

## Basic course ---9.IR control

### **!Note:**

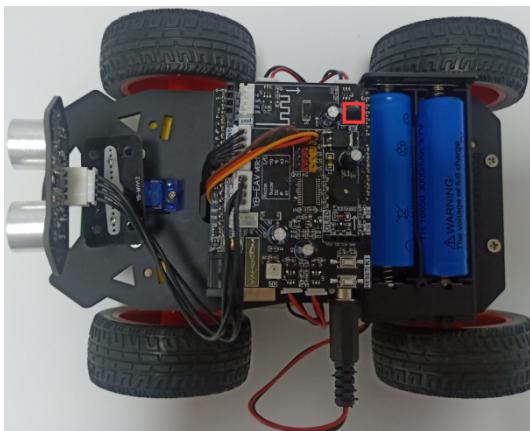
1. In order to avoid the interference of sunlight on infrared sensor, we need to carry out this experiment indoors.
2. During remote control, the infrared remote control needs to face the infrared receiver on the expansion board.

### **1. Learning goal**

In this lesson, we will learn how to control the car by infrared controller.

### **2. Preparation**

2.1 The position of the infrared receiver on the robot car. As shown below.



2.2 The pin of UNO board is connected the infrared receiver.

### **3. Principle of experimental**

From the hardware interface manual, we can know that infrared receiver are driven by Uno board.

Classification	Function	The number of Drive chip PCA9685	Drive Method	Connection with CPU	Uno board
Left Motor	Left front motor forward	LINB(13)	PCA9685	I2C_SDA/I2C_SCL	A4/A5
	Left front motor reverse	LINA(12)			
	Left rear motor forward	RINB(15)			
	Left rear motor reverse	RINA(14)			
Right Motor	Right front motor forward	LED10			
	Right front motor reverse	LED11			
	Right rear motor forward	LED8			
	Right rear motor reverse	LED9			
Servo	Control S1	LED0	Uno board drive directly		A0
	Control S2	LED1			
	Control S3	LED2			
	Control S4	S1 (3)			
LOGO light	Control bluelight	LED7			
Tracking sensor	Left tracking sensor				A1
	Middle tracking sensor				A2
	Right tracking sensor				12
Ultrasonic sensor	Ultrasonic Echo				11
	Ultrasonic RGB light				7
Key	K1				0
IR	IR control				1
Bluetooth interface	RX				6
	TX				10
On board RGB Light	RGB Light on expansion board				A3
Buzzer	Control buzzer				

#### 4. About code

For the code of this course, please refer to: [IR\\_Control\\_Car.ino](#) in the [IR\\_Control\\_Car](#) folder.

```
#include <Arduino.h>
#include <Adafruit_PWMServoDriver.h>
#include <Adafruit_NeoPixel.h>
#include <IRremote.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include "RGBLed.h"

#define RGB_GREEN    0xFF0000    //Define different
color(green,red,blue)
#define RGB_RED     0x00FF00
#define RGB_BLUE    0x0000FF
#define RGB_YELLOW   0xFFFF00
#define RGB_PURPLE   0x00FFFF
#define RGB_WHITE    0xFFFFFFFF
```

```

#define RGB_CYAN 0xFF00FF
#define RGB_OFF 0x000000
#define SERVOMIN 150 // this is the 'minimum' pulse length count (out
of 4096)
#define SERVOMAX 600 // this is the 'maximum' pulse length count (out
of 4096)
const int RgbPin = 11; //Define pin of Ultrasonic RGB light
RGBLed mRgb(RgbPin,2);
#define PIN 6 //Define the pins of the RGB light
#define MAX_LED 1 //Just one RGB light on the car
Adafruit_NeoPixel strip = Adafruit_NeoPixel(MAX_LED, PIN, NEO_RGB +
NEO_KHZ800);
Adafruit_PWM_Servo_Driver pwm = Adafruit_PWM_Servo_Driver(0x40);

/*Car running status enumeration*/
const typedef enum {
    enRUN = 1,
    enBACK,
    enLEFT,
    enRIGHT,
    enSPINLEFT,
    enSPINRIGHT,
    enSTOP
} enCarState;

const typedef enum {
    enServoL = 1,
    enServoR,
    enServoS
} enServoState;

const char enServo[] = {0, 1, 2, 3};

const int RECV_PIN = A3; //Infrared integrated receiver is connected to the A3
on the UNO
IRrecv irrecv(RECV_PIN);
decode_results results;
unsigned long last = millis();
int Servo_LR = 90;
static int CarSpeedControl = 150;
static int g_carstate = enSTOP;
static int g_ServoState = enServoS; //1Left 2Right 3Stop
const int buzzer = 10; //Buzzer is connected to the 10 on the UNO
.....

