EEE 187 Robotics

Laboratory 01

4WD BT Multi-purpose Car V2.0 Kit

Vigomar Kim Algador

Fall 2023

Table of Contents

INTRODUCTION

In this laboratory, we are required to build a mechanical robot smart car. It is powered by Arduino microcontroller which is open-source electronics platform to build digital devices for the mechanical smart car. The main feature required for this car is to able to move in different directions using a Bluetooth control connected from smart phone. Special features include obstacle avoidance, following, IR remote, ultrasonic following, and a display. On the other hand, the software part will be using C language code of Arduino IDE.

ASSEMBLY

With the given instructions by the laboratory kit, I was able to assemble the components required for the smart robot as shown below.

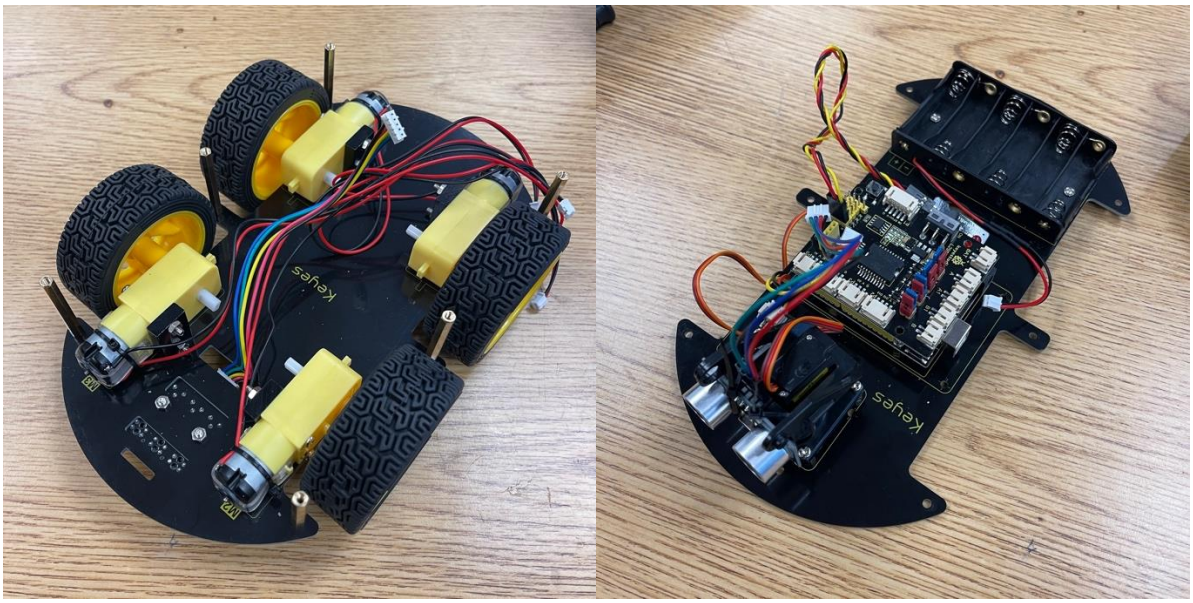

Figure 0.1 Bottom part (left) and top part (right) of the smart car

PROJECTS

Project 1: LED light

For the first project, we are required to test work on LED which is the fundamental for programming. This will also test if the Arduino microcontroller is working as well. Below is the diagram on how to hook up the LED light for testing.

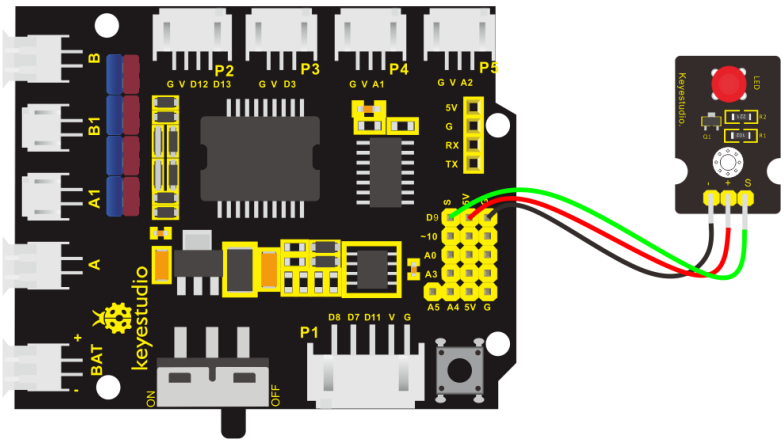

Figure 1.1. Hook up diagram for LED light testing

After hooking up, I was able to write the code for this project. Below is the complete code as well as the output for this project.

```
/*    Project 01: LED Light    */
void setup() {
  pinMode(9, OUTPUT);   // initialize digital pin 3 as an output.
}
void loop() {             // the loop function runs over and over again forever
  digitalWrite(9, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000);          // wait for 1 second
  digitalWrite(9, LOW);  // turn the LED off
  delay(1000);          // wait for 1 second
}
```

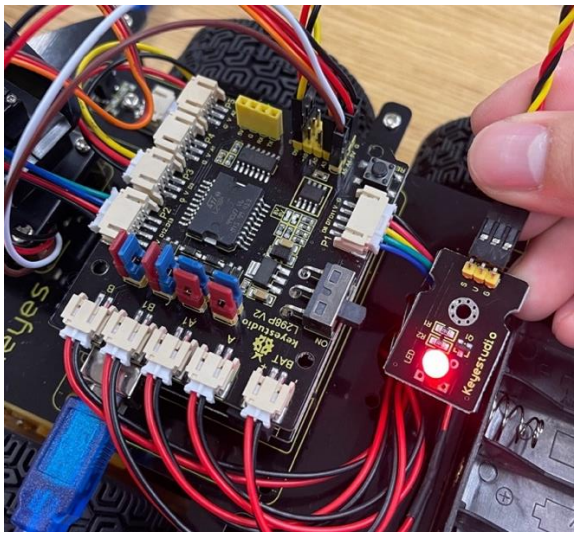

Figure 1.2. Actual test for the LED light

Project 2: Adjust LED Brightness

For this project, I was tasked to adjust the LED brightness from the previous project. I will be controlling the brightness of the LED through PWM, which is a means of controlling the analog output via digital means, to simulate breathing effects which similarly can change the step length and delay time in the code. Below is the code used for this project.

```
/*   Project 02: PWM   */

int ledPin = 9; // Define the LED pin at D9
int value;

void setup () {
  pinMode (ledPin, OUTPUT); // initialize ledpin as an output.
}

void loop () {
  for (value = 0; value <255; value = value + 1) {
    analogWrite (ledPin, value); // LED lights gradually light up
    delay (5); // delay 5MS
  }
  for (value = 255; value> 0; value = value-1) {
    analogWrite (ledPin, value); // LED gradually goes out
    delay (5); // delay 5MS
  }
}
```

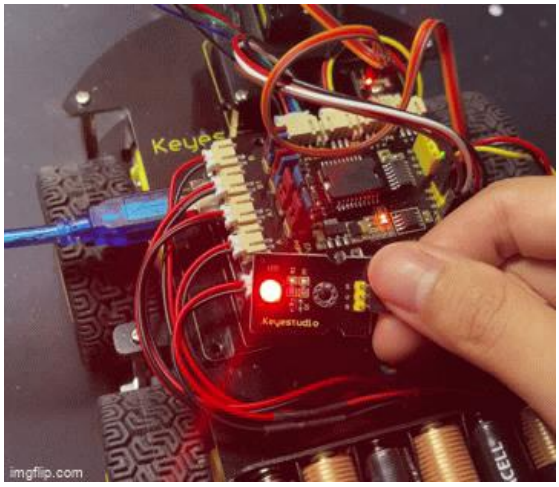

Figure 2.1. Actual test for LED Brightness blinking

Project 3: Servo Control

For this project, I will be testing the Servo motor which is a position control rotary actuator. This will help to rotate the ultrasonic sensor to detect any surrounding objects. Generally, the angle range of servo rotation is 0 to 180 degrees. It is controlled by regulating the duty cycle of PWM (Pulse-Width Modulation) signal. Below is the hook diagram for servo motor.

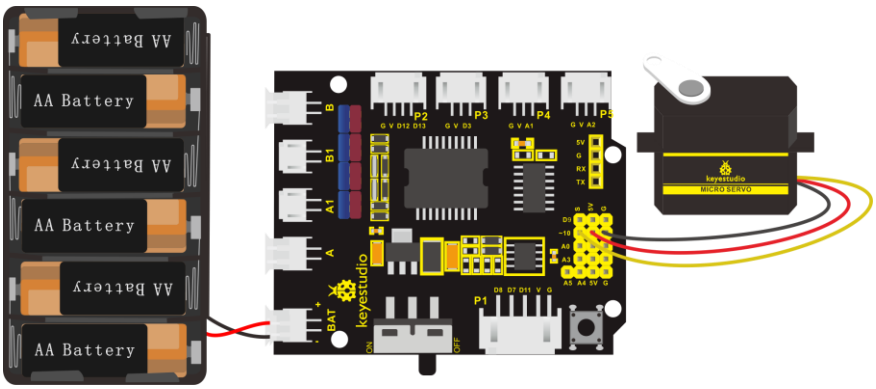

Figure 3.1. Hook up diagram for Servo motor testing

After hooking up the servo motor, we will be moving on writing the code to control the servo motor that will rotate from 0 to 180 degrees slowly. Below is the full code and demonstration.

```
/*   Project 03: Servo testing   */
#define servoPin 10  //servo Pin
int pos;           //the angle variable of servo
int pulsewidth;     // pulse width variable of servo

void setup() {
  pinMode(servoPin, OUTPUT);  // set the pins of servo to output
  procedure(0);               // set the angle of servo to 0 degree
}

void loop() {
  for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees
    // in steps of 1 degree
    procedure(pos);             // tell servo to go to position in variable 'pos'
    delay(15);                  //control the rotation speed of servo
  }
  for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees
    procedure(pos);             // tell servo to go to position in variable 'pos'
    delay(15);
  }
}

// function to control servo
void procedure(int myangle) {
  pulsewidth = myangle * 11 + 500;  //calculate the value of pulse width
  digitalWrite(servoPin,HIGH);
  delayMicroseconds(pulsewidth);   //The duration of high level is pulse width
  digitalWrite(servoPin,LOW);
  delay((20 - pulsewidth / 1000));  // the cycle is 20ms, the low level last for the rest of time
}
```

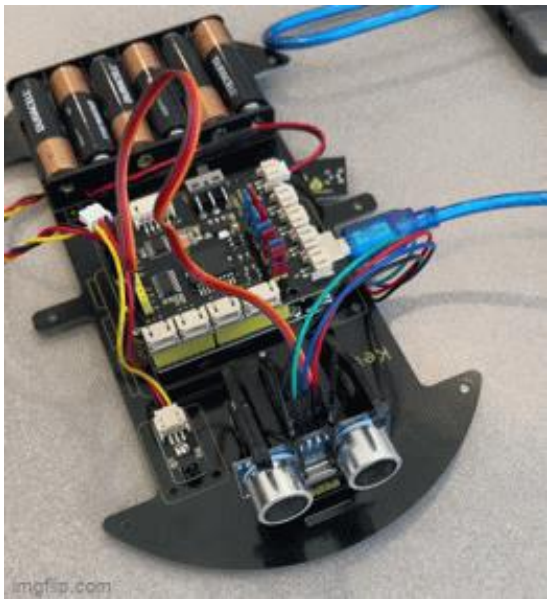

Figure 3.2. Servo Motor testing 180 degrees

Project 4: Ultrasonic Sensor

For this project, I will be now testing the Ultrasonic sensor. The HC-SR04 ultrasonic sensor uses sonar to determine distance to an object like what bats do. It is being used in a wide range of electronics projects for creating obstacle detection and distance measuring application as well as various other applications. Below is the full hook up diagram for this project.

