Lesson 6 Integrated all functions

Ⅰ. Introduction

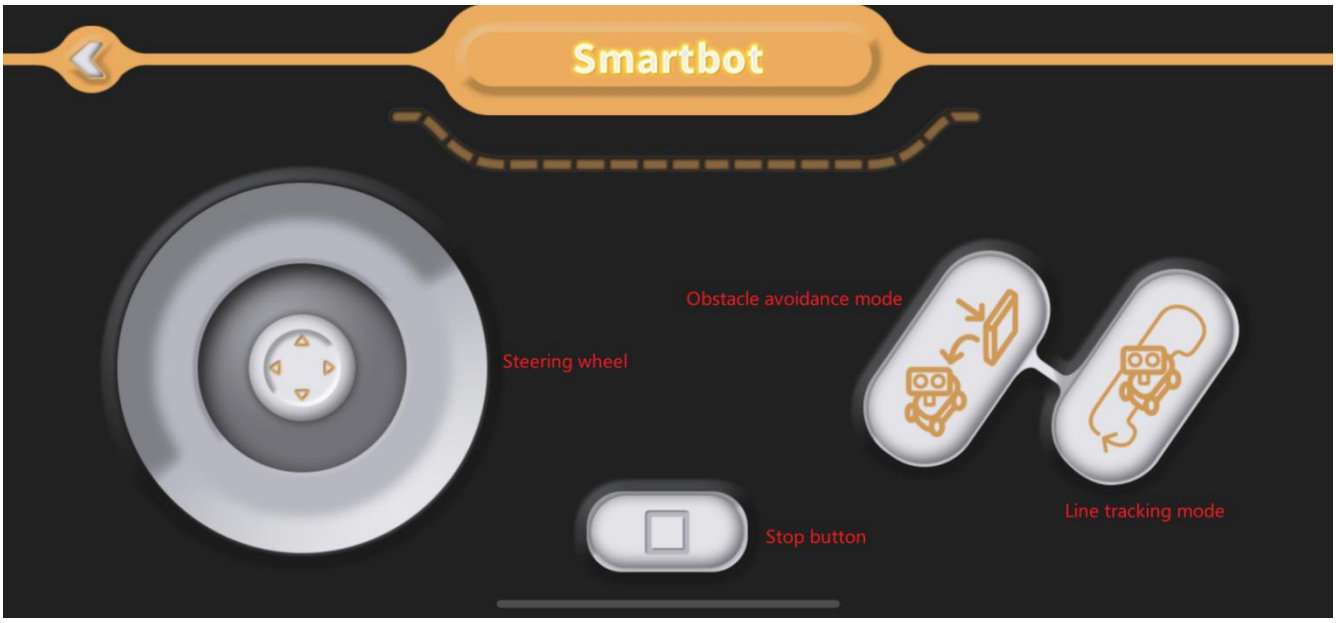
The program integrates Bluetooth, infrared remote control, line tracing, obstacle avoidance and other functions.

Infrared Remote Mode

Press any button on the infrared remote control to enter the infrared remote control mode. Press the key #1 button to enter the line tracking mode. Press the key #2 button to enter the obstacles avoidance mode. Press the four directional keys to control the car forward, backward and turn, and the "OK" key to stop.

