Habib University

Dhanani School of Science and Engineering

Microcontrollers & Interfacing

EE 375L-T1/T2

Milestone 3 report

Team: Circuit Cruiser

Group Members:	**Student ID:**
Aamaina Mukarram Tahir Ali	08391
Abdullah Khalid	08428
Fatima Binte Aijaz	08519

Table of contents

Intro

Objectives: **Modules and sensor interfacing:** In this milestone, functionality wise, ultrasonic and IR sensors integration was done, leading to obstacle detection and collision avoidance. The unloading mechanism was implemented, and the functionality of the robot is now solely controlled by bluetooth. All sensors and modules integrated into the robot are fully functional now. This report aims to explain how the hardware, sensors, are used to control the mechanism through programming via bluetooth.

Stance: Progressing with the hardware, the image of the final robot, how we imagined it and sketched it in the proposal in milestone 1 has all been put onto the robot. All the electronics on the robot are fully functional now. Of the various tasks the robot is supposed to carry out throughout the whole project, the robot avoids collision, detects walls and unloads, in addition to the mobility in milestone 2 which included linear motion and obstacle avoidance. This achieves a significant portion of the functionality requirements of the robot.

Images of updated robot

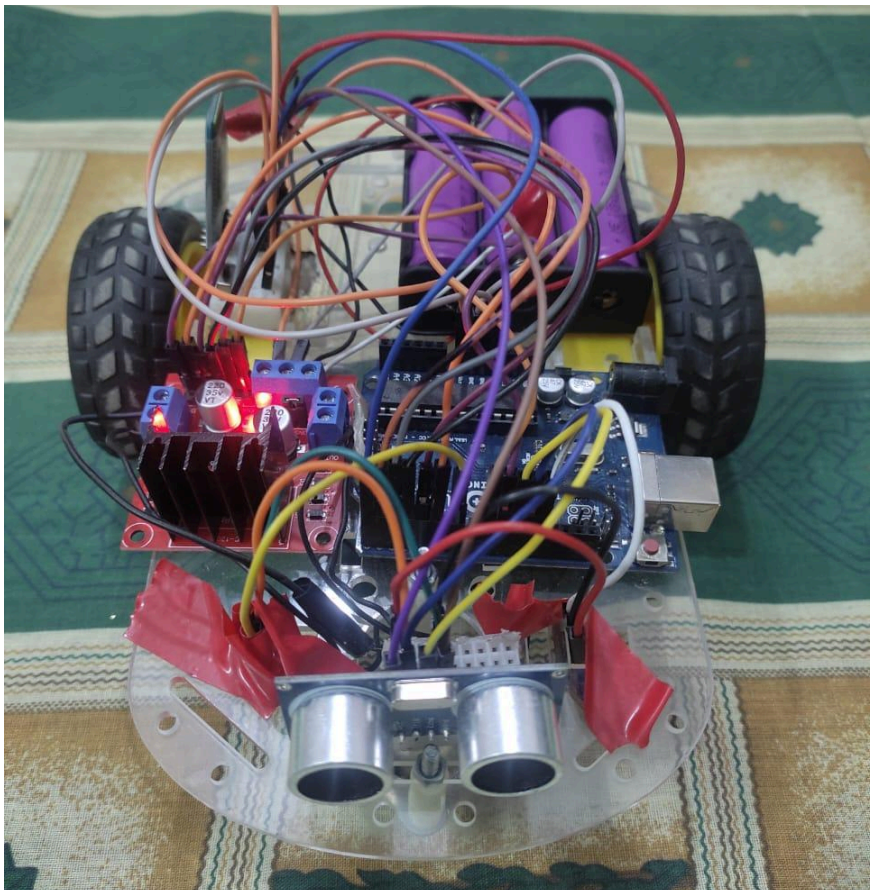

Figure 1: Robot front view

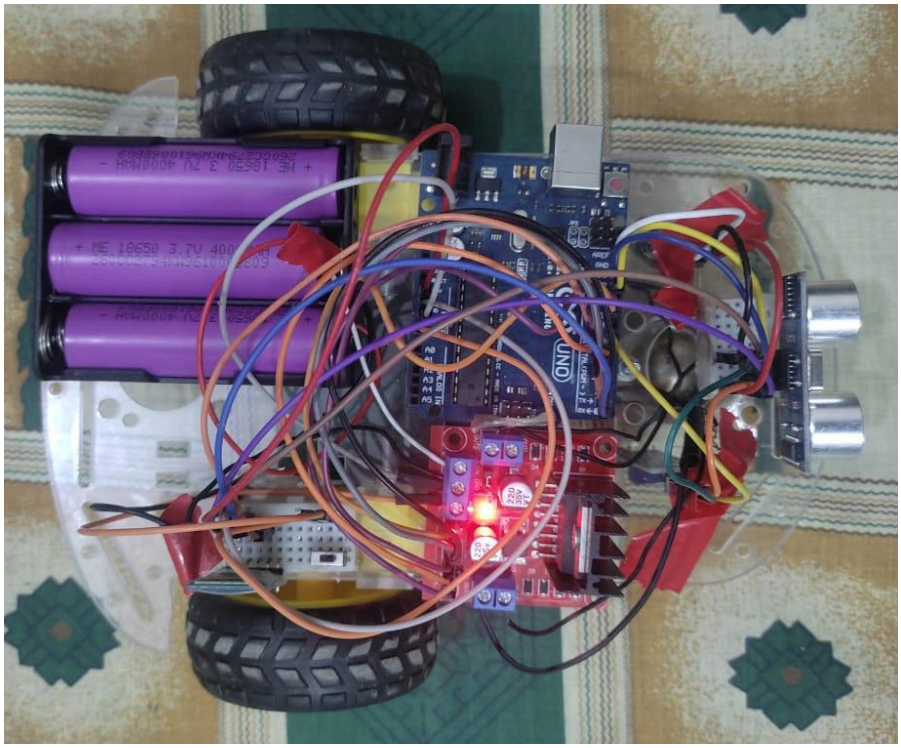

Figure 2: Robot top view

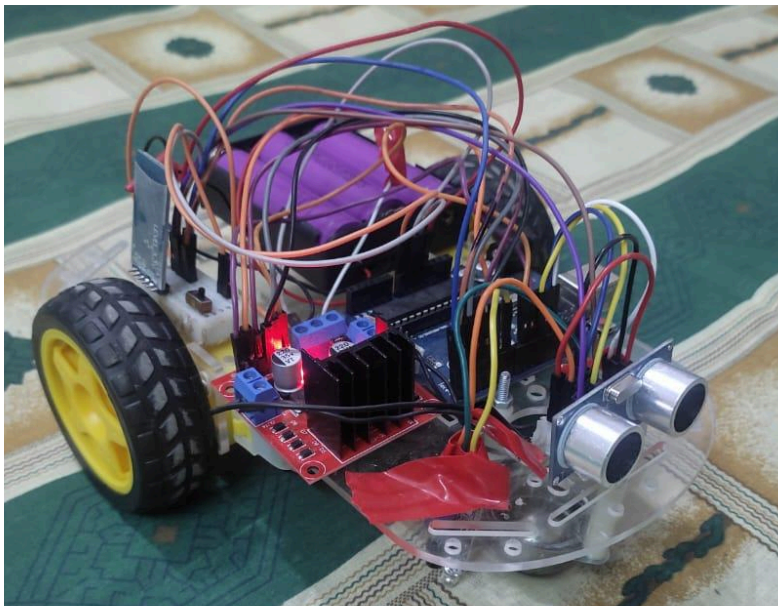

Figure 3: Robot 3/4th view

Explanation of functionality

Bluetooth communication module

Our robot is now fully controlled by bluetooth. The device used to control the bluetooth communication is called HC-05.

