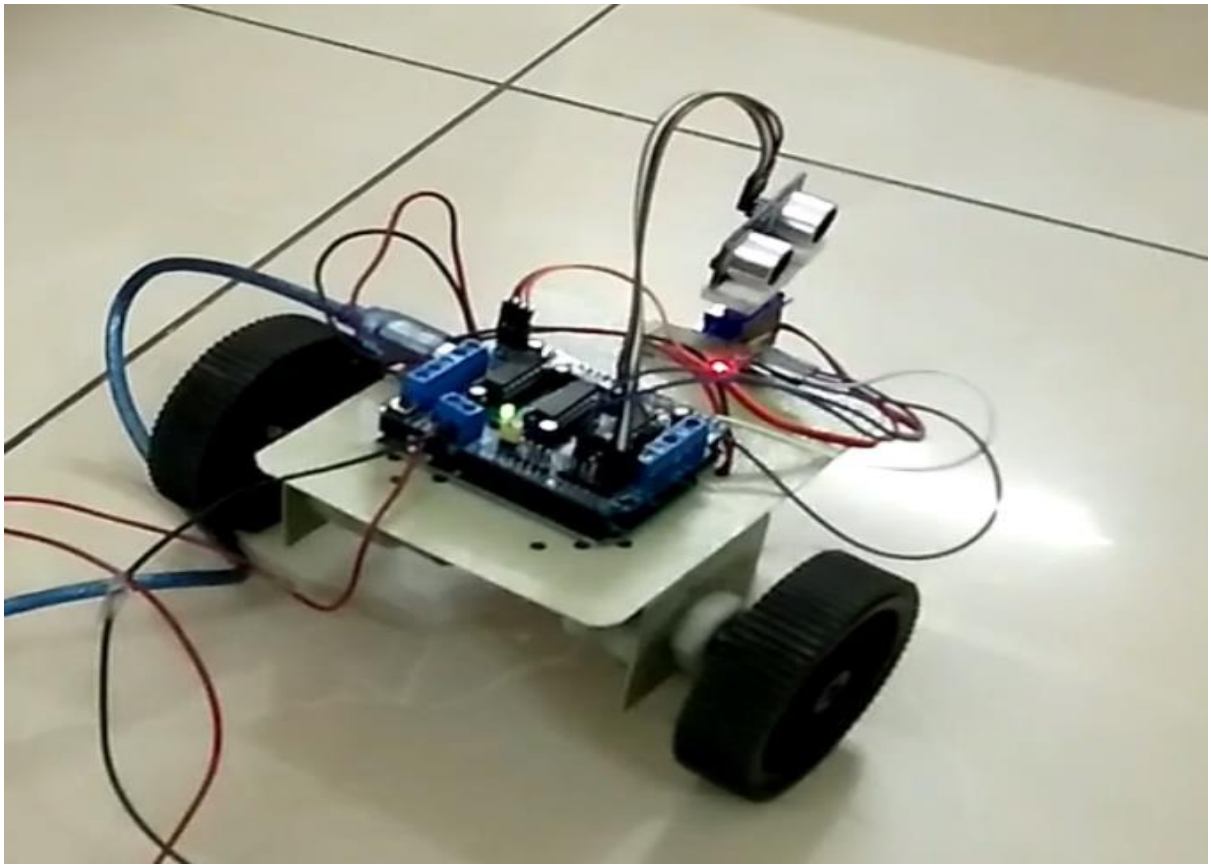
Sai Chaitanya Kathiri U101115FCS137(S6)

INTRODUCTION:

Role of sensing system is to detect the presence of objects and measure their positions. The objects can be obstacles and target. This approach of mapping is one that determines the geometric properties of the environment. This representation is very useful, but is sensitive to noise. Simultaneous localization and mapping determines the location of a rover and construct map of an unknown environment at the same time. Rover path and map are both unknown. An autonomous rover must recognize its position and find the path for itself. Rover need to plan a motion path through the environment to navigate without colliding with obstacles. Data obtained by ultrasonic range measurements is used to detect and avoid obstacles in environment. The design and implementation of trajectory mapping robot for SWARM application is done using Arduino UNO microcontroller. We present mapping of mobile rover in the indoor environment Ultrasonic sensor. The designed robot can be sent into an unknown building to produce a floor map. We used MATLAB software to plot map against co-ordinates send by mobile robot to PC.

