# Car Infrared Remote

Control RGB, buzzer and car motion status through infrared remote control.

```
Since the previous tutorial has explained the basic knowledge of all car modules,
the car control tutorial will not repeat it, and mainly explain the ideas for
implementing car functions!
```

## Device connection

### Hardware connection

Use Type-B data cable to connect Arduino Uno and computer.

### Software connection

Open the "Arduino IDE" software and select the model and serial port number corresponding to the development board.

## Implementation ideas

Use the infrared remote control with the infrared receiver to parse the data, and then control the corresponding peripheral status of the car according to the data.

| IR remote control | Parsed data | Function |
|---|---|---|
| Power | 0x00 | Stop all peripheral functions |
| RGB | 0x40 | Switch RGB colors |
| Buzzer | 0xA0 | Control buzzer sound |
| Car Forward | 0x80 | Car forward |
| Car Backward | 0x90 | Car backward |
| Car Left | 0x20 | Car moves to the left |
| Car Right | 0x60 | Car moves to the right |
| Car Left Spin | 0x10 | Car rotates to the left |
| Car Right Spin | 0x50 | Car rotates to the right |

| IR remote control | Parsed data | Function |
|---|---|---|
| Add | 0x30 | Increase speed |
| Sub | 0x70 | Decrease speed |
| Number 0 | 0xB0 | None |
| Number 1 | 0x08 | None |
| Number 2 | 0x88 | None |
| Number 3 | 0x48 | None |
| Number 4 | 0x28 | None |
| Number 5 | 0xA8 | None |
| Number 6 | 0x68 | None |
| Number 7 | 0x18 | None |
| Number 8 | 0x98 | None |
| Number 9 | 0x58 | None |

## Code Analysis

Here is only a brief introduction to the code content. For detailed code, please refer to the corresponding code file, which is provided in the download area!

- Include `Wire`、`IRremote`、`Adafruit_NeoPixel`、`Adafruit_PWMServoDriver` library

```
#include <Wire.h>                      // 包含Wire(I2C)通讯库 Include Wire library
#include <IRremote.hpp>                // 包含IRremote库 Include IRremote library
#include <Adafruit_NeoPixel.h>         // 包含Adafruit NeoPixel库 Include Adafruit NeoPixel library
#include <Adafruit_PWMServoDriver.h>   // 包含Adafruit PWMServoDriver库 Include Adafruit PWMServoDriver library
```

- Define motor, infrared, RGB, buzzer control pins, I2C address, motor frequency and initial speed, etc.

```
// 定义红外接收引脚 Define IR receiver pins
#define IR_RECEIVE_PIN A3

// 定义电机控制引脚 Define motor control pins
#define Motor_L1_F_PIN 11  // 控制小车左前方电机前进 Control the motor on the left front of the car
#define Motor_L1_B_PIN 10  // 控制小车左前方电机后退 Control the motor back on the left front of the car
#define Motor_L2_F_PIN 8   // 控制小车左后方电机前进 Control car left rear motor forward
#define Motor_L2_B_PIN 9   // 控制小车左后方电机后退 Control the car left rear motor back
```

```
#define Motor_R1_F_PIN 13  // 控制小车右前方电机前进 Control the right front motor of the car to move forward
#define Motor_R1_B_PIN 12  // 控制小车右前方电机后退 Control the motor back on the right front of the car
#define Motor_R2_F_PIN 14  // 控制小车右后方电机前进 Control car right rear motor forward
#define Motor_R2_B_PIN 15  // 控制小车右后方电机后退 Control car right rear motor back

// 控制LOGO对应的LED引脚 Control LED pins
#define LED_PIN 7

// 定义底层驱动芯片参数 Bottom-layer driver chip related parameters
#define Bottom_Layer_Driver_ADDR 0x40

// 定义PWM频率 Define PWM frequency
#define PWM_FREQUENCY 50

// 定义RGB控制引脚和数量 Define RGB control pins and quantity
#define RGB_PIN 6
#define RGB_NUM 1

// 定义蜂鸣器控制引脚 Define BUZZER control pins
#define BUZZER_PIN 10

// 定义红外键值对应的红外数据 Define the value corresponding to the infrared remote control key
const uint8_t Power = 0x00;
const uint8_t RGB_Light = 0x40;
const uint8_t Buzzer = 0xA0;
const uint8_t CarForward = 0x80;
const uint8_t CarBackward = 0x90;
const uint8_t CarLeft = 0x20;
const uint8_t CarRight = 0x60;
const uint8_t CarLeftSpin = 0x10;
const uint8_t CarRightSpin = 0x50;
const uint8_t Add = 0x30;
const uint8_t Sub = 0x70;
const uint8_t Number_0 = 0xB0;
const uint8_t Number_1 = 0x08;
const uint8_t Number_2 = 0x88;
const uint8_t Number_3 = 0x48;
const uint8_t Number_4 = 0x28;
const uint8_t Number_5 = 0xA8;
const uint8_t Number_6 = 0x68;
const uint8_t Number_7 = 0x18;
const uint8_t Number_8 = 0x98;
const uint8_t Number_9 = 0x58;

int iCarSpeed = 75;
unsigned int uRgbState = 0;
```