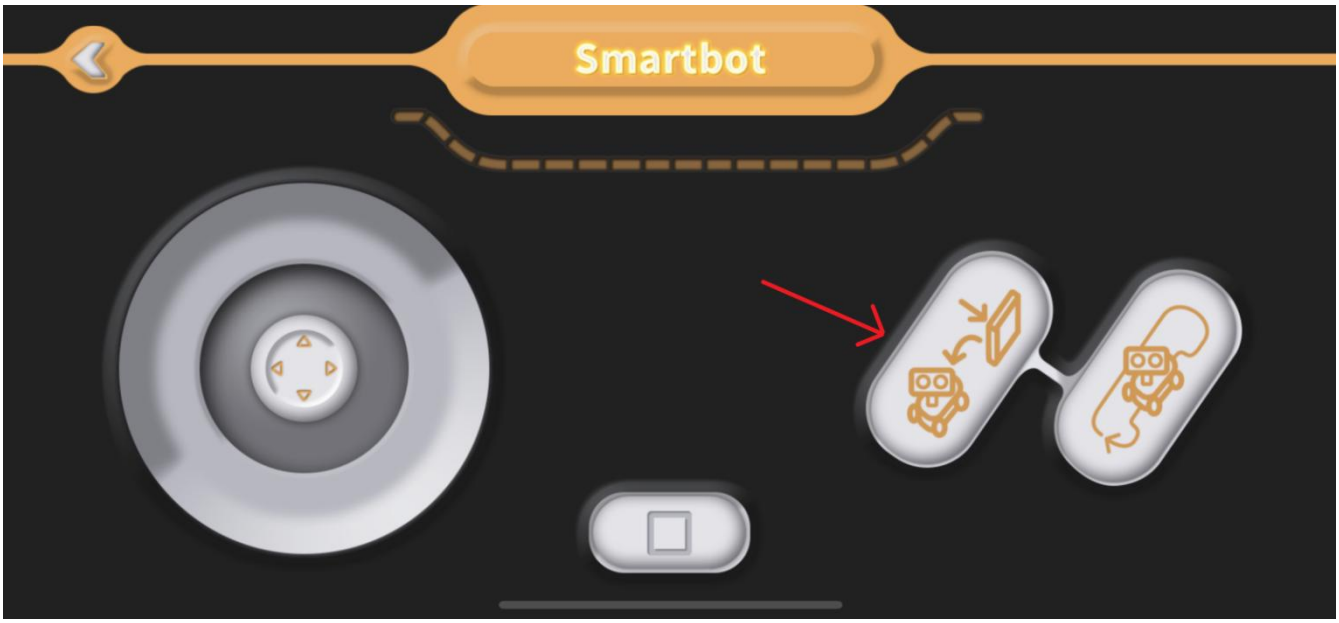
Bluetooth Mode

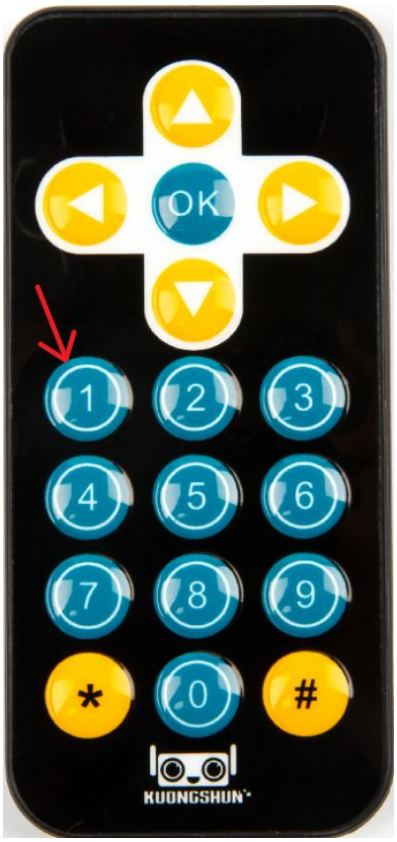
Press any button on the "kuongshun" app to connect the device and enter Bluetooth mode. Press the "Patrol Mode" button to enter the patrol mode. Press the "Obstacle Avoidance Mode" button to enter the obstacle avoidance mode. Drag the direction of the steering wheel to control the cart "forward", "backward", "left", "right", press the "stop" button to control the cart to stop.