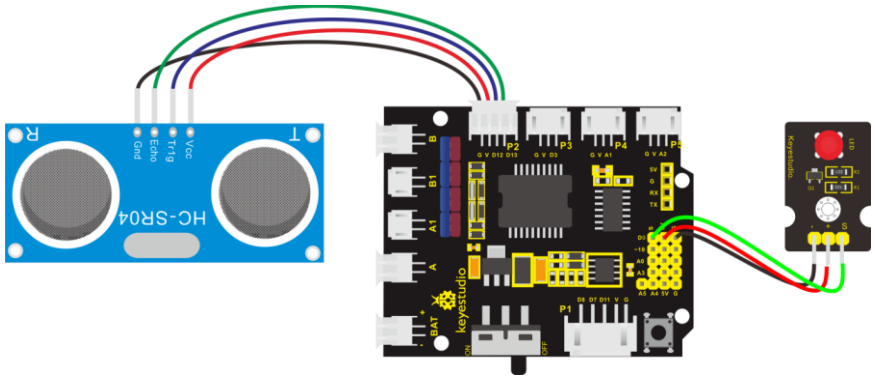

Figure 4.1. Hook up diagram for Ultrasonic Sensor testing

```
/*  Project 04: Ultrasonic Sensor  */
int trigPin = 12;    // Trigger
int echoPin = 13;    // Echo
long duration, cm, inches;
void setup() {
  Serial.begin (9600);  //Serial Port begin
  pinMode(trigPin, OUTPUT);  //Define inputs and outputs
  pinMode(echoPin, INPUT);
  pinMode(9, OUTPUT);
}

void loop() {
  // The sensor is triggered by a HIGH pulse of 10 or more microseconds.
  // Give a short LOW pulse beforehand to ensure a clean HIGH pulse:
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  // Read the signal from the sensor: a HIGH pulse whose
  // duration is the time (in microseconds) from the sending
  // of the ping to the reception of its echo off of an object.
  duration = pulseIn(echoPin, HIGH);
  // Convert the time into a distance
```

```
  cm = (duration/2) / 29.1;     // Divide by 29.1 or multiply by 0.0343
  inches = (duration/2) / 74;   // Divide by 74 or multiply by 0.0135
  Serial.print(inches);
  Serial.print("in, ");
  Serial.print(cm);
  Serial.print("cm");
  Serial.println();
  delay(50);
if (cm>=2 && cm<=10)
digitalWrite(9, HIGH);
else digitalWrite(9, LOW);
}//*************************************************************
```

```
Output   Serial Monitor ×
Message (Enter to send message to 'Arduino Uno' on '/dev/
10:55:36.417 -> 315in, 801cm
10:55:36.549 -> 315in, 801cm
10:55:36.647 -> 315in, 801cm
10:55:36.678 -> 6in, 15cm
10:55:36.777 -> 314in, 800cm
10:55:36.842 -> 1in, 4cm
10:55:36.908 -> 1in, 4cm
10:55:36.939 -> 6in, 16cm
10:55:37.006 -> 6in, 16cm
10:55:37.104 -> 315in, 801cm
10:55:37.202 -> 314in, 800cm
10:55:37.301 -> 315in, 801cm
10:55:37.399 -> 315in, 801cm
10:55:37.498 -> 314in, 800cm
10:55:37.596 -> 315in, 801cm
10:55:37.695 -> 314in, 800cm
```

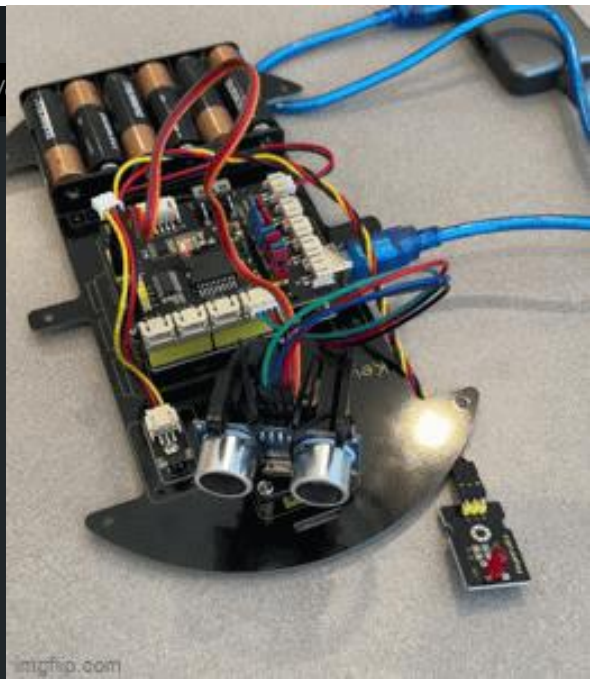

Figure 4.2. detection distance in inches and cm (left) and the actual testing with LED (right)

Project 5: Line Tracking Sensor

For this part, I tested the line tracking sensor. The tracking sensor is an infrared sensor which the component used is the TCRT5000 infrared tube. During the process of detection, back is active at HIGH level while white is active at LOW level which the detection height is 0-3 cm. The line tracking module has integrated 3 sets of TCRT5000 infrared tube on a single board,

which is more convenient for wiring and control. Below is the diagram of the line tracking sensor.

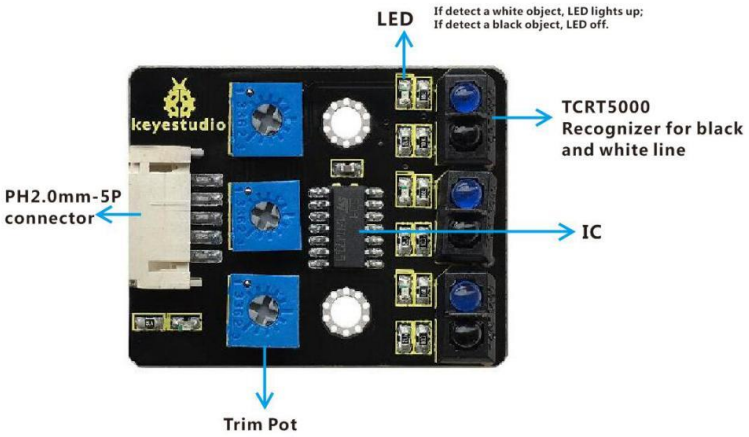

Figure 5.1. Diagram of line tracking sensor

After investigating and learning about the line tracking sensor, I then hook up the sensor including with the LED light. Below is the full diagram for this project.

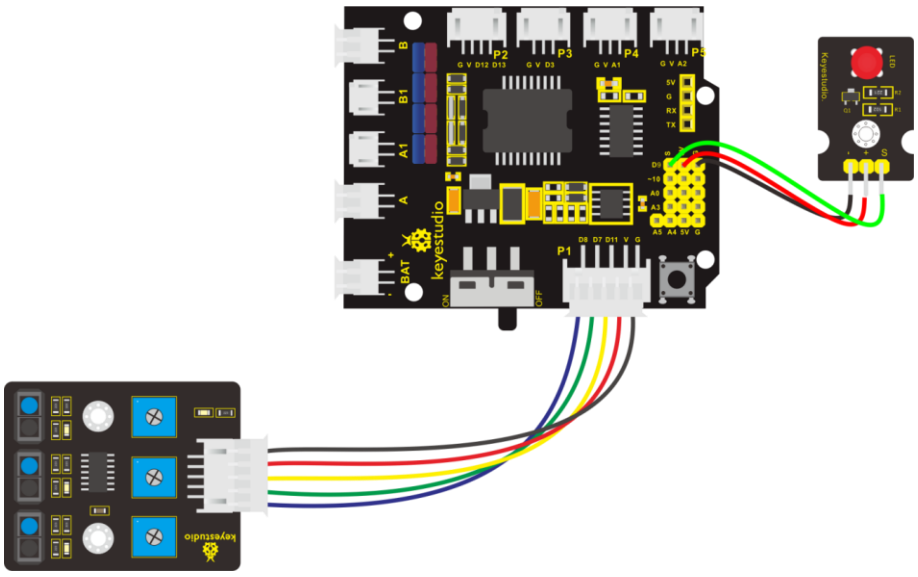

Figure 5.2. Hook up diagram for Line Tracking Sensor testing

```
/*   Project 05: Line Tracking Sensor   */

int L_pin = 11;       // pins of left line tracking sensor
int M_pin = 7;        // pins of middle line tracking sensor
int R_pin = 8;        // pins of right line tracking sensor
```

```
int val_L,val_R,val_M;  // define the variables of three sensors

void setup() {
  Serial.begin(9600);   // initialize serial communication at 9600 bits per second
  pinMode(L_pin,INPUT); // make the L_pin as an input
  pinMode(M_pin,INPUT); // make the M_pin as an input
  pinMode(R_pin,INPUT); // make the R_pin as an input
  pinMode(9, OUTPUT);
}

void loop() {
  val_L = digitalRead(L_pin); //read the L_pin:
  val_R = digitalRead(R_pin); //read the R_pin:
  val_M = digitalRead(M_pin); //read the M_pin:
  Serial.print("left:");
  Serial.print(val_L);
  Serial.print(" middle:");
  Serial.print(val_M);
  Serial.print(" right:");
  Serial.println(val_R);

  if (val_L == HIGH) {     //if left line tracking sensor detects signals
    digitalWrite(9, LOW);  //LED is off
  } else {                 //if left line tracking sensor doesn't detect signals
    digitalWrite(9, HIGH); //LED lights up
    delay(2000);
  }

  if (val_R == HIGH) {     //if right line tracking sensor detects signals
    digitalWrite(9, LOW);  //LED is off
  } else {                 //if right line tracking sensor doesn't detect signals
    digitalWrite(9, HIGH); //LED lights up
    delay(2000);
  }

  if (val_M == HIGH) {     //if middle line tracking sensor detects signals
    digitalWrite(9, LOW);  //LED is off
  } else {                 //if middle line tracking sensor doesn't detect signals
    digitalWrite(9, HIGH); //LED lights up
    delay(2000);
  }
}
```

