Lesson 6 Integrated all functions

The final lesson, we integrate all function of Smart Car into one sketch

Ⅰ. Introduction

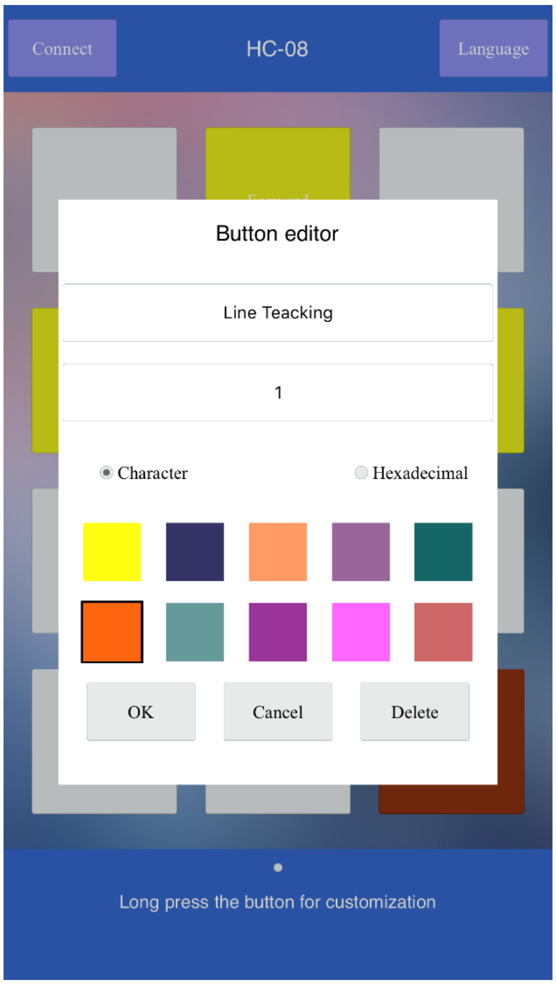
This is our default factory program. The program integrates Bluetooth, infrared remote control, line tracing, obstacle avoidance and other functions.