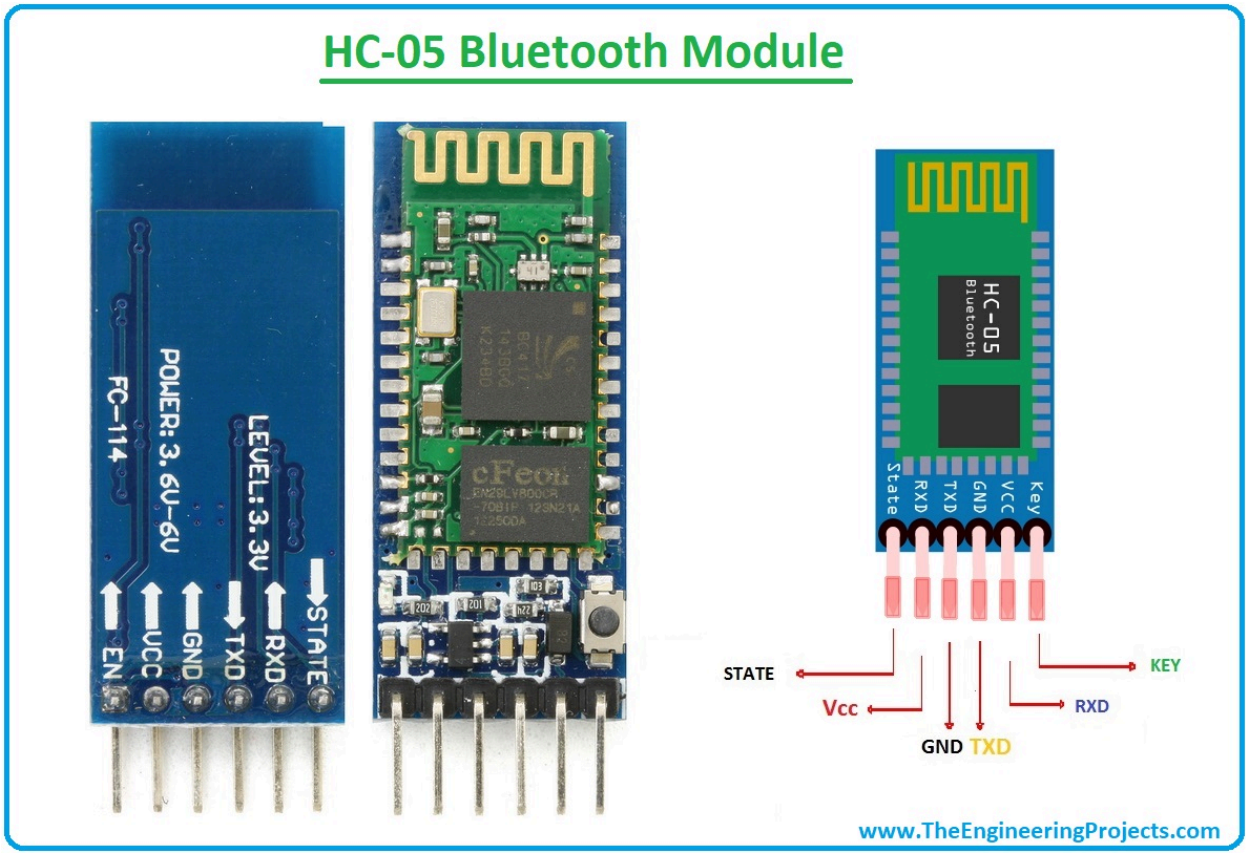

Figure 4: HC-05

An application called Serial Bluetooth is installed on a mobile phone or device of our choice from where to send commands. The HC-05 receives the commands and prompts the robot to perform the operation accordingly.