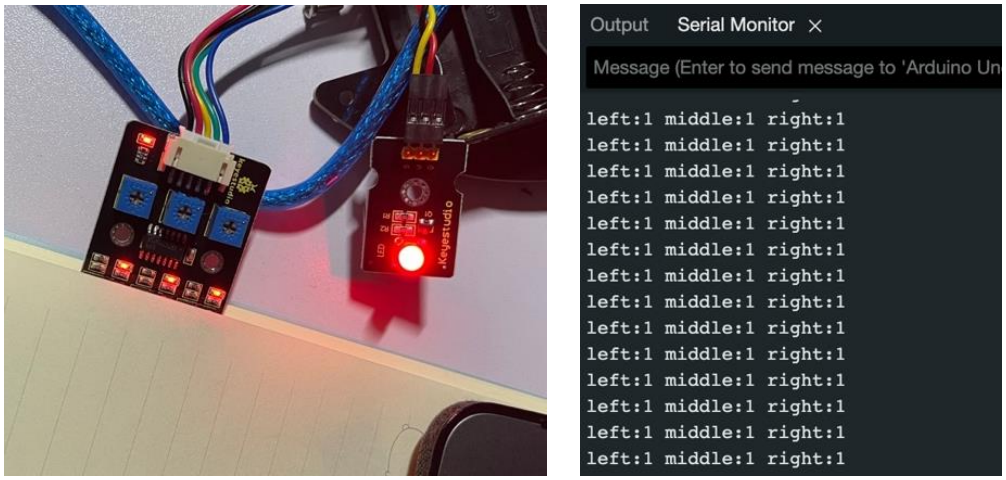

Figure 5.3. Actual testing of Line Tracking Sensor (left) and the output monitor (right)

Project 6: IR Reception

For this project, I will be testing the IR Reception with the remote control from this laboratory kit. Infrared remote control is composed of infrared transmitting and infrared receiving systems, that is, an infrared remote control and infrared receiving module and a single-chip microcomputer capable of decoding. When a remote control button is pressed, it sends out an infrared carried signal. When the IR receiver receives the signal, the program will decode the carrier signal and determines which key is pressed.

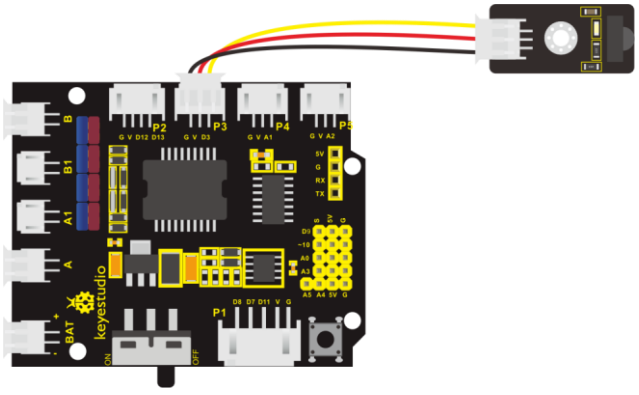

Figure 6.1. Hook up diagram for IR reception

```
/*    Project 06: IRremote    */
#include <IRremote.h>     // IRremote library statement
int RECV_PIN = 3;        //define the pins of IR receiver as3
IRrecv irrecv(RECV_PIN);
decode_results results;   // decode results exist in the“result” of “decode results”
void setup() {
      Serial.begin(9600);
      irrecv.enableIRIn(); // Enable receiver
}

void loop() {
  if (irrecv.decode(&results)) { //decode successfully, receive a set of infrared signals
    Serial.println(results.value, HEX);//Wrap word in 16 HEX to output and receive code
    irrecv.resume(); // Receive the next value
  }
  delay(100);
}
```

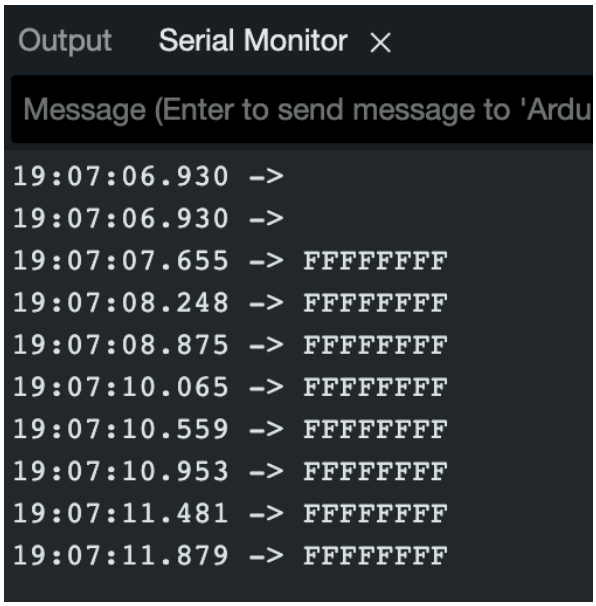

Figure 6.2. Output for pressing the remote control

Project 7: Bluetooth Remote Control

For this project, I will be testing the Bluetooth remote control through my phone. We will be learning about HM-10 BLE 4.0 with Arduino Board in which the HM-10 is a readily available Bluetooth 4.0 module that is used for establishing wireless data communication. It is designed by

using the Texas Instruments CC2540 or CC2541 Bluetooth low energy (BLE) System on Chip (SoC). Below is the full hook up diagram to connect the Bluetooth remote control.

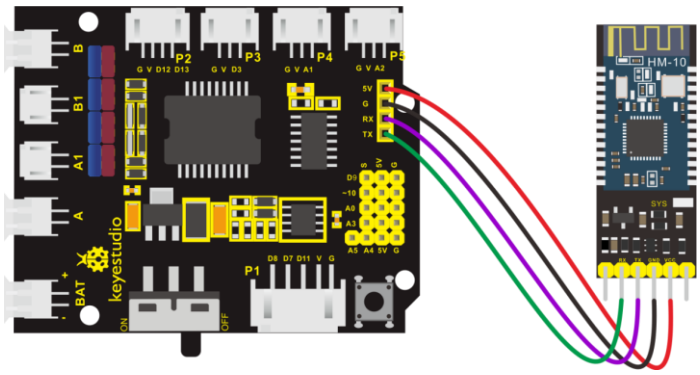

Figure 7.1. Hook up diagram for Bluetooth remote control

```
/*    Project 07: Bluetooth   */
char ble_val; //character variable, used to store the value received by Bluetooth

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

void loop() {
  if(Serial.available() > 0) { //make sure if there is data in serial buffer
    ble_val = Serial.read();  //Read data from serial buffer
    Serial.println(ble_val);  //Print
  }
}
```

After writing the code for this project, I am required to install a software called BLE Scanner 4.0 to scan any Bluetooth module, in which this one is called HMSoft. Then, I need to click Read,Notify,WriteWithoutResponse page and write a Value to enter in HEX or Text. When I enter 1, LED will be on; when I enter 0, it will be off.

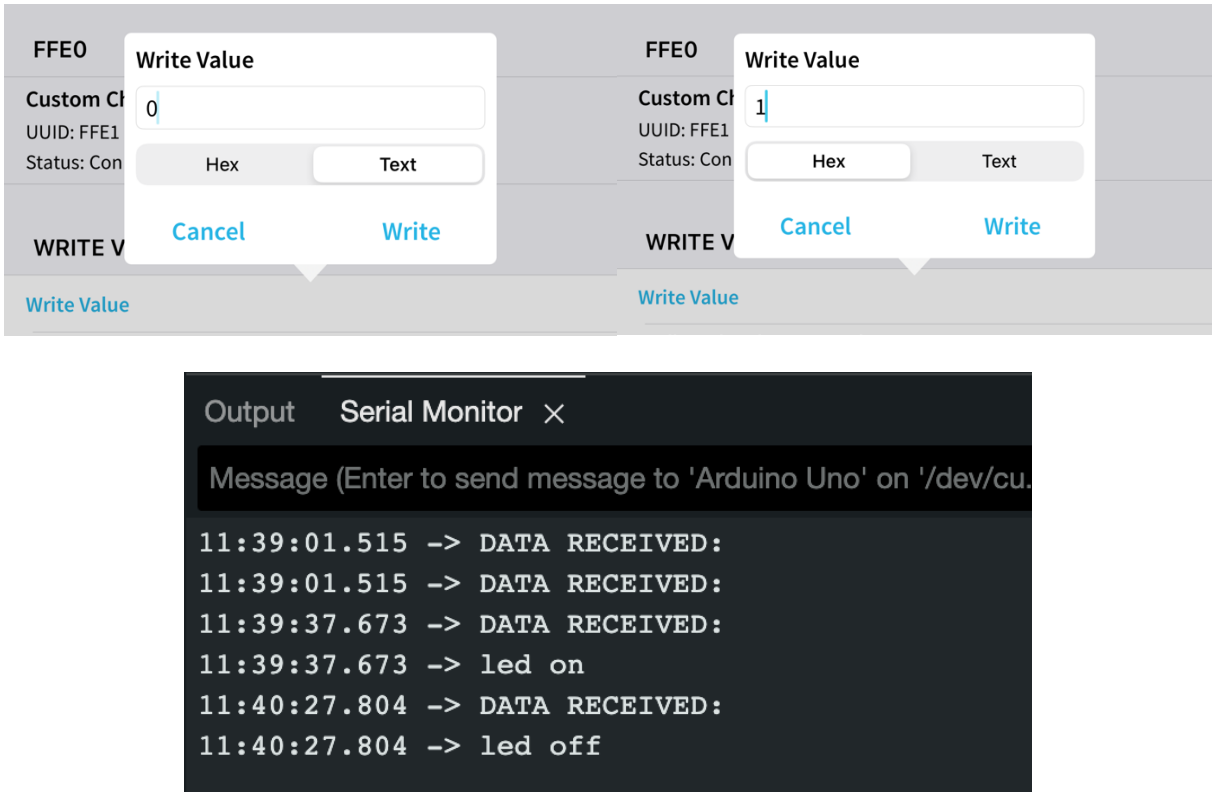

Figure 7.2. Output for changing the Value on the APP.