- Enumerate common colors

```
// 枚举常见颜色 Enumerate common colors
enum ColorType {
  BLACK,
  RED,
  GREEN,
  BLUE,
  YELLOW,
  MAGENTA,
  CYAN,
  WHITE,
};
```

- Enumerate the common movement modes of omnidirectional cars

```
// 枚举全向小车的常见运动方式 Enumerate the common movement modes of omnidirectional
cars
enum OmniDirectionalCar {
  STOP,
  FORWARD,
  BACKWARD,
  LEFT,
  RIGHT,
  LEFT_ROTATE,
  RIGHT_ROTATE,
  LEFT_FORWARD,
  RIGHT_BACKWARD,
  RIGHT_FORWARD,
  LEFT_BACKWARD,
};
```

- Create an instance of the Adafruit_PWMServoDriver class

```
// 创建Adafruit_PWMServoDriver类的实例 Create an instance of the
Adafruit_PWMServoDriver class
Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver(Bottom_Layer_Driver_ADDR);
// 创建Adafruit_NeoPixel类的实例 Create an instance of the Adafruit_NeoPixel class
Adafruit_NeoPixel RGB = Adafruit_NeoPixel(RGB_NUM, RGB_PIN, NEO_GRB +
NEO_KHZ800);
```

- Setting the Motor Speed

```
/**
 * @brief 设置单个电机速度 Setting the Motor Speed
 * @param motor_forward_pin: 控制电机前进引脚 Control the motor forward pin
 * @param motor_backward_pin: 控制电机后退引脚 Control the motor backward pin
 * @param motor_speed: 设置电机速度 Setting the Motor Speed
 * @retval 无 None
 */
void setMotorSpeed(uint16_t motor_forward_pin, uint16_t motor_backward_pin, int
motor_speed) {
  motor_speed = map(motor_speed, -255, 255, -4095, 4095);
  if (motor_speed >= 0) {
    pwm.setPWM(motor_forward_pin, 0, motor_speed);
    pwm.setPWM(motor_backward_pin, 0, 0);
  } else if (motor_speed < 0) {
    pwm.setPWM(motor_forward_pin, 0, 0);
    pwm.setPWM(motor_backward_pin, 0, -(motor_speed));
  }
}
```