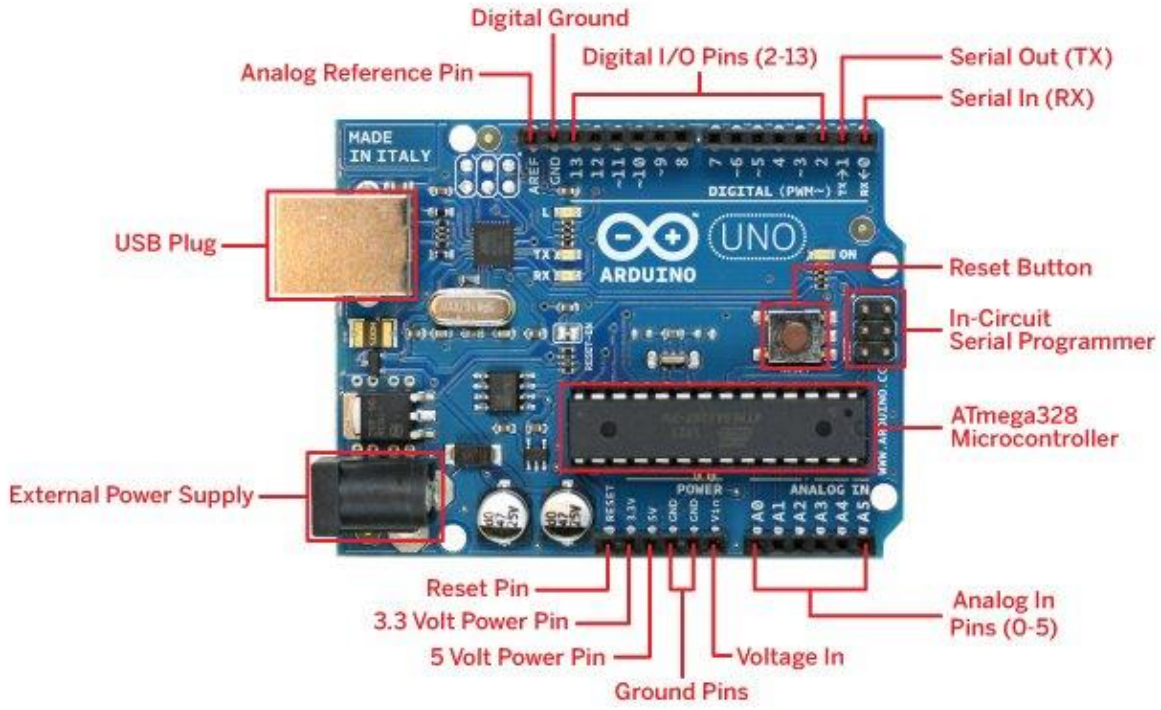
COMPONENTS:

-Arduino ATmega328p microcontroller
-Ultrasonic Sensor
-Motor Driver IC L293D
-Servo Motor



Arduino ATmega328p microcontroller:

The ATmega328P is a low-power CMOS 8-bit microcontroller. By executing instructions in a single clock cycle, the ATmega328P achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.
Arduino ATMega 328p Microcontroller combines 32KB ISP flash memory with read-while-write capabilities, 1024B EEPROM, 2KB SRAM, 23 general purpose I/O lines, 32 general purpose working registers, three flexible timer/counters with compare modes, internal and external interrupts, serial programmable USART, a byte-oriented 2-wire serial interface, SPI serial port, a 6-channel 10-bit A/D converter (8-channels in TQFP and QFN/MLF packages), programmable watchdog timer with internal oscillator, and five software selectable power saving modes. The device operates between 1.8-5.5 volts.