Project 8: Motor Driving and Speed Control

For this project, I will be now testing the motor for the wheels of our robot car. Below is the hook up diagram for the motor as well as the full code for testing.

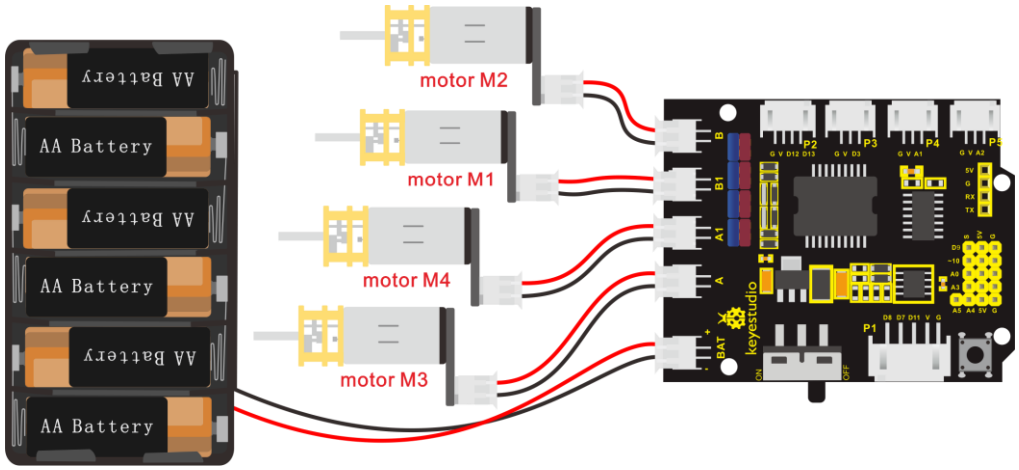

Figure 8.1. Hook up diagram for the motor

```
/*    Project 8: Motor    */
#define ML_Ctrl 4     // define the direction control pin of B motor
#define ML_PWM 5   //define the PWM control pin of B motor
#define MR_Ctrl 2    //define direction control pin of A motor
#define MR_PWM 6   //define the PWM control pin of A motor
void setup() {
  pinMode(ML_Ctrl, OUTPUT);//set direction control pin of B motor to output
  pinMode(ML_PWM, OUTPUT);//set PWM control pin of B motor to output
  pinMode(MR_Ctrl, OUTPUT);//set direction control pin of A motor to output.
  pinMode(MR_PWM, OUTPUT);//set the PWM control pin of A motor to output
}
void loop() {
  digitalWrite(ML_Ctrl,HIGH);//set the direction control pin of B motor to HIGH
  analogWrite(ML_PWM,200);//set the PWM control speed of B motor to 200
  digitalWrite(MR_Ctrl,HIGH);//set the direction control pin of A motor to HIGH
  analogWrite(MR_PWM,200);//set the PWM control speed of A motor to 200

   //front
  delay(2000);//delay in 2s
  digitalWrite(ML_Ctrl,LOW);//set the direction control pin of B motor to LOW
  analogWrite(ML_PWM,200);//set the PWM control speed of B motor to 200
  digitalWrite(MR_Ctrl,LOW);//set the direction control pin of A motor to LOW
  analogWrite(MR_PWM,200);//set the PWM control speed of A motor to 200
   //back
  delay(2000);//delay in 2s
  digitalWrite(ML_Ctrl,LOW);//set the direction control pin of B motor to LOW
  analogWrite(ML_PWM,200);//set the PWM control speed of B motor to 200
  digitalWrite(MR_Ctrl,HIGH);//set the direction control pin of A motor to HIGH
  analogWrite(MR_PWM,200);// set the PWM control speed of A motor to 200

    //left
  delay(2000);//delay in 2s
  digitalWrite(ML_Ctrl,HIGH);//set the direction control pin of B motor to HIGH
  analogWrite(ML_PWM,200);//set the PWM control speed of B motor to 200
  digitalWrite(MR_Ctrl,LOW);// set the direction control pin of A motor to LOW
  analogWrite(MR_PWM,200);//set the PWM control speed of A motor to 200

   //right
  delay(2000);//delay in 2s
  analogWrite(ML_PWM,0);//set the PWM control speed of B motor to 0
  analogWrite(MR_PWM,0);//set the PWM control speed of A motor to 0

    //stop
  delay(2000);//delay in 2s
}
```

The program will generate the smart car to goes forward and back for 2s, turns left and right for 2s, and stops for 2s alternately. Below is the snippet of the actual movement of the wheels.

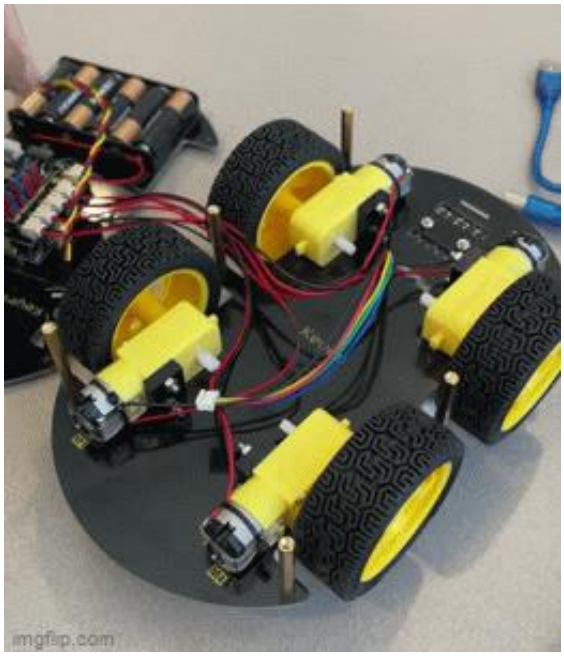

Figure 8.2. Actual testing of the motor wheels

Project 9: 8*16 LED Board

For this project, I will be showing how to use the LED board that can create facial emoticons, patterns, or other interesting displays. 8*16 LED light board comes with 128 LEDs. Using a recommended online version of dot matrix modulus tool, I designed what I want to output for me LED Board. It needs to set the height to 8, width to 16, and Little Endian shown below.

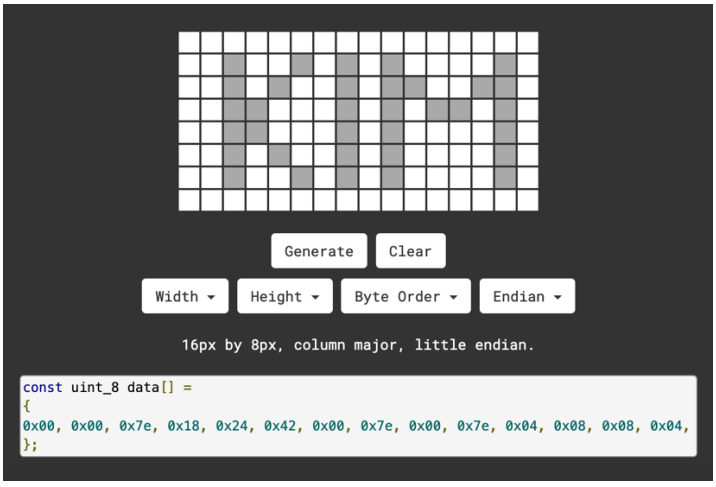

Figure 9.1. Snippet of the online version matrix modulus tool

After that, we need to hook up the device needed for this project and add the generated code from the matrix modulus tool to our program.

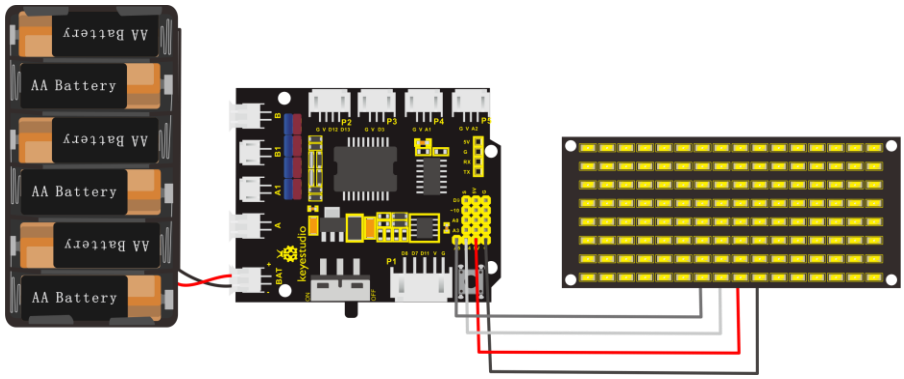

Figure 9.2. Hook up diagram for the 8*16 LED Board

```
/*    Project 9: matrix   */
const char data[] = {0x00, 0x00, 0x7e, 0x18, 0x24, 0x42, 0x00, 0x7e, 0x00, 0x7e, 0x04, 0x08, 0x08, 0x04, 0x7e,
0x00 };
#define SCL_Pin  A5  //Set clock pin to A5
#define SDA_Pin  A4  //Set data pin to A4
void setup() {
  //Set pin to output
  pinMode(SCL_Pin,OUTPUT);
  pinMode(SDA_Pin,OUTPUT);
  //Clear the matrix display
  //matrix_display(clear);
}
void loop() {
  matrix_display(data);  //display the pattern
}
//this function is used for dot matrix display
void matrix_display(unsigned char matrix_value[]) {
  IIC_start();  //the function to call the data transmission
  IIC_send(0xc0);  //Select address

  for(int i = 0;i < 16;i++) //Pattern data has 16 bytes
    IIC_send(matrix_value[i]); //data to convey patterns
  IIC_end();   //end the transmission of patterns data
  IIC_start();
  IIC_send(0x8A);  //display control, set pulse width to 4/16
  IIC_end();
}
//  the condition that data transmission starts
void IIC_start() {
  digitalWrite(SCL_Pin,HIGH);
  delayMicroseconds(3);
```

```
  digitalWrite(SDA_Pin,HIGH);
  delayMicroseconds(3);
  digitalWrite(SDA_Pin,LOW);
  delayMicroseconds(3);
}
// transmit data
void IIC_send(unsigned char send_data) {
  for(char i = 0;i < 8;i++) { //Every character has 8 bits
      digitalWrite(SCL_Pin,LOW);  //pull down the SCL_Pin to change the signal of SDA
      delayMicroseconds(3);
      if(send_data & 0x01)  //1 or 0 of byte  is used to set high and low level of SDA_Pin
        digitalWrite(SDA_Pin,HIGH);
      else
        digitalWrite(SDA_Pin,LOW);
      delayMicroseconds(3);
      digitalWrite(SCL_Pin,HIGH); //Pull up SCL_Pin to stop data transmission
      delayMicroseconds(3);
      send_data = send_data >> 1;  //Detect bit by bit, so move the data right by one bit
  }
}
//the sign that data transmission ends
void IIC_end() {
  digitalWrite(SCL_Pin,LOW);
  delayMicroseconds(3);
  digitalWrite(SDA_Pin,LOW);
  delayMicroseconds(3);
  digitalWrite(SCL_Pin,HIGH);
  delayMicroseconds(3);
  digitalWrite(SDA_Pin,HIGH);
  delayMicroseconds(3);
}
```