- Set the car movement mode and speed

```
/**
 * @brief 设置小车运动方式和速度 Set the car movement mode and speed
 * @param Movement: 小车运动方式 Car movement
 * @param Speed: 小车运动速度 Car speed
 * @retval 无 None
 */
void setCarMove(uint8_t Movement, int Speed) {
  switch (Movement) {
    case STOP:
      setMotorSpeed(Motor_L1_F_PIN, Motor_L1_B_PIN, 0);
      setMotorSpeed(Motor_L2_F_PIN, Motor_L2_B_PIN, 0);
      setMotorSpeed(Motor_R1_F_PIN, Motor_R1_B_PIN, 0);
      setMotorSpeed(Motor_R2_F_PIN, Motor_R2_B_PIN, 0);
      break;
    case FORWARD:
      setMotorSpeed(Motor_L1_F_PIN, Motor_L1_B_PIN, Speed);
      setMotorSpeed(Motor_L2_F_PIN, Motor_L2_B_PIN, Speed);
      setMotorSpeed(Motor_R1_F_PIN, Motor_R1_B_PIN, Speed);
      setMotorSpeed(Motor_R2_F_PIN, Motor_R2_B_PIN, Speed);
      break;
    case BACKWARD:
      setMotorSpeed(Motor_L1_F_PIN, Motor_L1_B_PIN, -Speed);
      setMotorSpeed(Motor_L2_F_PIN, Motor_L2_B_PIN, -Speed);
      setMotorSpeed(Motor_R1_F_PIN, Motor_R1_B_PIN, -Speed);
      setMotorSpeed(Motor_R2_F_PIN, Motor_R2_B_PIN, -Speed);
      break;
    case LEFT:
      setMotorSpeed(Motor_L1_F_PIN, Motor_L1_B_PIN, -Speed);
      setMotorSpeed(Motor_L2_F_PIN, Motor_L2_B_PIN, Speed);
      setMotorSpeed(Motor_R1_F_PIN, Motor_R1_B_PIN, Speed);
      setMotorSpeed(Motor_R2_F_PIN, Motor_R2_B_PIN, -Speed);
      break;
    case RIGHT:
      setMotorSpeed(Motor_L1_F_PIN, Motor_L1_B_PIN, Speed);
      setMotorSpeed(Motor_L2_F_PIN, Motor_L2_B_PIN, -Speed);
      setMotorSpeed(Motor_R1_F_PIN, Motor_R1_B_PIN, -Speed);
      setMotorSpeed(Motor_R2_F_PIN, Motor_R2_B_PIN, Speed);
      break;
    case LEFT_ROTATE:
      setMotorSpeed(Motor_L1_F_PIN, Motor_L1_B_PIN, -Speed);
      setMotorSpeed(Motor_L2_F_PIN, Motor_L2_B_PIN, -Speed);
      setMotorSpeed(Motor_R1_F_PIN, Motor_R1_B_PIN, Speed);
      setMotorSpeed(Motor_R2_F_PIN, Motor_R2_B_PIN, Speed);
      break;
    case RIGHT_ROTATE:
      setMotorSpeed(Motor_L1_F_PIN, Motor_L1_B_PIN, Speed);
      setMotorSpeed(Motor_L2_F_PIN, Motor_L2_B_PIN, Speed);
      setMotorSpeed(Motor_R1_F_PIN, Motor_R1_B_PIN, -Speed);
      setMotorSpeed(Motor_R2_F_PIN, Motor_R2_B_PIN, -Speed);
      break;
    case LEFT_FORWARD:
      setMotorSpeed(Motor_L1_F_PIN, Motor_L1_B_PIN, 0);
      setMotorSpeed(Motor_L2_F_PIN, Motor_L2_B_PIN, Speed);
      setMotorSpeed(Motor_R1_F_PIN, Motor_R1_B_PIN, Speed);
      setMotorSpeed(Motor_R2_F_PIN, Motor_R2_B_PIN, 0);
      break;
    case RIGHT_BACKWARD:
      setMotorSpeed(Motor_L1_F_PIN, Motor_L1_B_PIN, 0);
      setMotorSpeed(Motor_L2_F_PIN, Motor_L2_B_PIN, -Speed);
      setMotorSpeed(Motor_R1_F_PIN, Motor_R1_B_PIN, -Speed);
      setMotorSpeed(Motor_R2_F_PIN, Motor_R2_B_PIN, 0);
      break;
    case RIGHT_FORWARD:
      setMotorSpeed(Motor_L1_F_PIN, Motor_L1_B_PIN, Speed);
      setMotorSpeed(Motor_L2_F_PIN, Motor_L2_B_PIN, 0);
      setMotorSpeed(Motor_R1_F_PIN, Motor_R1_B_PIN, 0);
      setMotorSpeed(Motor_R2_F_PIN, Motor_R2_B_PIN, Speed);
      break;
    case LEFT_BACKWARD:
      setMotorSpeed(Motor_L1_F_PIN, Motor_L1_B_PIN, -Speed);
      setMotorSpeed(Motor_L2_F_PIN, Motor_L2_B_PIN, 0);
      setMotorSpeed(Motor_R1_F_PIN, Motor_R1_B_PIN, 0);
      setMotorSpeed(Motor_R2_F_PIN, Motor_R2_B_PIN, -Speed);
      break;
    default:
      setMotorSpeed(Motor_L1_F_PIN, Motor_L1_B_PIN, 0);
      setMotorSpeed(Motor_L2_F_PIN, Motor_L2_B_PIN, 0);
      setMotorSpeed(Motor_R1_F_PIN, Motor_R1_B_PIN, 0);
      setMotorSpeed(Motor_R2_F_PIN, Motor_R2_B_PIN, 0);
      break;
  }
}
```