```

```

void loop()
{
    // put your main code here, to run repeatedly:
    if (irrecv.decode(&results))
    {
        switch (results.value)
        {
            //Power switch button
            case 0X00FF00FF:
                Onboard_RGB_OFF();
                Ultrasonic_RGB_OFF();
                break;

            //Light button
            case 0x00FF40BF:
                Onboard_RGB(255,255,255);
                mRgb.setColor(0,RGB_WHITE);
                Rgb.show();
                break;
        ......

            //Beep button
            case 0x00FFA05F: whistle();    break;
            //Num 7 button
            case 0x00ff18e7: g_ServoState = enServoL;  break;
            //Num 8 button
            case 0x00ff9867:  break;
            //Num 9 button
            case 0x00ff58a7: g_ServoState = enServoR;  break;
            //Advance button
            case 0x00FF807F: g_carstate = enRUN;   break;
            //Left button
            case 0x00FF20DF: g_carstate = enLEFT; break;
            //Right button
            case 0x00FF609F: g_carstate = enRIGHT; break;
            //Back button
            case 0x00FF906F: g_carstate = enBACK; break;
            //Spin left button
            case 0x00FF10EF: g_carstate = enSPINLEFT; break;
            //Spin right button
            case 0x00FF50AF: g_carstate = enSPINRIGHT; break;
            default:   break;
        }
        last = millis();
        irrecv.resume();    //Receive the next code
    }
}

```

```

else if (millis() - last > 120)
{
    g_carstate = enSTOP;
    g_ServoState = enServoS;
    last = millis();
}
switch (g_carstate)
{
//case enSTOP: brake(); break;
case enRUN: advance(CarSpeedControl); break;
case enLEFT: left(CarSpeedControl); break;
case enRIGHT: right(CarSpeedControl); break;
case enBACK: back(CarSpeedControl); break;
case enSPINLEFT: spin_left(CarSpeedControl); break;
case enSPINRIGHT: spin_right(CarSpeedControl); break;
default: brake(); break;
}
switch (g_ServoState)
{
case enServoL:
    Servo_LR++;
    if (Servo_LR > 180)
    {
        Servo_LR = 180;
    }
    Servo180(1, Servo_LR);
    delay(5);
    break;
case enServoR:
    Servo_LR--;
    if (Servo_LR < 0)
    {
        Servo_LR = 0;
    }
    Servo180(1, Servo_LR);
    delay(5);
    break;
default:
    break;
}
}

```

## 5. Compiling and downloading code

5.1 We need to open the **IR\_Control\_Car.ino** file by Arduino IDE software.

Then click “√” under the menu bar to compile the code, and wait for the word “**Done compiling** ” in the lower left corner, as shown in the figure below.

The screenshot shows the Arduino IDE interface. The top menu bar includes File, Edit, Sketch, Tools, and Help. Below the menu is a toolbar with various icons. The main workspace shows a sketch named "IR\_Control\_Car" containing C++ code. The code includes header file includes like Arduino.h, Adafruit\_PWMServoDriver.h, Adafruit\_NeoPixel.h, IRremote.h, Adafruit\_GFX.h, Adafruit\_SSD1306.h, and RGBLed.h. It also defines constants for colors: RGB\_GREEN (0xFF0000) and RGB\_RED (0x00FF00). A status bar at the bottom displays the message "Done compiling." in a teal-colored box, which is highlighted with a red rectangle.