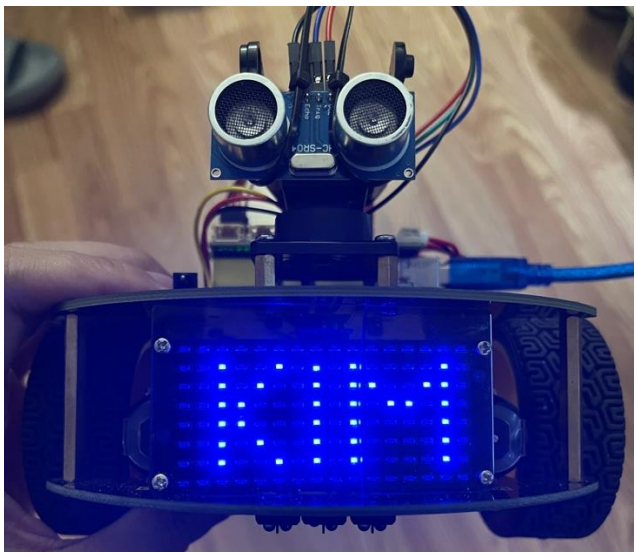

Figure 9.3. Actual output of the matrix to the LED board

Project 10: Line Tracking Smart Car

For this project, I will be working on Line Tracking Smart Car. The goal of this project is to design a line track that lets the smart car follows the line. This will focus on the working principle of the line tracking sensor for the smart car.

```
/*   Project 10: Line Tracking Smart Car   */
#define ML_Ctrl 4     //define direction control pin of B motor
#define ML_PWM 5   //define PWM control pin of B motor
#define MR_Ctrl 2    //define direction control pin of A motor
#define MR_PWM 6   //define PWM control pin of A motor
const int sensor_l = 11;//define the pin of left line tracking sensor
const int sensor_c = 7;//define the pin of middle line tracking sensor
const int sensor_r = 8;//define the pin of right line tracking sensor
int l_val,c_val,r_val;//define these variables

void setup() {
  Serial.begin(9600);//start serial monitor and set baud rate to 9600
  pinMode(ML_Ctrl, OUTPUT);//set direction control pin of B motor to OUTPUT
  pinMode(ML_PWM, OUTPUT);//set PWM control pin of B motor to OUTPUT
  pinMode(MR_Ctrl, OUTPUT);//set direction control pin of A motor to OUTPUT
  pinMode(MR_PWM, OUTPUT);//set PWM control pin of A motor to OUTPUT
  pinMode(sensor_l,INPUT);//set the pins of left line tracking sensor to INPUT
  pinMode(sensor_c,INPUT);//set the pins of middle line tracking sensor to INPUT
  pinMode(sensor_r,INPUT);//set the pins of right line tracking sensor to INPUT
}

void loop() {
  tracking(); //run main program
}

void tracking() {
  l_val = digitalRead(sensor_l);//read the value of left line tracking sensor
  c_val = digitalRead(sensor_c);//read the value of middle line tracking sensor
  r_val = digitalRead(sensor_r);//read the value of right line tracking sensor
  if(c_val == 1) { //if the state of middle one is 1, which means detecting black line
   front();    //car goes forward
  }
  else {
   if((l_val==1)&&(r_val==0))//if only left line tracking sensor detects black trace
      left();     //car turns left
   elseif((l_val==0)&&(r_val==1))//if only right line tracking sensor detects black trace
      right();    //car turns right
   else          // if line tracking sensors detect black trace or they don't
      Stop();     //car stops
  }
}
void front() { //define the status of going forward
  digitalWrite(ML_Ctrl,HIGH);//set direction control pin of B motor to HIGH
  analogWrite(ML_PWM,70);//set PWM control speed of B motor to 70
```

```
  digitalWrite(MR_Ctrl,HIGH);//set direction control pin of A motor to HIGH
  analogWrite(MR_PWM,70);//set PWM control speed of A motor to 70
}
void back() { //define the state of going back
  digitalWrite(ML_Ctrl,LOW);//set direction control pin of B motor to LOW
  analogWrite(ML_PWM,200);//set PWM control speed of B motor to 200
  digitalWrite(MR_Ctrl,LOW);//set direction control pin of A motor to LOW
  analogWrite(MR_PWM,200);//set PWM control speed of A motor to 200
}
void left() { //car turns left
  digitalWrite(ML_Ctrl,LOW);//set direction control pin of B motor to LOW
  analogWrite(ML_PWM,200);//set PWM control speed of B motor to 200
  digitalWrite(MR_Ctrl,HIGH);//set direction control pin of A motor to HIGH level
  analogWrite(MR_PWM,200);//set PWM control speed of A motor to 200
}
void right() { //define the right-turning state
  digitalWrite(ML_Ctrl,HIGH);//set direction control pin of B motor to HIGH level
  analogWrite(ML_PWM,200);//set PWM control speed of B motor to 200
  digitalWrite(MR_Ctrl,LOW);//set direction control pin of A motor to LOW
  analogWrite(MR_PWM,200);//set PWM control speed of A motor to 200
}
void Stop() {//define the state of stop
  analogWrite(ML_PWM,0);//set PWM control speed of B motor to 0
  analogWrite(MR_PWM,0);//set PWM control speed of A motor to 0
}
```

Project 11: Ultrasonic Follow Robot

This project will be featuring the ultrasonic sensor and servo motor. In this process, the ultrasonic sensor will detect the distance between the robot car and obstacles so as to control the robot car to move by the measured distance. The goal is to have program with a given distance condition to go forward and stop when it closes to the object or obstacle in front.

```
/*    Project 11: Ultrasonic Follow Robot   */
#define ML_Ctrl 4     //define direction control pin of B motor
#define ML_PWM 5   //define PWM control pin of B motor
#define MR_Ctrl 2    //define direction control pin of A motor
#define MR_PWM 6   //define PWM control pin of A motor
#include "SR04.h" //define the function library of ultrasonic sensor
#define TRIG_PIN 12// set the signal input of ultrasonic sensor to D12
#define ECHO_PIN 13//set the signal output of ultrasonic sensor to D13
SR04 sr04 = SR04(ECHO_PIN,TRIG_PIN);
long distance;
void setup() {
```

```
  Serial.begin(9600);//open serial monitor and set baud rate to 9600
  pinMode(ML_Ctrl, OUTPUT);//set direction control pin of B motor to OUTPUT
  pinMode(ML_PWM, OUTPUT);//set PWM control pin of B motor to OUTPUT
  pinMode(MR_Ctrl, OUTPUT);//set direction control pin of A motor to OUTPUT
  pinMode(MR_PWM, OUTPUT);//set PWM control pin of A motor to OUTPUT
  pinMode(TRIG_PIN,OUTPUT);// set TRIG_PIN to OUTPUT
  pinMode(ECHO_PIN,INPUT);// set ECHO_PIN to INPUT
}
void loop() {
  distance = sr04.Distance();// the distance detected by ultrasonic sensor
   if(distance<8)//if distance is less than 8
    back();//go back
  else if((distance>=8)&&(distance<13))// if 8≤distance＜13
    Stop();//stop
  else if((distance>=13)&&(distance<35))//if 13≤distance＜35
    front();//follow
  else//otherwise
    Stop();//stop
}

void front() { //go front
  digitalWrite(ML_Ctrl,HIGH);//set direction control pin of B motor to HIGH
  analogWrite(ML_PWM,100);//Set PWM control speed of B motor to 100
  digitalWrite(MR_Ctrl,HIGH);//set direction control pin of A motor to HIGH
  analogWrite(MR_PWM,100);//Set PWM control speed of A motor to 100
}
void back() { //go back
  digitalWrite(ML_Ctrl,LOW);//set direction control pin of B motor to LOW
  analogWrite(ML_PWM,100);//Set PWM control speed of B motor to 100
  digitalWrite(MR_Ctrl,LOW);//set direction control pin of A motor to LOW
  analogWrite(MR_PWM,100);//Set PWM control speed of A motor to 100
}
void Stop() {   //stop
  analogWrite(ML_PWM,0);//set PWM control speed of B motor to 0
  analogWrite(MR_PWM,0);//set PWM control speed of A motor to 0
}
```