Ultrasonic Sensor(HC-SR04):

The Ultrasonic Sensor sends out a high-frequency sound pulse and then times how long it takes for the echo of the sound to reflect back. The sensor has 2 openings on its front. One opening transmits ultrasonic waves, (like a tiny speaker), the other receives them, (like a tiny microphone).
The speed of sound is approximately 341 meters (1100 feet) per second in air. The ultrasonic sensor uses this information along with the time difference between sending and receiving the sound pulse to determine the distance to an object. It uses the following mathematical equation:

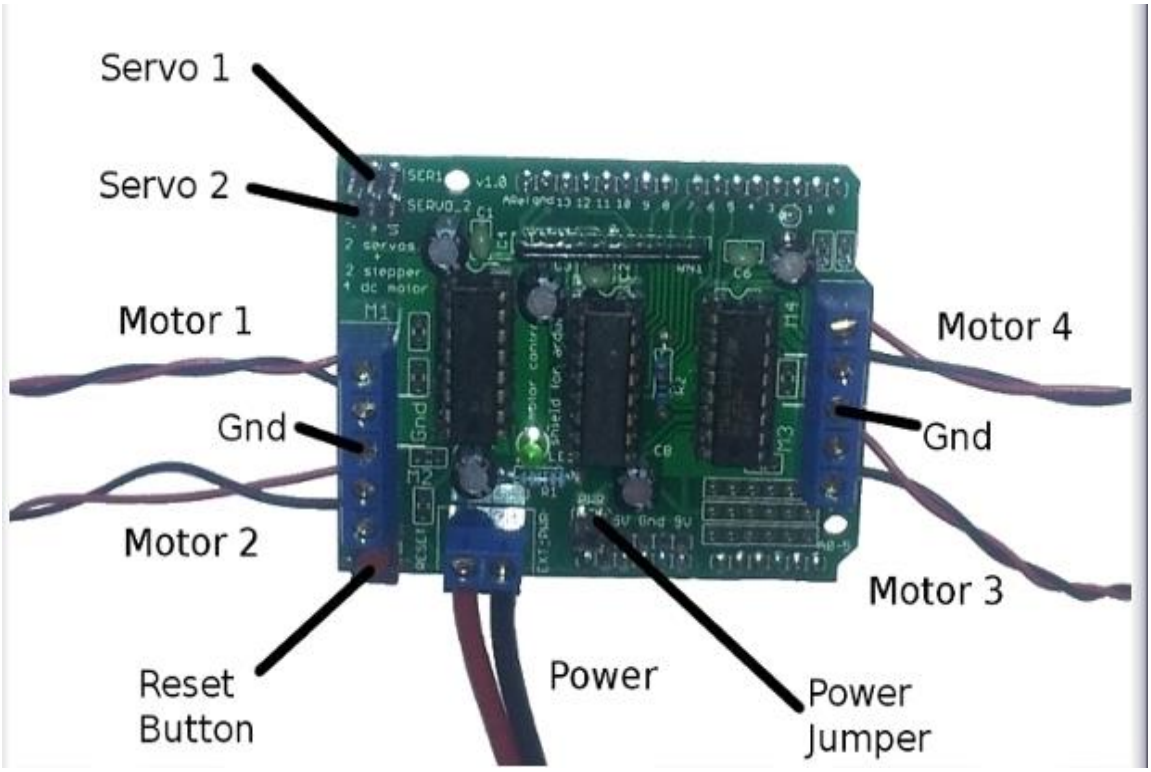
Distance = Time x Speed of Sound divided by 2

Time = the time between when an ultrasonic wave is transmitted and when it is received
You divide this number by 2 because the sound wave has to travel to the object and back

Vcc- Connects to 5V of positive voltage for power
Trig- A pulse is sent here for the sensor to go into ranging mode for object detection
Echo- The echo sends a signal back if an object has been detected or not. If a signal is returned, an object has been detected. If not, no object has been detected.
GND- Completes electrical pathway of the power.