- Get infrared key value data

```
/**
 * @brief 获取红外键值数据 Get infrared key value data
 * @param 无 None
 * @retval 红外键值数据 Infrared key value
 */
uint8_t getIRValue() {
  if (IrReceiver.decode()) {
    uint8_t IR_Value = bitreverseOneByte(IrReceiver.decodedIRData.command);
    // Serial.println(IR_Value, HEX);
    IrReceiver.resume();
    return IR_Value;
  }
  return 0xFF;
}
```

- Set RGB display color

```
/**
 * @brief 设置RGB显示的颜色 Set RGB display color
 * @param color: 显示的颜色 Set the color
 * @retval 无 None
 */
void setRGBColor(ColorType color) {
  switch (color) {
    case RED:
      RGB.setPixelColor(0, RGB.Color(255, 0, 0));
      RGB.show();
      break;
    case GREEN:
      RGB.setPixelColor(0, RGB.Color(0, 255, 0));
      RGB.show();
      break;
    case BLUE:
      RGB.setPixelColor(0, RGB.Color(0, 0, 255));
      RGB.show();
      break;
    case YELLOW:
      RGB.setPixelColor(0, RGB.Color(255, 255, 0));
      RGB.show();
      break;
    case MAGENTA:
      RGB.setPixelColor(0, RGB.Color(255, 0, 255));
      RGB.show();
      break;
    case CYAN:
      RGB.setPixelColor(0, RGB.Color(0, 255, 255));
      RGB.show();
      break;
    case WHITE:
      RGB.setPixelColor(0, RGB.Color(255, 255, 255));
      RGB.show();
      break;
    default:
      RGB.setPixelColor(0, RGB.Color(0, 0, 0));
      RGB.show();
      break;
  }
}
```

- Set buzzer sound

```
/**
 * @brief 设置蜂鸣器声音 Set buzzer sound
 * @param 无 None
 * @retval 无 None
 */
void setBuzzerSound() {
  for (int i = 0; i < 50; i++) {
    digitalWrite(BUZZER_PIN, HIGH);
    delay(1);
    digitalWrite(BUZZER_PIN, LOW);
    delay(1);
  }
}
```

- Set the Logo breathing effect

```
/**
 * @brief 设置Logo呼吸效果 Set the Logo breathing effect
 * @param delayTime: 延时时间 Delay time
 * @param increment: 亮度递增数字 Brightness variation
 * @retval 无 None
 */
void setLogoBreath(unsigned int delayTime, unsigned int increment) {
  for (int Brightness = 0; Brightness <= 4095; Brightness += increment) {
    pwm.setPWM(LED_PIN, 0, Brightness);
    delay(delayTime);
  }
  for (int Brightness = 4095; Brightness >= 0; Brightness -= increment) {
    pwm.setPWM(LED_PIN, 0, Brightness);
    delay(delayTime);
  }
}
```