The HC-05 is a module that's made for bluetooth communication. It consists of 6 pins, which are as follows:

- State: It's a pin that is connected to an LED to indicate the operating state of the bluetooth module. Currently, we are not using this pin in our project.
- RXD: Serial receiving pin, which receives a signal
- TXD: Serial transmitting pin
- GND: ground pin
- Vcc pin, for power supply. It takes 5 Volts as input.
- En: to set a data value at a higher or lower value.

In the main loop of our code, Receive Bluetooth data, a function, is called.

```
void loop() {
    //L298N_SampleTest();
    Receive_Bluetooth_Data();
```

This ensures that this function runs every time. The function is as follows:

```
void Receive_Bluetooth_Data()
{
  int data;
  //char[] string ;
  if(Serial.available()>0)
  {
    while(Serial.available()>0)
    {
      data = Serial.read();
      Input = char(data);
      Serial.print(Input);
      break;
    }
  }
  else {
  Input = '0';
  }
}
```

It checks availability of data using the function Serial.available. Incase of data present the data received is stored in the variable ‘data’ using Serial.read function, and the information is printed. The data is stored in the variable ‘Input’. After this runs in the main loop,

```
Receive_Bluetooth_Data();

 // Sensors

// CHeck if Input is for automovement or manual movement
if(Input == 'A'){Auto = true; Followline = false; }
if(Input == 'M'){Auto = false; Followline = false; }
if(Input == 'L'){Followline = true; Auto = false; }
```

Variables values are set (boolean values) which will direct what operation to perform.

Ultrasonic sensor

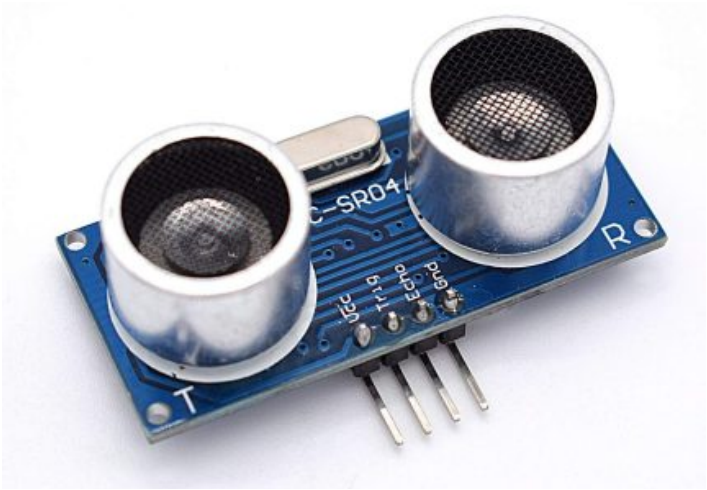

Figure 5: The ultrasonic sensor

An Ultrasonic sensor is a sensor that uses ultrasonic sound waves to detect an object and calculate distance between itself and the object. It does so by sending and receiving pulses; it releases sound in a frequency that is not audible for humans and calculates distance by tracking the time it took for the wave to reach the object in front and back to it. Its trigger pin starts the sensor and the echo pin receives it.

In our robot, we use the ultrasonic sensor to detect obstacles. In case of an obstacle, We use if else statements to rotate the robot to guide it to another direction. Collision is avoided in this way, keeping a safe threshold distance which enables the robot to stop and move.

```
void Automovement()
{
  if(obstacle)
  {
    rotate360(0,70);
  }
  else{
    moveForward(0);
  }
}
```

(for more information on the functionality of the ultrasonic sensor, please find the code attached.)