Figure 11.1. Actual robot car following an object

Project 12: Obstacle Avoidance Smart Car

Since we were able to follow an object from the last project, we will now program our smart robot car to avoid obstacle using the same principles and features of the ultrasonic sensor.

```
/*    Project 12: ultrasonic avoiding robot   */
#define SCL_Pin  A5  //Set clock pin to A5
#define SDA_Pin  A4  //Set data pin to A4
#define ML_Ctrl 4     //define direction control pin of B motor
#define ML_PWM 5   //define PWM control pin of B motor
#define MR_Ctrl 2    //define direction control pin of A motor
#define MR_PWM 6   //define PWM control pin of A motor
#include "SR04.h"//define the library of ultrasonic sensor
#define TRIG_PIN 12// set the signal input of ultrasonic sensor to D12
#define ECHO_PIN 13//set the signal output of ultrasonic sensor to D13
SR04 sr04 = SR04(ECHO_PIN,TRIG_PIN);
long distance,a1,a2;//define three distance
const int servopin = 10;//set the pin of servo to D10

void setup() {
  Serial.begin(9600);//open serial monitor and set baud rate to 9600
  pinMode(ML_Ctrl, OUTPUT);//set direction control pin of B motor to OUTPUT
  pinMode(ML_PWM, OUTPUT);//set PWM control pin of B motor to OUTPUT
  pinMode(MR_Ctrl, OUTPUT);//set direction control pin of A motor to OUTPUT

  pinMode(MR_PWM, OUTPUT);//set PWM control pin of A motor to OUTPUT
  servopulse(servopin,90);// the angle of servo is 90 degree
  delay(300);
  pinMode(SCL_Pin,OUTPUT);//  set clock pin to OUTPUT
  pinMode(SDA_Pin,OUTPUT);//set data pin to OUTPUT
}

void loop() {
  avoid();//run the main program
}
void avoid() {
  distance=sr04.Distance(); //obtain the value detected by ultrasonic sensor
  if((distance < 20)&&(distance > 0)) { //if the distance is greater than 0 and less than 20
    car_Stop();//stop
    delay(100);
    servopulse(servopin,180);//servo rotates to 180°
    delay(500);
    a1=sr04.Distance();//measure the distance
```

```
    delay(100);
    servopulse(servopin,0);//rotate to 0 degree
    delay(500);
    a2=sr04.Distance();//measure the distance
    delay(100);
    if(a1 > a2) { //if distance a1 is greater than a2
      car_left();//turn left
      servopulse(servopin,90);//servo rotates to 90 degree
      delay(300);
    }
    else { //if the right distance is greater than the left
      car_right();// turn right
      servopulse(servopin,90);// servo rotates to 90 degree
      delay(300);
    }
  }
  else {
    car_front(); //go forward
  }
}
void servopulse(int servopin,int myangle) {//the running angle of servo
  for(int i=0; i<30; i++) {
    int pulsewidth = (myangle*11)+500;
    digitalWrite(servopin,HIGH);
    delayMicroseconds(pulsewidth);
    digitalWrite(servopin,LOW);
    delay(20-pulsewidth/1000);
  }
}
void car_front() { //car goes forward
  digitalWrite(ML_Ctrl,HIGH);//set direction control pin of B motor to HIGH level
  analogWrite(ML_PWM,150);//set PWM control speed of B motor to 150
  digitalWrite(MR_Ctrl,HIGH);//set direction control pin of A motor to HIGH level
  analogWrite(MR_PWM,150);//set PWM control speed of A motor to 150
}
void car_back() {//go back
  digitalWrite(ML_Ctrl,LOW);//set direction control pin of B motor to LOW
  analogWrite(ML_PWM,200);//set PWM control speed of B motor to 200
  digitalWrite(MR_Ctrl,LOW);//set direction control pin of A motor to LOW
  analogWrite(MR_PWM,200);//set PWM control speed of A motor to 200
}
void car_left() {//car turns left
  digitalWrite(ML_Ctrl,LOW);//set direction control pin of B motor to LOW
  analogWrite(ML_PWM,200);//set PWM control speed of B motor to 200
  digitalWrite(MR_Ctrl,HIGH);//set direction control pin of A motor to HIGH
  analogWrite(MR_PWM,200);//set PWM control speed of A motor to 200
}
void car_right() {//car turns right
  digitalWrite(ML_Ctrl,HIGH);//set direction control pin of B motor to HIGH
  analogWrite(ML_PWM,200);//set PWM control speed of B motor to 200
  digitalWrite(MR_Ctrl,LOW);//set direction control pin of A motor to LOW
  analogWrite(MR_PWM,200);//set PWM control speed of A motor to 200
}
```

```
void car_Stop() {//car stops
  digitalWrite(ML_Ctrl,LOW);
  analogWrite(ML_PWM,150);
  digitalWrite(MR_Ctrl,LOW);
  analogWrite(MR_PWM,150);
  delay(50);
  analogWrite(ML_PWM,0);//set PWM control speed of B motor to 0
  analogWrite(MR_PWM,0);//set PWM control speed of A motor to 0
}

//  the condition that data transmission starts
void IIC_start() {
  digitalWrite(SCL_Pin,HIGH);
  delayMicroseconds(3);
  digitalWrite(SDA_Pin,HIGH);
  delayMicroseconds(3);
  digitalWrite(SDA_Pin,LOW);
  delayMicroseconds(3);
}

// transmit data
void IIC_send(unsigned char send_data) {
  for(char i = 0;i < 8;i++) {  //Every character has 8 bits
    digitalWrite(SCL_Pin,LOW);  //pull down the SCL_Pin to change the signal of SDA
    delayMicroseconds(3);
    if(send_data & 0x01) { //1 or 0 of byte  is used to set high and low level of SDA_Pin
      digitalWrite(SDA_Pin,HIGH);
    }
    else {
      digitalWrite(SDA_Pin,LOW);
    }
    delayMicroseconds(3);
    digitalWrite(SCL_Pin,HIGH); //Pull up SCL_Pin to stop data transmission
    delayMicroseconds(3);
    send_data = send_data >> 1;  //Detect bit by bit, so move the data right by one bit
  }
}
//the sign that data transmission ends
void IIC_end() {
  digitalWrite(SCL_Pin,LOW);
  delayMicroseconds(3);
  digitalWrite(SDA_Pin,LOW);
  delayMicroseconds(3);
  digitalWrite(SCL_Pin,HIGH);
  delayMicroseconds(3);
  digitalWrite(SDA_Pin,HIGH);
  delayMicroseconds(3);
}
```

Figure 12.1. Actual robot car avoiding obstacles

Project 13: IR Remote Smart Car

For this project, we will be using the remote control to control our smart car. Pressing the button on IR remote control will move the robot car and also show the corresponding state pattern display for the special feature.

```
/*    Project 14: IR Remote Control   */
//Array, used to store the data of pattern, can be calculated by yourself or obtained from the modulus tool
unsigned char start01[] =
{0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80,0x80,0x40,0x20,0x10,0x08,0x04,0x02,0x01};
unsigned char front[] = {0x00,0x00,0x00,0x00,0x00,0x24,0x12,0x09,0x12,0x24,0x00,0x00,0x00,0x00,0x00,0x00};
unsigned char back[] = {0x00,0x00,0x00,0x00,0x00,0x24,0x48,0x90,0x48,0x24,0x00,0x00,0x00,0x00,0x00,0x00};
unsigned char left[] = {0x00,0x00,0x00,0x00,0x00,0x00,0x44,0x28,0x10,0x44,0x28,0x10,0x44,0x28,0x10,0x00};
unsigned char right[] = {0x00,0x10,0x28,0x44,0x10,0x28,0x44,0x10,0x28,0x44,0x00,0x00,0x00,0x00,0x00,0x00};
unsigned char STOP01[] =
{0x2E,0x2A,0x3A,0x00,0x02,0x3E,0x02,0x00,0x3E,0x22,0x3E,0x00,0x3E,0x0A,0x0E,0x00};
unsigned char clear[] = {0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00};
#define SCL_Pin  A5  //Set clock pin to A5
#define SDA_Pin  A4  //Set data pin to A4
#define ML_Ctrl 4     //define direction control pin of B motor
#define ML_PWM 5   //define PWM control pin of B motor
#define MR_Ctrl 2    //define direction control pin of A motor
#define MR_PWM 6   //define PWM control pin of A motor
#include <IRremote.h>//function library of IR remote control
int RECV_PIN =3;// set the pin of IR receiver to 3
IRrecv irrecv(RECV_PIN);
```

```
long irr_val;
decode_results results;
void setup() {
  pinMode(ML_Ctrl, OUTPUT);//define direction control pin of B motor to OUTPUT
  pinMode(ML_PWM, OUTPUT);//define PWM control pin of B motor to OUTPUT
  pinMode(MR_Ctrl, OUTPUT);//define direction control pin of A motor to OUTPUT
  pinMode(MR_PWM, OUTPUT);//define PWM control pin of A motor to OUTPUT
    Serial.begin(9600);//Start serial printing, baud rate is 9600
  // In case the interrupt driver crashes on setup, give a clue
  // to the user what's going on.
  irrecv.enableIRIn(); // Start the receiver
  Serial.println("Enabled IRin");
  //Set pin to output
  pinMode(SCL_Pin,OUTPUT);
  pinMode(SDA_Pin,OUTPUT);
  //Clear the matrix display
  matrix_display(clear);
  matrix_display(start01);
}
void loop() {
  if (irrecv.decode(&results)) {
    irr_val = results.value;
    Serial.println(irr_val, HEX);//serial reads the IR remote signals
    switch(irr_val) {
      case 0xFF629D : car_front(); matrix_display(front); break;
      case 0xFFA857 : car_back(); matrix_display(back); break;
      case 0xFF22DD : car_left(); matrix_display(left); break;
      case 0xFFC23D : car_right(); matrix_display(right); break;
      case 0xFF02FD : car_Stop(); matrix_display(STOP01); break;
    }
        irrecv.resume(); // Receive the next value
  }
}
void car_front() {//car goes forward
  digitalWrite(ML_Ctrl,HIGH);//set direction control pin of B motor to HIGH level
  analogWrite(ML_PWM,200);//Set PWM control speed of B motor to 20
  digitalWrite(MR_Ctrl,HIGH);//set direction control pin of A motor to HIGH level
  analogWrite(MR_PWM,200);//Set PWM control speed of A motor to 20
}
void car_back() { //car goes back
  digitalWrite(ML_Ctrl,LOW);//set direction control pin of B motor to LOW
  analogWrite(ML_PWM,200);//set PWM control speed of B motor to 200
  digitalWrite(MR_Ctrl,LOW);//set direction control pin of A motor to LOW
  analogWrite(MR_PWM,200);//set PWM control speed of A motor to 200
}
void car_left() { //car turns left
  digitalWrite(ML_Ctrl,LOW);//set direction control pin of B motor to LOW
  analogWrite(ML_PWM,200);//set PWM control speed of B motor to 200
  digitalWrite(MR_Ctrl,HIGH);//set direction control pin of A motor to HIGH level
  analogWrite(MR_PWM,200);//set PWM control speed of A motor to 200
}
void car_right() { //car turns right
  digitalWrite(ML_Ctrl,HIGH);//set direction control pin of B motor to HIGH level
```

```
  analogWrite(ML_PWM,200);//set PWM control speed of B motor to 200
  digitalWrite(MR_Ctrl,LOW);//set direction control pin of A motor to LOW
  analogWrite(MR_PWM,200);//set PWM control speed of A motor to 200
}
void car_Stop() { //car stops
  analogWrite(ML_PWM,0);//set PWM control speed of B motor to 0
  analogWrite(MR_PWM,0);//set PWM control speed of A motor to 0
}
//this function is used for dot matrix display
void matrix_display(unsigned char matrix_value[]) {
  IIC_start();  //the function to call the data transmission
  IIC_send(0xc0);  //Select address
    for(int i = 0;i < 16;i++) //Pattern data has 16 bytes
      IIC_send(matrix_value[i]); //data to convey patterns
  IIC_end();   //end the transmission of patterns data
  IIC_start();
  IIC_send(0x8A);  //display control, set pulse width to 4/16
  IIC_end();
}
//  the condition that data transmission starts
void IIC_start() {
  digitalWrite(SCL_Pin,HIGH);
  delayMicroseconds(3);
  digitalWrite(SDA_Pin,HIGH);
  delayMicroseconds(3);
  digitalWrite(SDA_Pin,LOW);
  delayMicroseconds(3);
}
// transmit data
void IIC_send(unsigned char send_data) {
  for(char i = 0;i < 8;i++) {  //Every character has 8 bits
      digitalWrite(SCL_Pin,LOW);  //pull down the SCL_Pin to change the signal of SDA
      delayMicroseconds(3);
      if(send_data & 0x01)  //1 or 0 of byte is used to set high and low level of SDA_Pin
        digitalWrite(SDA_Pin,HIGH);
      else
        digitalWrite(SDA_Pin,LOW);
      delayMicroseconds(3);
      digitalWrite(SCL_Pin,HIGH); //Pull up SCL_Pin to stop data transmission
      delayMicroseconds(3);
      send_data = send_data >> 1;  //Detect bit by bit, so move the data right by one bit
  }
}
//the sign that data transmission ends
void IIC_end() {
  digitalWrite(SCL_Pin,LOW);
  delayMicroseconds(3);
  digitalWrite(SDA_Pin,LOW);
  delayMicroseconds(3);
  digitalWrite(SCL_Pin,HIGH);
  delayMicroseconds(3);
  digitalWrite(SDA_Pin,HIGH);
  delayMicroseconds(3);
```

```
}//*****************************************************
```