- Infrared control car

```
/**
 * @brief 红外遥控小车 Infrared control car
 * @param IrValue: 红外遥控器解析的键值 Infrared remote control key value
 * @retval 无 None
 */
void IR_Control(uint8_t IrValue) {
  switch (IrValue) {
    case Power:
      setCarMove(STOP, 0);
      setRGBColor(BLACK);
      break;
    case RGB_Light:
      uRgbState++;
      uRgbState = uRgbState % 8;
      setRGBColor(uRgbState);
      break;
    case Buzzer:
      setBuzzerSound();
      break;
      case CarForward:
        setCarMove(FORWARD, iCarSpeed);
        delay(150);
        setCarMove(STOP, 0);
        break;
      case CarBackward:
        setCarMove(BACKWARD, iCarSpeed);
        delay(150);
        setCarMove(STOP, 0);
        break;
      case CarLeft:
        setCarMove(LEFT, iCarSpeed);
        delay(150);
        setCarMove(STOP, 0);
        break;
      case CarRight:
        setCarMove(RIGHT, iCarSpeed);
        delay(150);
        setCarMove(STOP, 0);
        break;
      case CarLeftSpin:
        setCarMove(LEFT_ROTATE, iCarSpeed);
        delay(150);
        setCarMove(STOP, 0);
        break;
      case CarRightSpin:
        setCarMove(RIGHT_ROTATE, iCarSpeed);
        delay(150);
        setCarMove(STOP, 0);
        break;
      case Add:
        if (iCarSpeed > 255) {
          iCarSpeed = 200;
        } else if (iCarSpeed < 50) {
          iCarSpeed = 50;
        } else {
          iCarSpeed += 15;
        }
        break;
      case Sub:
        if (iCarSpeed > 255) {
          iCarSpeed = 200;
        } else if (iCarSpeed < 50) {
          iCarSpeed = 50;
        } else {
          iCarSpeed -= 15;
        }
        break;
      case Number_0: Serial.println("Number_0"); break;
      case Number_1: Serial.println("Number_1"); break;
      case Number_2: Serial.println("Number_2"); break;
      case Number_3: Serial.println("Number_3"); break;
      case Number_4: Serial.println("Number_4"); break;
      case Number_5: Serial.println("Number_5"); break;
      case Number_6: Serial.println("Number_6"); break;
      case Number_7: Serial.println("Number_7"); break;
```

```
    case Number_8: Serial.println("Number_8"); break;
    case Number_9: Serial.println("Number_9"); break;
    default: break;
  }
}
```

- Initialization Code

```
void setup() {
  IrReceiver.begin(IR_RECEIVE_PIN, ENABLE_LED_FEEDBACK);  // 初始化红外接收器 Initialize the infrared receiver
  Wire.begin();                                           // 初始化I2C通讯 Initialize I2C communication
  delay(1000);                                            // 如果小车功能异常，可以增加这个延时 If the function is abnormal, you can increase the delay
  pwm.begin();                                            // PWM初始化 Initialize the Pulse Width Modulation (PWM) library
  pwm.setPWMFreq(PWM_FREQUENCY);                          // 设置PWM频率 Set the PWM frequency
  RGB.begin();                                            // 初始化RGB Initialize RGB
  RGB.show();                                             // 刷新RGB显示 Refresh RGB display
  setCarMove(STOP, 0);                                    // 设置小车停止状态 Set the car to stop state
  pinMode(BUZZER_PIN, OUTPUT);                            // 设置蜂鸣器引脚输出模式 Set the buzzer pin output mode
  setLogoBreath(10, 25);                                  // 设置Logo呼吸效果 Set the Logo breathing effect
}
```