Ultrasonic Avoidance Mode

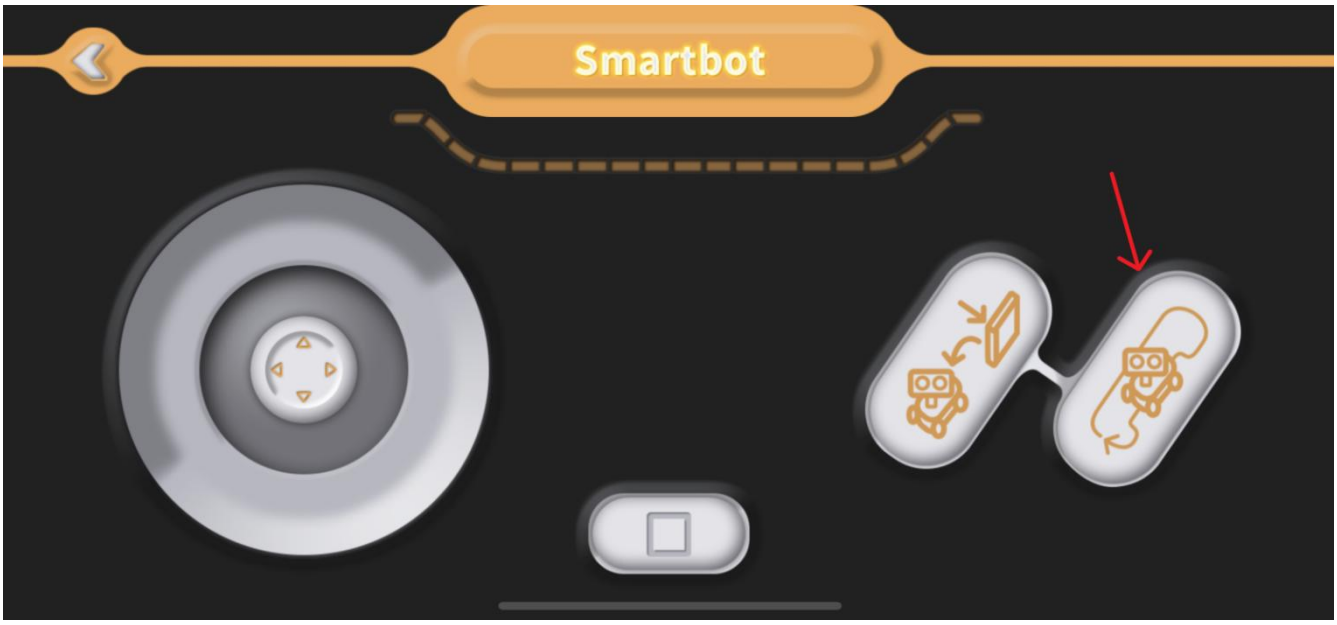
This mode can be accessed by pressing the Obstacle avoidance mode button on Bluetooth, or by pressing 1 through the infrared remote control.

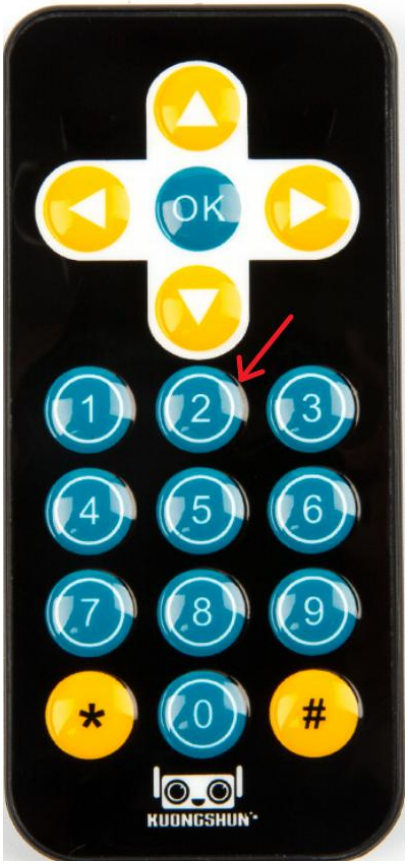




Line Tracking Mode

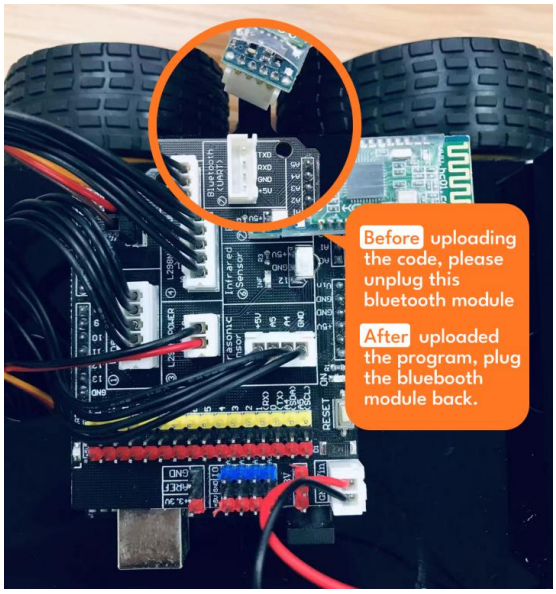
This mode can be accessed by pressing the Bluetooth Line mode button, or by pressing 2 through the infrared remote control.