IR sensor

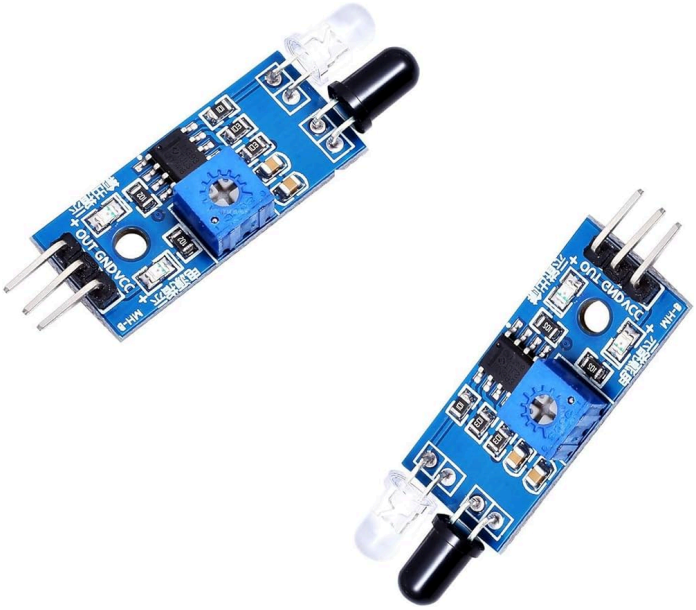

Figure 6: The IR sensor

An IR sensor is a device that detects infrared radiation in its environment and outputs an electrical signal. It is basically a photodiode which is sensitive to infrared light. It consists of an IR LED and a photodiode. The IR LED emits infrared light and the photodiode detects the light. In our robot, this sensor is placed on the bottom of the robot. The arena, where the robot will run, has a white background with black line. The resultant signal is then processed by the code in arduino. It takes 5 readings of the signal, and then decides whether a black line is present or not. It also has a clue if the black line is on the left or right, so the robot motion can be directed accordingly. The respective detected statements are also printed for debugging. (For more information on the working, please find the code attached).

```
bool updateIR(const int PIN)
{
  int i;
  //if (PIN == 13){}
  for (i=0; i<5 ; i++)
  {
    if(digitalRead(PIN) == HIGH)
    {
      //Serial.println("OBJECT IN FRONT");
      if (PIN == 13){Serial.println("BLACK LINE on LEFT IR");}
      else{Serial.println("BLACK LINE on RIGHT IR");}
      return true;
    }
    delay(1) ; // take 5 accurate readings
  }
  return false;
}
```

Unloading mechanism (servo)

Figure 7: Servo motor

In our robot, we have used a servo motor to control the unloading mechanism. A small drawer like compartment is attached to the moving part of the servo (white, in the picture). We can use servo motion to move and rotate the drawer compartment, leading to unloading of an object.

Task division

Abdullah mainly dealt with hardware, Amena with the arena, and Fatima with the report. Amena and Fatima also worked on the code.

Milestone 3 summary

Progress: Further delving into the functionality of the robot, additional features have just been added and the robot has been upgraded. A robot isolated from the PC/laptop and can now work with bluetooth. Furthermore, all using sensors have been integrated into the robot which provide obstacle detection and hence collision avoidance.

Key achievement: Upgraded hardware functionality

Feedback and challenges: Although the robot was fully functional when testing, it sometimes creates errors and does not run when we try to replicate it, at a different time or in a different environment. Considering integration of a lot of components, the robot now has a lot of wires which gets tricky to handle, also because the jumper wires are fragile and easily moveable. We have to treat our robot with extreme precaution in order to protect the functionality. We faced a hindrance because once a Vcc wire was accidentally connected to the ground terminal.

Remarks: Its nevertheless fun to work on this project, especially if the results are fruitful.