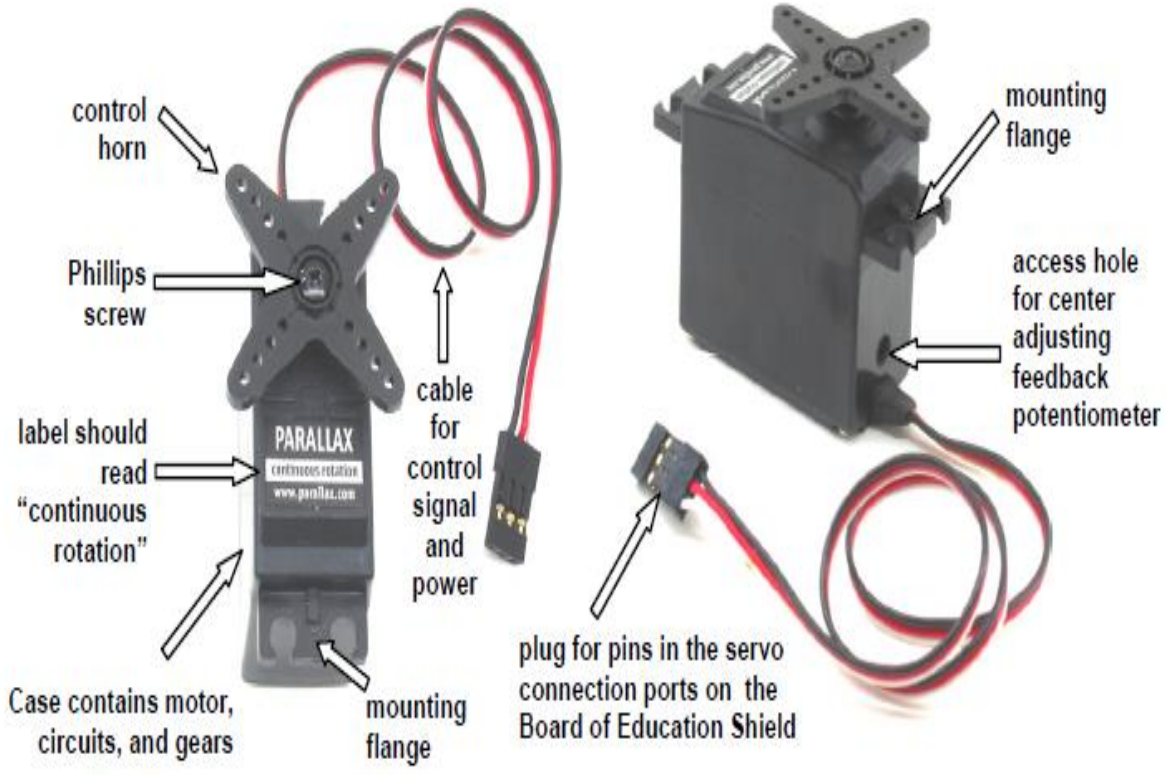
Motor Driver IC L293D:

This is a motor driver IC that can drive two motor simultaneously. L293D IC is a dual H-bridge motor driver IC. One H-bridge is capable to drive a dc motor in bidirectional. L293D IC is a current enhancing IC as the output from the sensor is not able to drive motors itself so L293D is used for this purpose. L293D is a 16 pin IC having two enables pins which should always be remain high to enable both the H-bridges.