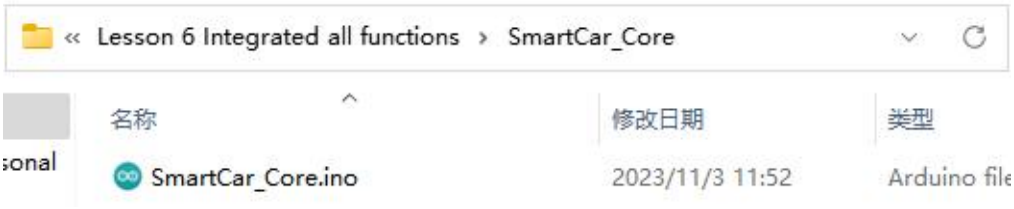


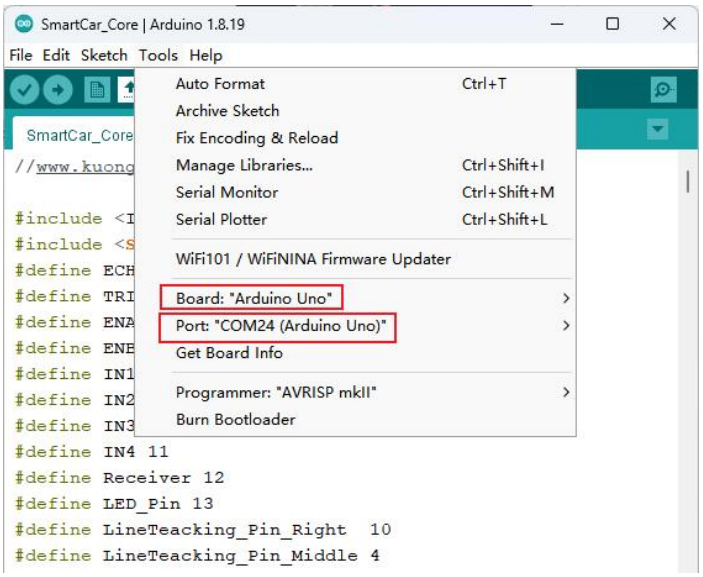


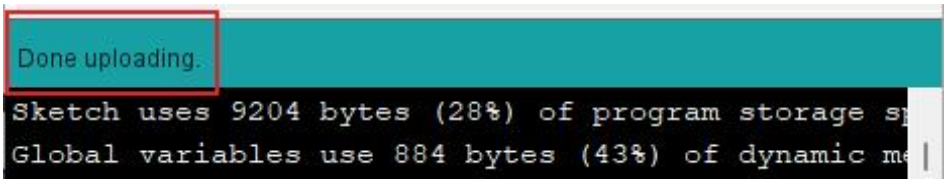
Ⅱ. Upload program

We need a USB cable to connect the car to the computer. First take off the Bluetooth module and find the path of the integrated multi-function car SmartCar_Core program. Open the program and swipe in the program. After uploading successfully, connect the Bluetooth module back.