Project 14: Bluetooth Control

For this project, we will now be making a Bluetooth remote smart car. This requires an APP called “keyes BT car” rolled out by keyestudio team that can control the robot car by it readily.

```
/*    Project 14: Bluetooth Remote Control    */
//Array, used to store the data of pattern, can be calculated by yourself or obtained from the modulus tool
unsigned char start01[] =
{0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80,0x80,0x40,0x20,0x10,0x08,0x04,0x02,0x01};
unsigned char front[] = {0x00,0x00,0x00,0x00,0x00,0x24,0x12,0x09,0x12,0x24,0x00,0x00,0x00,0x00,0x00,0x00};
unsigned char back[] = {0x00,0x00,0x00,0x00,0x00,0x24,0x48,0x90,0x48,0x24,0x00,0x00,0x00,0x00,0x00,0x00};
unsigned char left[] = {0x00,0x00,0x00,0x00,0x00,0x00,0x44,0x28,0x10,0x44,0x28,0x10,0x44,0x28,0x10,0x00};
unsigned char right[] = {0x00,0x10,0x28,0x44,0x10,0x28,0x44,0x10,0x28,0x44,0x00,0x00,0x00,0x00,0x00,0x00};
unsigned char STOP01[] =
{0x2E,0x2A,0x3A,0x00,0x02,0x3E,0x02,0x00,0x3E,0x22,0x3E,0x00,0x3E,0x0A,0x0E,0x00};
unsigned char clear[] = {0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00};
#define SCL_Pin  A5  //Set clock pin to A5
#define SDA_Pin  A4  //Set data pin to A4
unsigned char data_line = 0;
unsigned char delay_count = 0;
#define ML_Ctrl 4     //define direction control pin of B motor
#define ML_PWM 5   //define PWM control pin of B motor
#define MR_Ctrl 2    //define direction control pin of A motor
#define MR_PWM 6   //define PWM control pin of A motor
char BLE_val;
void setup() {
  Serial.begin(9600);
  pinMode(ML_Ctrl, OUTPUT);//set direction control pin of B motor to OUTPUT
  pinMode(ML_PWM, OUTPUT);//set PWM control pin of B motor to OUTPUT
  pinMode(MR_Ctrl, OUTPUT);//set direction control pin of A motor to OUTPUT
  pinMode(MR_PWM, OUTPUT);//Set PWM control pin of A motor to OUTPUT
  //Set pin to output
  pinMode(SCL_Pin,OUTPUT);
  pinMode(SDA_Pin,OUTPUT);
  //Clear the matrix display
  matrix_display(clear);
  matrix_display(start01);
}

void loop() {
  if(Serial.available()>0) {
    BLE_val = Serial.read();
    Serial.println(BLE_val);
  }
  switch(BLE_val) {
```

```
    case 'F': car_front(); matrix_display(front); break;
    case 'B': car_back(); matrix_display(back); break;
    case 'L': car_left(); matrix_display(left); break;
    case 'R': car_right(); matrix_display(right); break;
    case 'S': car_Stop();matrix_display(STOP01); break;
  }

}
void car_front() {
  digitalWrite(ML_Ctrl,HIGH);//set direction control pin of B motor to HIGH
  analogWrite(ML_PWM,200);//set PWM control speed of B motor to 200
  digitalWrite(MR_Ctrl,HIGH);//set direction control pin of A motor to HIGH
  analogWrite(MR_PWM,200);//set PWM control speed of A motor to 200
}
void car_back() {
  digitalWrite(ML_Ctrl,LOW);//set direction control pin of B motor to LOW
  analogWrite(ML_PWM,200);//set PWM control speed of B motor to 200
  digitalWrite(MR_Ctrl,LOW);//set direction control pin of A motor to LOW
  analogWrite(MR_PWM,200);//set PWM control speed of A motor to 200
}
void car_left() {
  digitalWrite(ML_Ctrl,LOW);//set direction control pin of B motor to LOW
  analogWrite(ML_PWM,200);//set PWM control speed of B motor to 200
  digitalWrite(MR_Ctrl,HIGH);//set direction control pin of A motor to HIGH
  analogWrite(MR_PWM,200);//set PWM control speed of A motor to 200
}
void car_right() {
  digitalWrite(ML_Ctrl,HIGH);//set direction control pin of B motor to HIGH
  analogWrite(ML_PWM,200);//set PWM control speed of B motor to 200
  digitalWrite(MR_Ctrl,LOW);//set direction control pin of A motor to LOW
  analogWrite(MR_PWM,200);//set PWM control speed of A motor to 200
}
void car_Stop()
{
  analogWrite(ML_PWM,0);//set PWM control speed of B motor to 0
  analogWrite(MR_PWM,0);//set PWM control speed of A motor to 0
}
//this function is used for dot matrix display
void matrix_display(unsigned char matrix_value[]) {
  IIC_start();  //the function that calls the data transmission
  IIC_send(0xc0);  //Select address
    for(int i = 0;i < 16;i++) //Pattern data has 16 bytes
     IIC_send(matrix_value[i]); //data to convey patterns
  IIC_end();   //end the transmission of patterns data
  IIC_start();
  IIC_send(0x8A);  //display control, set pulse width to 4/16
  IIC_end();
}
// the condition of data transmission starts
void IIC_start() {
  digitalWrite(SCL_Pin,HIGH);
  delayMicroseconds(3);
  digitalWrite(SDA_Pin,HIGH);
```

```
  delayMicroseconds(3);
  digitalWrite(SDA_Pin,LOW);
  delayMicroseconds(3);
}
// transmit data
void IIC_send(unsigned char send_data) {
  for(char i = 0;i < 8;i++) { //Every character has 8 bits
      digitalWrite(SCL_Pin,LOW);  //pull down the SCL_Pin to change the signal of SDA
      delayMicroseconds(3);
      if(send_data & 0x01)  //1 or 0 of byte is used to set high and low level of SDA_Pin
        digitalWrite(SDA_Pin,HIGH);
      else
        digitalWrite(SDA_Pin,LOW);
      delayMicroseconds(3);
      digitalWrite(SCL_Pin,HIGH); //Pull up SCL_Pin to stop data transmission
      delayMicroseconds(3);
      send_data = send_data >> 1;  //Detect bit by bit, so move the data right by one bit
  }
}
//the sign that data transmission ends
void IIC_end() {
  digitalWrite(SCL_Pin,LOW);
  delayMicroseconds(3);
  digitalWrite(SDA_Pin,LOW);
  delayMicroseconds(3);
  digitalWrite(SCL_Pin,HIGH);
  delayMicroseconds(3);
  digitalWrite(SDA_Pin,HIGH);
  delayMicroseconds(3);
} //*********************************************************************
```