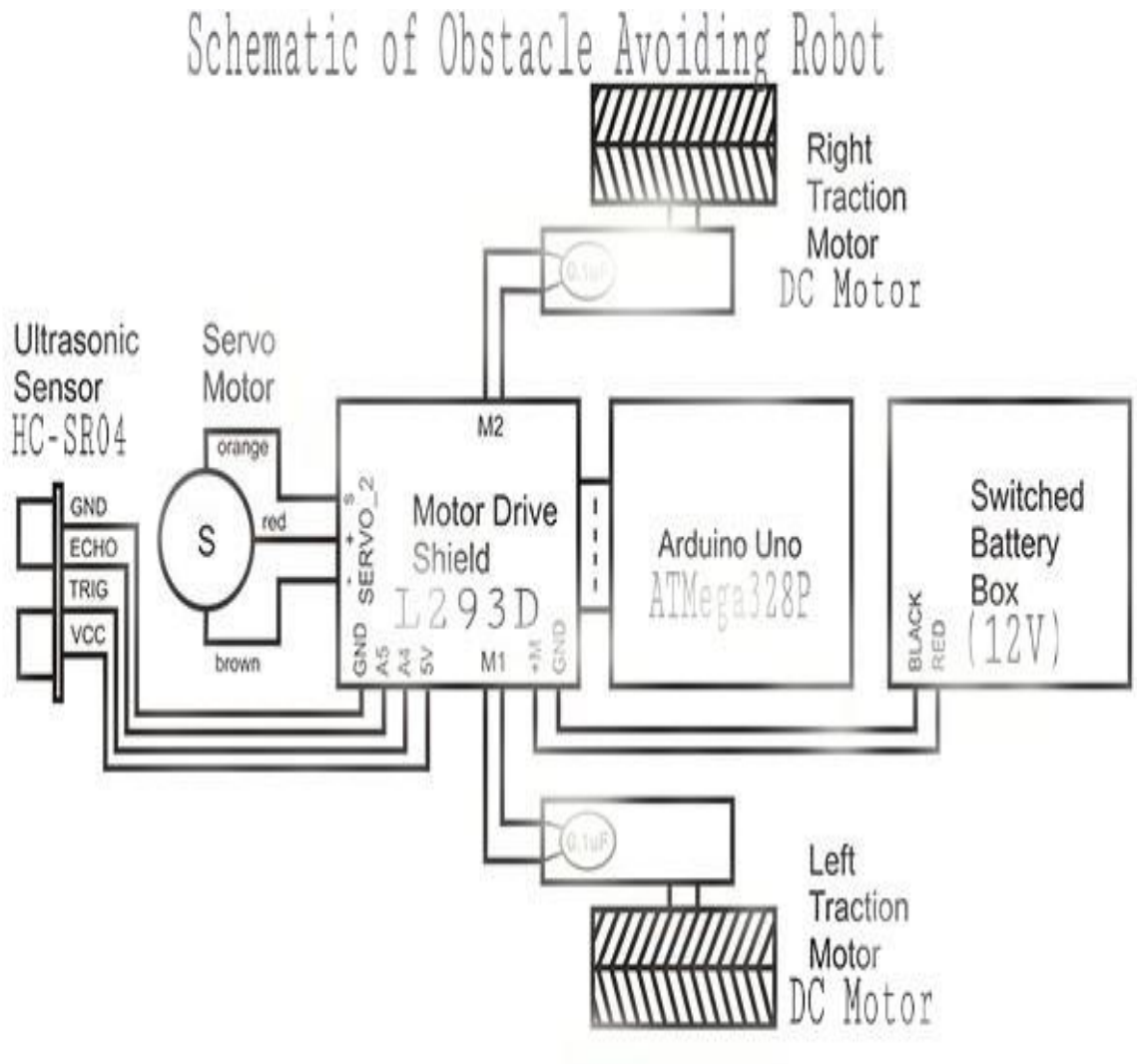
Servo Motor

This is nothing but a simple electrical motor, controlled with the help of servomechanism. There are some special types of application of electrical motor where rotation of the motor is required for just a certain angle not continuously for long period of time. For these applications, some special types of motor are required with some special arrangement which makes the motor to rotate a certain angle for a given electrical input (signal). For this purpose, servo motor comes into picture. This is normally a simple DC motor which is controlled for specific angular rotation with the help of additional servomechanism (a typical closed loop feedback control system).



WORKING OF COMPONENTS:

Mapping rover has Arduino UNO board with ATmega328 microcontroller with other electronic components which can be programmed using the software. Rover has a ultrasonic distance sensor HC - SR04 used for navigation. The I^2C serial bus allows for easy interface. Working range of Ultrasonic ranging module HC - SR04 is 2cm to 400cm with accuracy of 3mm. Output voltage from sensor is corresponding to the detection distance from sensor to an object. Robot has two DC geared motors for motion control. One caster wheel is attached at the front end of rover for support. Driving system of rover allows it to move forward, backward and rotate clockwise or anticlockwise. Communication between rover and PC is achieved using a connection from com port of the Arduino to pc. This rover has 12 Volts battery for powering of driving system and 9 Volts battery for Arduino. PC has MATLAB software with to communicate with PC's COM port. Live coordinates send by mobile rover receives by PC and map is plotted on MATLAB graph.



Applications:

- Indoor Positioning System
- Map Database Management
- Patrol Bot
- Diffusion Mapping etc.,

Conclusion:

We presented hardware design of mapping robot using Arduino UNO controller board and other sensors. Obstacle avoidance algorithm is successfully implemented on robot for navigation. Robot successfully sends the distance and width of obstacle to base station and mapping of Obstacle(Dimensions) is observed from experiment. The designed robot uses a metric, world-centric approach for mapping algorithm. All experiments show expected results. Future work would include serial data communication with Bluetooth module(HC-05) and mapping of real time coordinates using Magnetometer. Also, large area mapping with use of GPS can be developed in future.