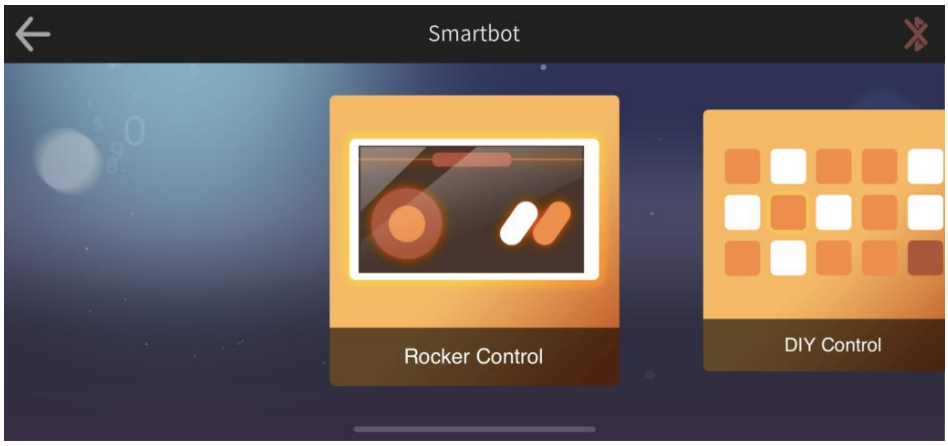




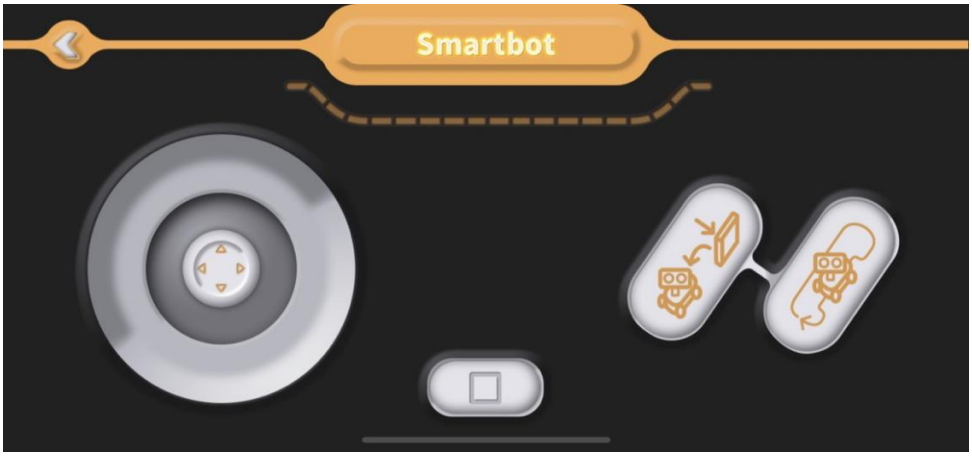
After uploading the program to the UNO control board, connect back to the Bluetooth module. Then disconnect the data cable, put the car on the ground and turn on the power. Get a mobile phone or tablet with "kuongshun" installed. Open the software and Bluetooth on your phone to connect the device by following these steps:

- Open kuongshun and select Smartbot

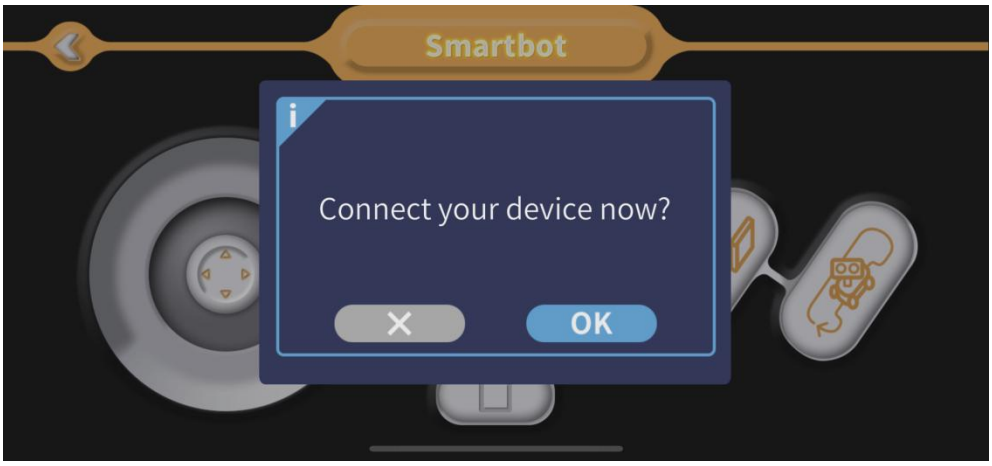
- Select joystick control



- After opening the interface as follows, operate any button

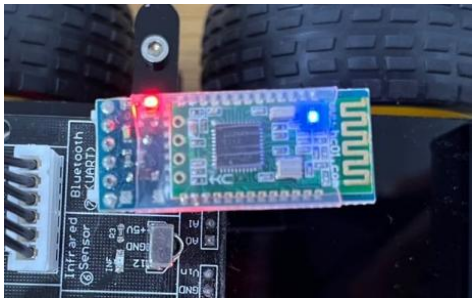


- A selection box will appear as shown below, select OK



- An animation will appear

- Put the cell phone close to the car when the car Bluetooth module stops blinking to stay lit, that is, the connection is successful(if the connection fails, please power off and restart the car)