All codes

```
//#include<SoftwareSerial.h>

//==============TODO's=================
//Acceleration mechanism for better line following
//=====================================

// Operationmodes
bool Auto;
bool Followline = true;

// L298 pins
int motor1pin1 = 2;
int motor1pin2 = 3;

int motor2pin1 = 4;
int   motor2pin2 = 5;

//Ultrasonic
const int echoPin = 11;
const int trigPin = 12;

// HC_05
char Input; // will have values zero onewards
bool obstacle;

// IR sensor
const int IR1 = 7;
const int IR2 = 13; // pin 8 conflicting with echo pin of ultrasonic
bool left;
bool right;

int speed;
// int RX = 0;
// int TX = 1;
// SoftwareSerial bt(RX,TX);

void setup() {
  Serial.begin(9600);
```

```
  init_L298N();
  initUltrasonic();
  InitIR();
  Auto = false ;
  Followline = false ;
  speed = 70;
  //L298N_SampleTest();
}

void loop() {
   //L298N_SampleTest();
   Receive_Bluetooth_Data();

    // Sensors

   // CHeck if Input is for automovement or manual movement
   if(Input == 'A'){Auto = true; Followline = false; }
   if(Input == 'M'){Auto = false; Followline = false; }
   if(Input == 'L'){Followline = true; Auto = false; }

  Handle_movement() ;// for interupts

   if(Auto) // detect collisions
   {
    obstacle = is_Obstacle(40);
      Automovement();
      delay(100);
   }
   else if(Followline) // line following mode
   {
    right = updateIR(IR1);
   left = updateIR(IR2);
    LineFollower();
    delay(20) ;
   }
   else{ // manual movement mode
      Handle_movement();
      //moveForward(0);
   }
  delay(10) ;
```

```
}


//====================== Motor driver management ======================

// initialize motor
void init_L298N()
{
  pinMode(motor1pin1, OUTPUT);
  pinMode(motor1pin2, OUTPUT);
  pinMode(motor2pin1,   OUTPUT);
  pinMode(motor2pin2, OUTPUT);

  //(Optional)
  pinMode(9,   OUTPUT);
  pinMode(10, OUTPUT);

  // initialize motor speeds
  resetspeed(speed);
}

void resetspeed(int sp)
{
  int newspeed = sp;
  if (sp < 0)
  {
    newspeed = 0;
  }

  if (sp>250)
  {
    newspeed = 250;
  }

  analogWrite(9, newspeed); //ENA   pin
  analogWrite(10, newspeed); //ENB pin
}

// dry run motor
void L298N_SampleTest()
```

```
{

  digitalWrite(motor1pin1,   HIGH);
  digitalWrite(motor1pin2, LOW);

  digitalWrite(motor2pin1, HIGH);
   digitalWrite(motor2pin2, LOW);
  delay(3000);

  digitalWrite(motor1pin1,   LOW);
  digitalWrite(motor1pin2, HIGH);

  digitalWrite(motor2pin1, LOW);
   digitalWrite(motor2pin2, HIGH);
  delay(3000);

  // turn off both motors after that
  Stop_Motors();
}

//============ bluetooth management ========================

void Receive_Bluetooth_Data()
{
  int data;
  //char[] string ;
  if(Serial.available()>0)
  {
    while(Serial.available()>0)
    {
      data = Serial.read();
      Input = char(data);
      Serial.print(Input);
      break;
    }
  }
  else {
  Input = '0';
  }
}
```

```
// ====================== movement ============================

void moveForward(int time) {
  resetspeed(speed);
  digitalWrite(motor1pin1, HIGH);
  digitalWrite(motor1pin2, LOW);
  digitalWrite(motor2pin1, HIGH);
  digitalWrite(motor2pin2, LOW);
  delay(time); // Adjust delay for desired duration of forward movement
  //Stop_Motors();
}

// Rotate 360 degrees
void rotate360(int time, int sp) {

  resetspeed(sp);
  // Implement rotation logic by running motors appropriately for
360-degree turn
  digitalWrite(motor1pin1, HIGH);
  digitalWrite(motor1pin2, LOW);
  digitalWrite(motor2pin1, LOW);
  digitalWrite(motor2pin2, HIGH);
  delay(time); // Adjust delay for desired duration of rotation
  //Stop_Motors();
}

// rotate anticlockwise
void antirotate360(int time,int sp)