Code used:

MATLAB CODE:

```
theta = 0:(pi/180):pi;

    s = serial('COM5');
    s.BaudRate=9600
    fopen(s)
    i = 0;

    inc = 1;

    while i<180
      A = fgets(s);
      num(i+1) = str2num(A);
      i = i+1;
    end
    fclose(s)

    j = 1

    while j<181
       tab(j,1) = (j-1)*inc
       tab(j,2) = num(j)
       tab(j,3) = num(j)*cosd((j-1)*inc)
       tab(j,4) = num(j)*sind((j-1)*inc)
       j = j+1
    end
    %figure
    %polar(theta,num)

    plot(tab(:,3),tab(:,4))
```

ROVER CODE:

```
#include <AFMotor.h>                                  //includes AFmotor Library
#include <NewPing.h>                                  //includes NewPing Library
#include <Servo.h>                                    //includes Servo Library
#define TRIGPIN A4
#define ECHOPIN A5
#define MAXDISTANCE 350
#define MAX_PEED 180                                   // sets speed of DC Motors
#define MAXSPEEDOFFSET 20
NewPing sonar(TRIGPIN, ECHOPIN, MAXDISTANCE);
AF_DCMotor motor1(1, MOTOR12_64KHZ);
AF_DCMotor motor3(3, MOTOR12_64KHZ);
Servo myservo;
int pos = 0;
int it = 10;
boolean goesForward=false;
int distance = 100;
int speedSet = 0;
void setup()
 {
  Serial.begin(9600);
 myservo.attach(9);
  myservo.write(90);
  delay(2000);
  distance = readPing();
  delay(100);
  distance = readPing();
  delay(100);
  distance = readPing();
  delay(100);
  distance = readPing();
  delay(100);
}

void loop()
{
int distanceR = 0;
int distanceL =  0;
delay(40);
if(distance<=15)
          {
          moveStop();
          delay(100);
```

```
 moveBackward();
            delay(300);
            moveStop();
            delay(200);
            plotpath();
            delay(200);
            distanceR = lookRight();
            delay(200);
            distanceL = lookLeft();
            delay(200);
  if(distanceR>=distanceL)
            {
            turnRight();
            moveStop();
            }
else
             {
            turnLeft();
            moveStop();
            }
            }
else
            {
            moveForward();
            }
 distance = readPing();
}

int lookRight()
{
   myservo.write(50);
   delay(500);
   int distance = readPing();
   delay(100);
   myservo.write(90);
   return distance;
}

int lookLeft()
{
   myservo.write(170);
   delay(500);
   int distance = readPing();
   delay(100);
```

```
   myservo.write(90);
   return distance;
   delay(100);
}

int readPing()
{
 delay(70);
 int cm = sonar.ping_cm();
 if(cm==0)
          {
          cm = 250;
          }
 return cm;
}

void moveStop()
 {
 motor1.run(RELEASE);
 motor3.run(RELEASE);
 }
  void moveForward()
 {
 if(!goesForward)
          {
          goesForward=true;
          motor1.run(FORWARD);
          motor3.run(FORWARD);
          for (speedSet = 0; speedSet < MAX_SPEED; speedSet +=2)
                    {
                    motor1.setSpeed(speedSet);
                    motor3.setSpeed(speedSet+MAX_SPEED_OFFSET);
                    delay(5);
                    }
          }
}
void moveBackward()
{
 goesForward=false;
 motor1.run(BACKWARD);
 motor3.run(BACKWARD);
 for (speedSet = 0; speedSet < MAX_SPEED; speedSet +=2)
          {
          motor1.setSpeed(speedSet);
```

```
        motor3.setSpeed(speedSet+MAX_SPEED_OFFSET);
        delay(5);
        }
}

void turnRight()
 {
  motor1.run(FORWARD);
  motor3.run(BACKWARD);
  delay(300);
  motor1.run(FORWARD);
  motor3.run(FORWARD);
}

void turnLeft()
 {
  motor1.run(BACKWARD);
  motor3.run(FORWARD);
  delay(300);
  motor1.run(FORWARD);
  motor3 .run(FORWARD);
}
void plotpath()
{
  int i=0;
  int t=0;
  int a=0;
  for (i = 0; i < 180; i ++)
        {
        unsigned int uS = sonar.ping();
        myservo.write(i);
        delay(1);
        for (t = 0; t < it; t++)
                {
                uS = sonar.ping();
                a = uS/US_ROUNDTRIP_CM + a;
                delay(2);
                }
        a = a / (it-1);
        t = 0;
        Serial.println(a);
        a = 0;
        }
}
```