Now, you can drive the car as much as you like, there are four modes you can switch at will.(it is recommended to stop the car before switching)

Code review

```
//www.kuongshun.com

#include <IRremote.h>
#include <Servo.h>
#define ECHO_PIN  A4
#define TRIG_PIN  A5
#define ENA 5
#define ENB 6
#define IN1 7
#define IN2 8
#define IN3 9
#define IN4 11
#define Receiver 12
#define LED_Pin 13
#define LineTeacking_Pin_Right  10
#define LineTeacking_Pin_Middle 4
#define LineTeacking_Pin_Left   2
#define LineTeacking_Read_Right   !digitalRead(10)
#define LineTeacking_Read_Middle  !digitalRead(4)
#define LineTeacking_Read_Left    !digitalRead(2)
#define carSpeed 200

Servo servo;
IRrecv irrecv(Receiver);
uint32_t last_decodedRawData = 0;
decode_results results;
unsigned long IR_PreMillis;
unsigned long LT_PreMillis;
int rightDistance = 0, leftDistance = 0, middleDistance = 0;

enum FUNCTIONMODE {
  IDLE,
  LineTeacking,
  ObstaclesAvoidance,
  Bluetooth,
  IRremote
} func_mode = IDLE;

enum MOTIONMODE {
  STOP,
  FORWARD,
  BACK,
```

```
  LEFT,
  RIGHT
} mov_mode = STOP;

void delays(unsigned long t) {
  for (unsigned long i = 0; i < t; i++) {
    getBTData();
    getIRData();
    delay(1);
  }
}

int getDistance() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  return (int)pulseIn(ECHO_PIN, HIGH) / 58;
}

void forward(bool debug = false) {
  analogWrite(ENA, carSpeed);
  analogWrite(ENB, carSpeed);
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
  if (debug) Serial.println("Go forward!");
}

void back(bool debug = false) {
  analogWrite(ENA, carSpeed);
  analogWrite(ENB, carSpeed);
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  if (debug) Serial.println("Go back!");
}

void left(bool debug = false) {
  analogWrite(ENA, carSpeed);
  analogWrite(ENB, carSpeed);
```

```
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
  if (debug) Serial.println("Go left!");
}

void right(bool debug = false) {
  analogWrite(ENA, carSpeed);
  analogWrite(ENB, carSpeed);
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  if (debug) Serial.println("Go right!");
}

void stop(bool debug = false) {
  digitalWrite(ENA, LOW);
  digitalWrite(ENB, LOW);
  if (debug) Serial.println("Stop!");
}

void getBTData() {
  if (Serial.available()) {
    switch (Serial.read()) {
      case 'f': func_mode = Bluetooth; mov_mode = FORWARD;  break;
      case 'b': func_mode = Bluetooth; mov_mode = BACK;     break;
      case 'l': func_mode = Bluetooth; mov_mode = LEFT;     break;
      case 'r': func_mode = Bluetooth; mov_mode = RIGHT;    break;
      case 's': func_mode = Bluetooth; mov_mode = STOP;     break;
      case '1': func_mode = LineTeacking;                  break;
      case '2': func_mode = ObstaclesAvoidance;            break;
      default:  break;
    }
  }
}

void getIRData() // takes action based on IR code received
{
  if (irrecv.decode()) // have we received an IR signal?
  {
    if (irrecv.decodedIRData.flags)// Check if it is a repeat IR code
    {
```

```
      irrecv.decodedIRData.decodedRawData = last_decodedRawData;    //set the
current decodedRawData to the last decodedRawData
      Serial.println("REPEAT!");
    } else//output the IR code on the serial monitor
    {
      Serial.print("IR code:0x");
      Serial.println(irrecv.decodedIRData.decodedRawData, HEX);
    }
    switch (irrecv.decodedIRData.decodedRawData)//map the IR code to the remote
key
    {
      case 0xB946FF00: Serial.println("UP"); func_mode = IRremote; mov_mode =
FORWARD; break;
      case 0xEA15FF00: Serial.println("DOWN"); func_mode = IRremote; mov_mode =
BACK; break;
      case 0xBB44FF00: Serial.println("LEFT"); func_mode = IRremote; mov_mode =
LEFT; break;
      case 0xBC43FF00: Serial.println("RIGHT"); func_mode = IRremote; mov_mode
= RIGHT; break;