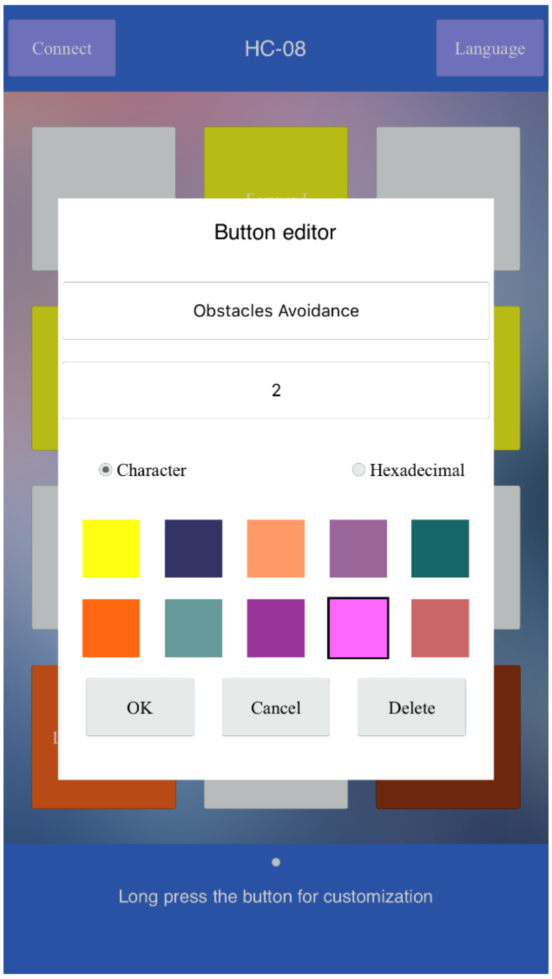
Infrared Remote Mode

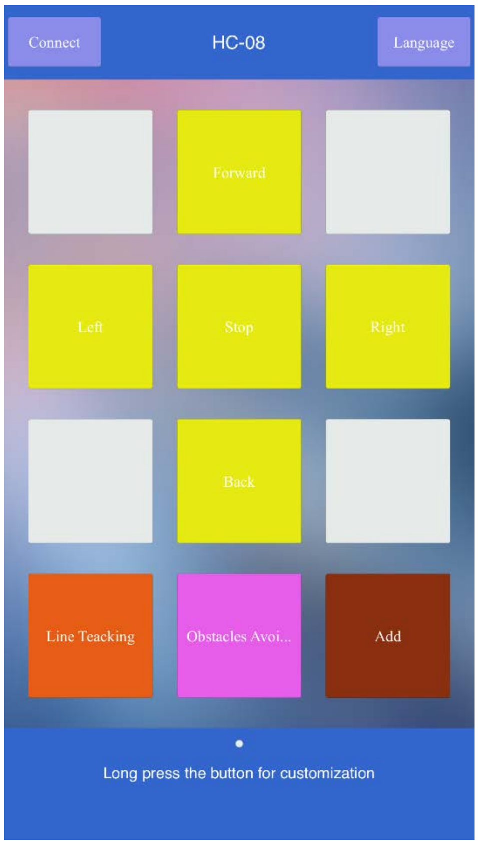
Press any button on the infrared remote control to enter the infrared remote control mode. Press the key #1 button to enter the line tracking mode. Press the key #2 button to enter the obstacles avoidance mode. Press the ‘f’, ‘b’, ‘l’, ‘r’, ‘s’ button to control the movement of the car.

Bluetooth Mode

Press any button on the ble app to enter the bluetooth mode. Press the “Line Teacking” button to enter the line tracking mode. Press the “Obstacles Avoidance” button to enter the obstacles avoidance mode. Press the ‘Forward’, ‘Back’, ‘Left’, ‘Right’, ‘Stop’ button to control the movement of the car.
Long press to customize the buttons.







Ⅱ. Upload program

You need to open the the code file "\Lesson 6 Integrated all functions\SmartCar_Core\SmartCar_Core.ino" and upload the program to the UNO controller board.

Code review

```
#include <IRremote.h>
#include <Servo.h>

#define f 16736925  // FORWARD
#define b 16754775  // BACK
#define l 16720605  // LEFT
#define r 16761405  // RIGHT
#define s 16712445  // STOP
#define KEY1 16738455 //Line Teacking mode
#define KEY2 16750695 //Obstacles Avoidance mode
#define KEY3 16756815
#define KEY4 16724175
#define KEY5 16718055
```

Define the key code of the remote control

```
#define KEY6 16743045
#define KEY7 16716015
#define KEY8 16726215
#define KEY9 16734885
#define KEY0 16730805
#define KEY_STAR 16728765
#define KEY_HASH 16732845
```

Define the io pin

```
#define RECV_PIN  12
#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 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 250
```

Define the PWM value

Define the object

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

Define the function mode

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

enum MOTIONMODE {
  STOP,
  FORWARD,
```

Define the motion

```
  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);
```

Detecting the distance of obstacles

```
  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!");
}
```

Receive Bluetooth commands

```
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;
    }
  }
}
```

Receiving infrared remote control commands

```
void getIRData() {
  if (irrecv.decode(&results)){
    IR_PreMillis = millis();
    switch(results.value){
      case f:   func_mode = IRremote; mov_mode = FORWARD;  break;
      case b:   func_mode = IRremote; mov_mode = BACK;     break;
      case l:   func_mode = IRremote; mov_mode = LEFT;     break;
      case r:   func_mode = IRremote; mov_mode = RIGHT;    break;
      case s:   func_mode = IRremote; mov_mode = STOP;     break;
      case KEY1:  func_mode = LineTeacking;                break;
      case KEY2:  func_mode = ObstaclesAvoidance;          break;
      default: break;
    }
    irrecv.resume();
  }
}

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;
    }
    if(millis() - IR_PreMillis > 500){
      mov_mode = STOP;
      IR_PreMillis = millis();
    }
  }
}

void line_teacking_mode() {
  if(func_mode == LineTeacking){
```

Bluetooth mode

Infrared remote control mode

Line teacking mode

```
    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(1000);
      rightDistance = getDistance();

      delays(500);
      servo.write(90);
      delays(1000);
      servo.write(170);
      delays(1000);
      leftDistance = getDistance();
```

Obstacle avoidance mode

```
      delays(500);
      servo.write(90);
      delays(1000);
      if(rightDistance > leftDistance) {
        right();
        delays(360);
      } else if(rightDistance < leftDistance) {
        left();
        delays(360);
      } 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();
```

Robot car initialization

```
  line_teacking_mode();
  obstacles_avoidance_mode();
}
```