Project 15: Multi-purpose Bluetooth Robot

For the final project, we will now combining the all the features including the Bluetooth, Ultrasonic follow and avoidance, and Line Tracking.

```
/*    Project 15: Multifunctional Robot car   */
unsigned char start01[] =
{0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80,0x80,0x40,0x20,0x10,0x08,0x04,0x02,0x01};
unsigned char front_matrix[] =
{0x00,0x00,0x00,0x00,0x00,0x24,0x12,0x09,0x12,0x24,0x00,0x00,0x00,0x00,0x00,0x00};
unsigned char back_matrix[] =
{0x00,0x00,0x00,0x00,0x00,0x24,0x48,0x90,0x48,0x24,0x00,0x00,0x00,0x00,0x00,0x00};
```

```
unsigned char left_matrix[] =
{0x00,0x00,0x00,0x00,0x00,0x00,0x44,0x28,0x10,0x44,0x28,0x10,0x44,0x28,0x10,0x00};
unsigned char right_matrix[] =
{0x00,0x10,0x28,0x44,0x10,0x28,0x44,0x10,0x28,0x44,0x00,0x00,0x00,0x00,0x00,0x00};
unsigned char STOP01[] =
{0x2E,0x2A,0x3A,0x00,0x02,0x3E,0x02,0x00,0x3E,0x22,0x3E,0x00,0x3E,0x0A,0x0E,0x00};
unsigned char clear[] = {0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00};

#define SCL_Pin  A5
#define SDA_Pin  A4

#include "SR04.h"
#define TRIG_PIN 12
#define ECHO_PIN 13
SR04 sr04 = SR04(ECHO_PIN,TRIG_PIN);
long distance,distance1,distance2,distance3;

const int left_ctrl = 4;
const int left_pwm = 5;
const int right_ctrl = 2;
const int right_pwm = 6;
const int sensor_l = 11;
const int sensor_c = 7;
const int sensor_r = 8;
int l_val,c_val,r_val;
const int servopin = 10;
char BLE_val;

void setup() {
  Serial.begin(9600);
  //irrecv.enableIRIn(); // Start the receiver
  servopulse(servopin,90);
  pinMode(left_ctrl,OUTPUT);
  pinMode(left_pwm,OUTPUT);
  pinMode(right_ctrl,OUTPUT);
  pinMode(right_pwm,OUTPUT);
  pinMode(sensor_l,INPUT);
  pinMode(sensor_c,INPUT);
  pinMode(sensor_r,INPUT);
  pinMode(SCL_Pin,OUTPUT);
  pinMode(SDA_Pin,OUTPUT);
  //Clear the screen
  matrix_display(clear);
  matrix_display(start01);
}

void loop() {
  if(Serial.available()>0) {
    BLE_val = Serial.read();
    Serial.println(BLE_val);
  }
  switch(BLE_val) {
    case 'F': front(); matrix_display(front_matrix); break;
```

```
    case 'B': back(); matrix_display(back_matrix); break;
    case 'L': left(); matrix_display(left_matrix); break;
    case 'R': right(); matrix_display(right_matrix); break;
    case 'S': Stop(); matrix_display(STOP01); break;
    case 'X': tracking(); break;
    case 'Y': avoid();break;
    case 'U': follow_car();break;
  }
}

void avoid() {
  matrix_display(start01);
  int track_flag = 0;
  while(track_flag == 0) {
    distance1=sr04.Distance();
    if((distance1 < 20)&&(distance1 != 0)) {
      Stop2();
      delay(100);
      servopulse(servopin,180);
      delay(500);
      distance2=sr04.Distance();
      delay(100);
      servopulse(servopin,0);
      delay(500);
      distance3=sr04.Distance();
      delay(100);
      if(distance2 > distance3) {
        left();
        servopulse(servopin,90);
      }
      else {
        right();
        servopulse(servopin,90);
      }
    }
    else {
      front();
    }
    if(Serial.available()>0) {
      BLE_val = Serial.read();
      if(BLE_val == 'S') {
        track_flag = 1;
      }
    }
  }
}

void follow_car() {
  matrix_display(start01);
  servopulse(servopin,90);
  int track_flag = 0;
  while(track_flag == 0) {
    distance = sr04.Distance();
```

```
    if(distance<8) {
      back2();
    }
    else if((distance>=8)&&(distance<13)) {
      Stop();
    }
    else if((distance>=13)&&(distance<35)) {
      front();
    }
    else {
      Stop();
    }
    if(Serial.available()>0) {
      BLE_val = Serial.read();
      if(BLE_val == 'S') {
        track_flag = 1;
      }
    }
  }
}

void servopulse(int servopin,int myangle) {
  for(int i=0;i<30;i++){
    int pulsewidth = (myangle*11)+500;
    digitalWrite(servopin,HIGH);
    delayMicroseconds(pulsewidth);
    digitalWrite(servopin,LOW);
    delay(20-pulsewidth/1000);
  }
}

void tracking() {
  matrix_display(start01);
  int track_flag = 0;
  while(track_flag == 0) {
    l_val = digitalRead(sensor_l);
    c_val = digitalRead(sensor_c);
    r_val = digitalRead(sensor_r);

    if(c_val == 1) {
      front2();
    }
    else {
      if((l_val == 1)&&(r_val == 0))
        left();
      else if((l_val == 0)&&(r_val == 1))
        right();
      else
        Stop();
    }
    if(Serial.available()>0) {
      BLE_val = Serial.read();
      if(BLE_val == 'S')
```

```
        track_flag = 1;
    }
  }
}

void front() {
  digitalWrite(left_ctrl,HIGH);
  analogWrite(left_pwm,220);
  digitalWrite(right_ctrl,HIGH);
  analogWrite(right_pwm,190);
}
void front2() {
  digitalWrite(left_ctrl,HIGH);
  analogWrite(left_pwm,75);
  digitalWrite(right_ctrl,HIGH);
  analogWrite(right_pwm,70);
}
void back() {
  digitalWrite(left_ctrl,LOW);
  analogWrite(left_pwm,220);
  digitalWrite(right_ctrl,LOW);
  analogWrite(right_pwm,190);
}
void back2() {
  digitalWrite(left_ctrl,LOW);
  analogWrite(left_pwm,110);
  digitalWrite(right_ctrl,LOW);
  analogWrite(right_pwm,90);
}
void left() {
  digitalWrite(left_ctrl,LOW);
  analogWrite(left_pwm,220);
  digitalWrite(right_ctrl,HIGH);
  analogWrite(right_pwm,190);
}
void right() {
  digitalWrite(left_ctrl,HIGH);
  analogWrite(left_pwm,220);
  digitalWrite(right_ctrl,LOW);
  analogWrite(right_pwm,190);
}
void Stop() {
  analogWrite(left_pwm,0);
  analogWrite(right_pwm,0);
}
void Stop2() {
  digitalWrite(left_ctrl,LOW);
  analogWrite(left_pwm,200);
  digitalWrite(right_ctrl,LOW);
  analogWrite(right_pwm,200);
  delay(50);
  analogWrite(left_pwm,0);
  analogWrite(right_pwm,0);
```

```
}

//this function is used for dot matrix display
void matrix_display(unsigned char matrix_value[]) {
  IIC_start();  // the function to transmit data
  IIC_send(0xc0);  //select address
    for(int i = 0;i < 16;i++) //pattern data has 16 bytes
     IIC_send(matrix_value[i]); //data transmits patterns
  IIC_end();   //end the transmission of patterns data
  IIC_start();
  IIC_send(0x8A);  //display the control, set pulse width to 4/16
  IIC_end();
}
// The condition of data transmission starts
void IIC_start() {
  digitalWrite(SCL_Pin,HIGH);
  delayMicroseconds(3);
  digitalWrite(SDA_Pin,HIGH);
  delayMicroseconds(3);
  digitalWrite(SDA_Pin,LOW);
  delayMicroseconds(3);
}
// transmit data
void IIC_send(unsigned char send_data) {
  for(char i = 0;i < 8;i++) {  //Every character has 8 bits
      digitalWrite(SCL_Pin,LOW);  //pull down the SCL_Pin to change the signal of SDA
      delayMicroseconds(3);
      if(send_data & 0x01)  // 1 or 0 of byte is used to set high and low level of SDA_Pin
        digitalWrite(SDA_Pin,HIGH);
      else
        digitalWrite(SDA_Pin,LOW);
      delayMicroseconds(3);
      digitalWrite(SCL_Pin,HIGH); //pull up the SCL_Pin to stop transmitting data      delayMicroseconds(3);
      send_data = send_data >> 1;  //Detect bit by bit, so move the data right by one bit detect bit by bit, move data
  }
}
//the sign that data ends transmitting
void IIC_end() {
  digitalWrite(SCL_Pin,LOW);
  delayMicroseconds(3);
  digitalWrite(SDA_Pin,LOW);
  delayMicroseconds(3);
  digitalWrite(SCL_Pin,HIGH);
  delayMicroseconds(3);
  digitalWrite(SDA_Pin,HIGH);
  delayMicroseconds(3);
}//*****************************************************
```

FINAL DEMONSTRATION

For this project, I was able to demonstrate to the instructor the complete smart car in addition to 3 required demonstrations: simple moving car, a square pattern, and obstacle avoidance. For the first demonstration, I was required to simply move the smart car with the Bluetooth. The second demonstration requires to make the smart car a square pattern where the car moves 4 rights and goes back to original spot. Lastly, the smart car must avoid any obstacle which featured in one of the projects. These 3 demonstrations were able to complete and approved by the instructor. On the other hand, my smart car doesn't include an arm robot as the required kit were out of stock by the time I started to purchase for the laboratory kit. In addition, there are missing and different project orders in this report since I followed my smart car's documentation.

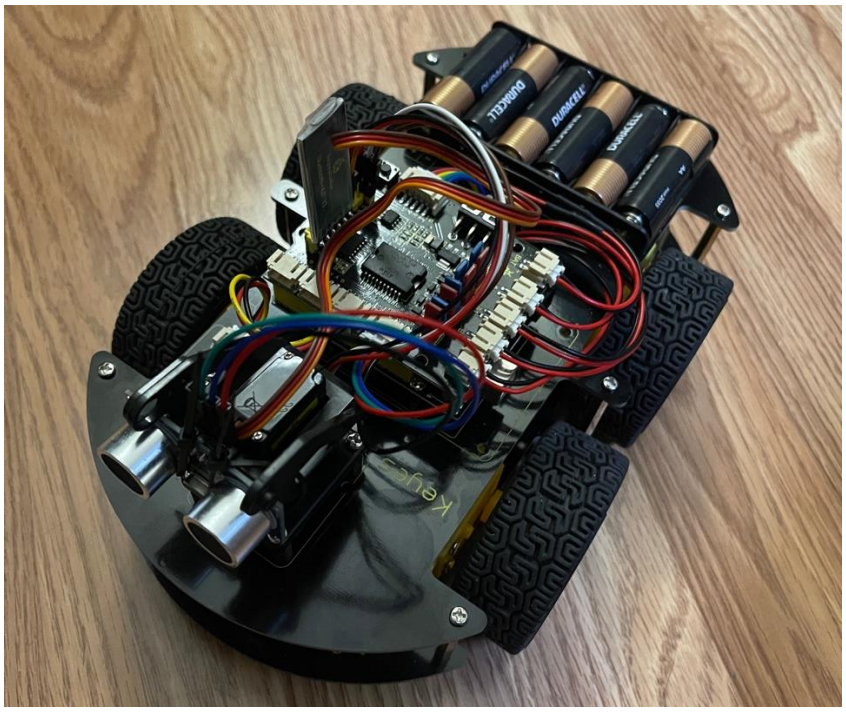

Figure 16. Final product of the Smart Car