- Looping code

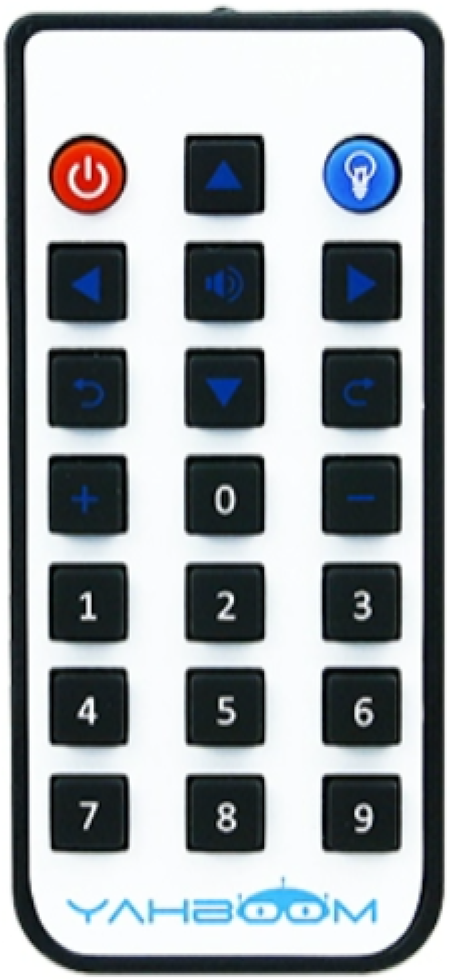
```
void loop() {
  // 根据红外接收器接收的数据进行小车控制 The car is controlled according to the data received by the infrared receiver
  IR_Control(getIRValue());
}
```

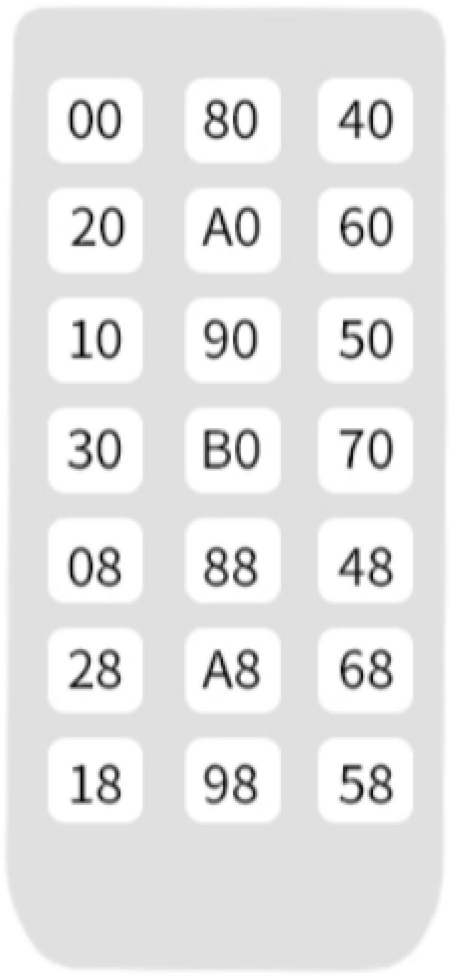
## Experimental results

After compiling the program successfully, upload the code to the Arduino Uno development board.

After the program is started, we can control the car's peripherals and motion status through the infrared remote control.

Use the infrared remote control to aim at the infrared receiver on the car expansion board to control the buttons. The APP only sets the functions of some buttons.





| Infrared remote control | Parse data | Function |
|---|---|---|
| Power | 0x00 | Stop all peripheral functions |
| RGB | 0x40 | Switch RGB colors |
| Buzzer | 0xA0 | Control the buzzer sound |
| Car Forward | 0x80 | Car forward |
| Car Backward | 0x90 | Car backward |
| Car Left | 0x20 | Car moves to the left |
| Car Right | 0x60 | Car moves to the right |
| Car Left Spin | 0x10 | Car spins left |
| Car Right Spin | 0x50 | Car spins right |
| Add | 0x30 | Speed increase |
| Sub | 0x70 | Speed reduction |

The infrared remote controller controls the movement of the car. You need to press and hold to perform corresponding control: for example, long press the car to move forward, and release the button to stop the car.

## Notes

In order to avoid infrared light interfering with the sensor, we need to use this module indoors.

```
If the car is abnormally controlled, we can burn the code [1.Car peripherals
foundation\10.Infrared_Remote_Print] to the car, and the serial port will print
the key value of the infrared remote control. We can use this program to
determine whether there is interference in the current environment.
```