      case 0xBF40FF00: Serial.println("OK"); func_mode = IRremote; mov_mode = STOP;
break;
      case 0xAD52FF00: Serial.println("0");    break;
      case 0xE916FF00: Serial.println("1"); func_mode = LineTeacking; break;
      case 0xE619FF00: Serial.println("2"); func_mode = ObstaclesAvoidance;
break;
      case 0xF20DFF00: Serial.println("3");    break;
      case 0xF30CFF00: Serial.println("4");    break;
      case 0xE718FF00: Serial.println("5");    break;
      case 0xA15EFF00: Serial.println("6");    break;
      case 0xF708FF00: Serial.println("7");    break;
      case 0xE31CFF00: Serial.println("8");    break;
      case 0xA55AFF00: Serial.println("9");    break;
      case 0xBD42FF00: Serial.println("*");    break;
      case 0xB54AFF00: Serial.println("#");    break;
      default:
        Serial.println(" other button   ");
    }// End Case
    last_decodedRawData = irrecv.decodedIRData.decodedRawData;//store the last
decodedRawData
    delay(100); // Do not get immediate repeat
  }
  irrecv.resume(); // receive the next value
} //END translateIR
```

```
void bluetooth_mode() {
  if (func_mode == Bluetooth) {
    switch (mov_mode) {
      case FORWARD: forward();  break;
      case BACK:    back();     break;
      case LEFT:    left();     break;
      case RIGHT:   right();    break;
      case STOP:    stop();     break;
      default: break;
    }
  }
}

void irremote_mode() {
  if (func_mode == IRremote) {
    switch (mov_mode) {
      case FORWARD: forward();  break;
      case BACK:    back();     break;
      case LEFT:    left();     break;
      case RIGHT:   right();    break;
      case STOP:    stop();     break;
      default: break;
    }
  }
}

void line_teacking_mode() {
  if (func_mode == LineTeacking) {
    if (LineTeacking_Read_Middle) {
      forward();
      LT_PreMillis = millis();
    } else if (LineTeacking_Read_Right) {
      right();
      while (LineTeacking_Read_Right) {
        getBTData();
        getIRData();
      }
      LT_PreMillis = millis();
    } else if (LineTeacking_Read_Left) {
      left();
      while (LineTeacking_Read_Left) {
        getBTData();
        getIRData();
      }
```

```
      LT_PreMillis = millis();
    } else {
      if (millis() - LT_PreMillis > 150) {
        stop();
      }
    }
  }
}

void obstacles_avoidance_mode() {
  if (func_mode == ObstaclesAvoidance) {
    servo.write(90);
    delays(500);
    middleDistance = getDistance();
    if (middleDistance <= 40) {
      stop();
      delays(500);
      servo.write(10);
      delays(500);
      rightDistance = getDistance();

      delays(500);
      servo.write(90);
      delays(500);
      servo.write(170);
      delays(500);
      leftDistance = getDistance();

      delays(500);
      servo.write(90);
      delays(500);
      if (rightDistance > leftDistance) {
        stop();
        delay(100);
        back();
        delay(200);
        right();
        delay(300);
      } else if (rightDistance < leftDistance) {
        stop();
        delay(100);
        back();
        delay(200);
        left();
```

```
        delay(300);
      } else if ((rightDistance <= 40) || (leftDistance <= 40)) {
        back();
        delays(180);
      } else {
        forward();
      }
    } else {
      forward();
    }
  }
}

void setup() {
  Serial.begin(9600);
  servo.attach(3, 500, 2400); // 500: 0 degree  2400: 180 degree
  servo.write(90);
  irrecv.enableIRIn();
  pinMode(ECHO_PIN, INPUT);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
  pinMode(ENA, OUTPUT);
  pinMode(ENB, OUTPUT);
  pinMode(LineTeacking_Pin_Right, INPUT);
  pinMode(LineTeacking_Pin_Middle, INPUT);
  pinMode(LineTeacking_Pin_Left, INPUT);
}

void loop() {
  getBTData();
  getIRData();
  bluetooth_mode();
  irremote_mode();
  line_teacking_mode();
  obstacles_avoidance_mode();
}
```