{
  resetspeed(sp);
  digitalWrite(motor1pin1, LOW);
  digitalWrite(motor1pin2, HIGH);
  digitalWrite(motor2pin1, HIGH);
  digitalWrite(motor2pin2, LOW);
  delay(time);
 // delay(100); // Adjust delay for desired duration of rotation
}

// Rotate right by 45 degrees
```

```
void rotateRight45() {
  // Implement rotation logic by running motors appropriately for right
turn
  digitalWrite(motor1pin1, HIGH);
  digitalWrite(motor1pin2, LOW);
  digitalWrite(motor2pin1, LOW);
  digitalWrite(motor2pin2, HIGH);
  delay(100); // Adjust delay for desired duration of rotation
  Stop_Motors();
}

// Rotate left by 45 degrees
void rotateLeft45() {
  // Implement rotation logic by running motors appropriately for left
turn
  digitalWrite(motor1pin1, LOW);
  digitalWrite(motor1pin2, HIGH);
  digitalWrite(motor2pin1, HIGH);
  digitalWrite(motor2pin2, LOW);
  delay(100); // Adjust delay for desired duration of rotation
  Stop_Motors();
}

// istantly stop both motors
void Stop_Motors()
{
  digitalWrite(motor1pin1,   LOW);
  digitalWrite(motor1pin2, LOW);

  digitalWrite(motor2pin1, LOW);
   digitalWrite(motor2pin2, LOW);

   Auto = false;
   Followline = false;
}
void Handle_movement()
{
switch (Input) {
    case 'F':
      moveForward(100);
```

```
      break;
    case 'R':
      rotate360(1000,80);
      break;
    case 'r':
      rotateRight45();
      break;
    case 'l':
      rotateLeft45();
      break;
    case 'S':
      Stop_Motors();
      break;
    case 'a':
      Accelerate();
      break;
    case 'd':
      Decelerate();
      break;

    default:
      // No action for other inputs
      break;
}
}

//======================Ultrasonic=============================

void initUltrasonic()
{
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}

// take Inputs from ultrasonic and return a true if object found within
the desired distance in cm
bool is_Obstacle(long range)
{
  long duration;
  long cm;
```

```
  //Setting Trig Signal HIGH for 10us to produce burst of 8 pulses at
40KHz
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  //digitalRead(anypin/switch) //Use this function to read the state of
any Pin/Switch i.e. SW1 and SW2

  duration = pulseIn(echoPin, HIGH);   // Reads duration (microseconds)
for which Echo pin reads HIGH till wave is reflected
  cm = microsecondsToCentimeters(duration); // Handle Average Reading
mechanism Separately

  Serial.print("Distance=");
  //Serial.print("\t");
  Serial.print(cm);
  Serial.print("\n");

  if(range > cm){return true;}
  else {return false;}


}

long microsecondsToCentimeters(long microseconds)
{
long distance;
// The speed of sound is 340 m/s or 29 microseconds per centimeter.
distance = microseconds / 29 ;
return distance / 2;
}

void Automovement()
{
  if(obstacle)
   {
      rotate360(0,70);
```

```
    }
    else{
       moveForward(0);
    }
}

void LineFollower()
{
  if(right)
  {
    rotate360(70,speed);
  }
  else if(left)
  {
    antirotate360(70,speed);
  }
  else
  {
    moveForward(0);
  }
}
//======================IR sensor=================
void Accelerate()
{
  speed += 5;
}
void Decelerate()
{
  speed -= 5;
}
void InitIR()
{
  pinMode(IR1,INPUT);
  pinMode(IR2,INPUT);
}

bool updateIR(const int PIN)
{
  int i;
  //if (PIN == 13){}
```

```
  for (i=0; i<5 ; i++)
  {
    if(digitalRead(PIN) == HIGH)
    {
      //Serial.println("OBJECT IN FRONT");
      if (PIN == 13){Serial.println("BLACK LINE on LEFT IR");}
      else{Serial.println("BLACK LINE on RIGHT IR");}
      return true;
    }
    delay(1) ; // take 5 accurate readings
  }
  return false;
}
```

References

www.theengineeringprojects.com
www.robocraze.com