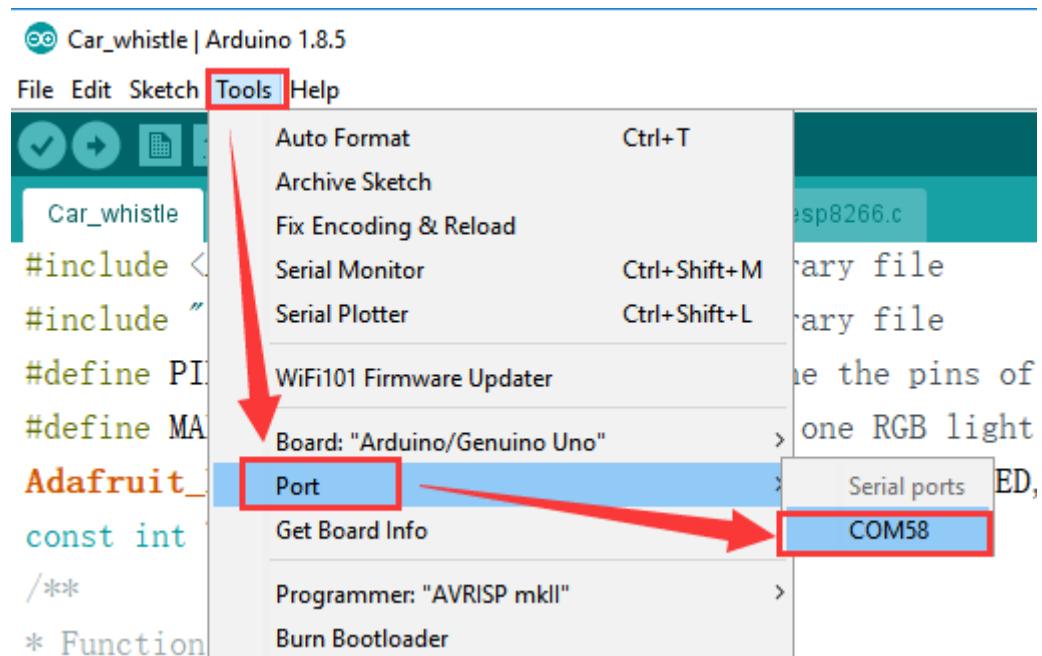
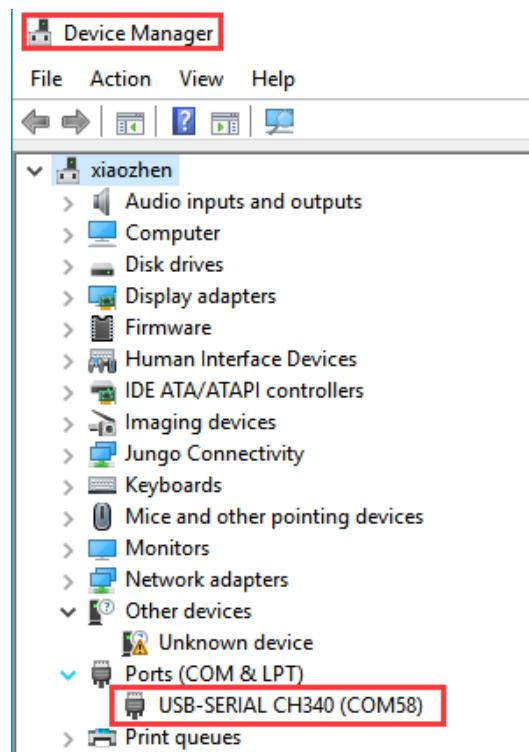
```

IR_Control_Car | Arduino 1.8.5
File Edit Sketch Tools Help
IR_Control_Car RGBLed.cpp RGBLed.h
* @brief
* @details
* @par History NO
*/
#include <Arduino.h>
#include <Adafruit_PWMServoDriver.h>
#include <Adafruit_NeoPixel.h>
#include <IRremote.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include "RGBLed.h"

#define RGB_GREEN    0xFF0000      //Define different color(green, red, blue)
#define RGB_RED     0x00FF00

```

5.2 In the menu bar of Arduino IDE, we need to select 【Tools】---【Port】--- selecting the port that the serial number displayed by the device manager just now, as shown in the figure below.



5.3 After the selection is completed, you need to click “→”under the menu bar to upload the code to the UNO board. When the word “**Done uploading**” appears in the lower left corner, the code has been successfully uploaded to the UNO board, as shown in the figure below.

IR\_Control\_Car | Arduino 1.8.5

File Edit Sketch Tools Help

IR\_Control\_Car RGBLed.cpp RGBLed.h

```
* @brief
* @details
* @par History NO
*/
#include <Arduino.h>
#include <Adafruit_PWMServoDriver.h>
#include <Adafruit_NeoPixel.h>
#include <IRremote.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include "RGBLed.h"

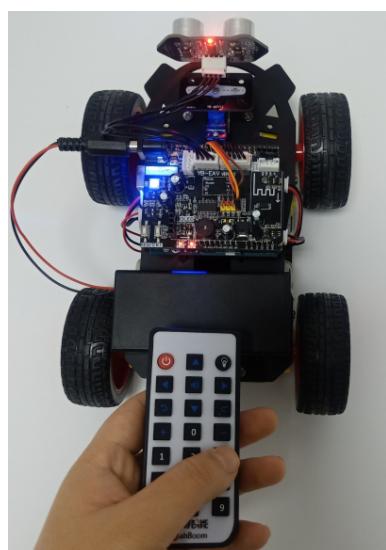
#define RGB_GREEN 0xFF0000 //Define different color(green, red, blue)
#define RGB_RED 0x00FF00
```

Done uploading.

Sketch uses 11996 bytes (37%) of program storage space. Maximum is 32256 bytes.  
Global variables use 843 bytes (41%) of dynamic memory, leaving 1205 bytes for loc

## 6. Experimental phenomena

After the program is downloaded, we can control robot car by infrared controller. As shown below.



The following functions are controlled by the infrared remote controller.

Num 0,8 button, no function defined at the moment

