Color recognition

We can view the WiFi camera screen through the APP and see the effect of the car's color recognition.

```
Since the previous tutorial has explained the basic knowledge of all car modules,
the car control tutorial will not be repeated, and the main focus is on the car
function implementation ideas!
```

Device connection

Hardware connection

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

Before this, you need to unplug the connection between the WiFi camera and Roboduino, otherwise the serial port will be occupied and the computer cannot recognize the serial port number of Arduino Uno.

Software connection

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

Implementation ideas

Analyze the data sent by the WiFi camera through the serial port and control the car according to the data.

Code analysis

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

1. esp32_wifi

- Define a macro `WIFI_MODE` to select different WiFi modes based on preprocessor directives (#if)

```
#if MODE_AP_STA
#define WIFI_MODE '2'

#elif MODE_STA
#define WIFI_MODE '1'

#elif MODE_AP
#define WIFI_MODE '0'

#else
#define WIFI_MODE '2'
#endif
```

- Enumerate AI modes, including cat and dog detection, face detection, color detection, etc.

```
typedef enum AI_mode_t
{
    Nornal_AI = 0, //不检测 No detection
    Cat_Dog_AI,   //猫狗检测 Cat and Dog Detection
    FACE_AI,      //人物检测 People Detection
    COLOR_AI ,   //颜色识别 Color recognition
    REFACE_AI,   //人脸识别 Face Recognition
    QR_AI = 5,   //二维码识别 QR code recognition
    AI_MAX      //最大值 Maximum
}AI_mode;
```

- `SET_ESP_WIFI_MODE` function sets the WiFi mode of the ESP32 camera.

```
void SET_ESP_WIFI_MODE(void) //设置模式选择 Setting Mode Selection
{
  //选择模式STA+AP模式共存 Select mode STA+AP mode coexistence
  sprintf(send_buf,"wifi_mode:%c",WIFI_MODE);
  ESPWIFISerial.print(send_buf);
  memset(send_buf,0,sizeof(send_buf));

  delay(2000);//等待复位重启成功 Wait for reset to restart successfully
}
```

- The `SET_ESP_AI_MODE` function sets the AI mode of the ESP32 camera.

```
void SET_ESP_AI_MODE(AI_mode Mode) //设置AI模式 Setting AI Mode
{

  sprintf(send_buf,"ai_mode:%d",Mode);
  ESPWIFISerial.print(send_buf);
  memset(send_buf,0,sizeof(send_buf));
  runmode = Mode;

  delay(2000);//等待复位重启成功 Wait for reset to restart successfully
}
```

2. esp_key

- The `key_state` function is used to detect short press and long press events of a button and return the corresponding state value.

- 0 means the button is not pressed.
- 1 means short press.
- 2 means long press.

```
//长按和短按
//长按:2 短按：1 不按：0
//Long press and short press
//Long press: 2 Short press: 1 No press: 0
int key_state()
{
  int touchReading = digitalRead(KEY_PIN);
  int8_t key_state = 0;
  if (touchReading != lastTouchState) {
    lastDebounceTime = millis();
  }

  if ((millis() - lastDebounceTime) > debounceDelay) {
    if (touchReading != touchState) {
      touchState = touchReading;

      if (touchState == LOW) {
        // 触摸按下 Touch Press
        touchDownTime = millis();
        touchInProgress = true;
        singleClick = false;
        longPress = false;
      } else {
        // 触摸释放 Touch Release
        if (touchInProgress) {
          if ((millis() - touchDownTime) < longPressDelay)
          {
            // 短按判定 Short press judgment
            if (!singleClick)
            {
              singleClick = true;
            }
          }
          else
          {
            // 长按判定 Long press judgment
            longPress = true;
          }
          touchInProgress = false;
        }
      }
    }
  }

  if (singleClick) {
    //ESPWIFISerial.println("1");
    key_state = 1;
    singleClick = false;
  }
  if (longPress) {
    //ESPWIFISerial.println("2");
    key_state = 2;
    longPress = false;
  }

  lastTouchState = touchReading;

  return key_state;

}
```

- The `send_key` function sends different command strings according to the value of Virtual_key and outputs them through the `ESPWIFISerial.print` function.

```
//esp32 虚拟按键 Virtual buttons
uint8_t Virtual_key = 1; //1: Send KEY_MENU 2: Send KEY_PLAY 3: KEY_DOWN (3 only works for color recognition)
static void send_key(void)
{
  switch(Virtual_key)
  {
    case 1:ESPWIFISerial.print("KEY_MENU");break; //RGB灯亮绿色 RGB light turns green
    case 2:ESPWIFISerial.print("KEY_PLAY");break; //RGB灯亮蓝色 RGB light turns blue
    case 3:ESPWIFISerial.print("KEY_DOWN");break; //RGB灯亮红色 RGB light turns red
    case 4:ESPWIFISerial.print("KEY_PLAY");break; //RGB灯亮蓝色 /RGB light turns blue
  }
}
```

- The `key_RGB` function sets the color of the RGB LED according to the value of Virtual_key.

  The value of Virtual_key determines the color of the LED

  - 1: Set to green (GREEN)
  - 3: Set to red (RED)
  - 2 or 4: Set to blue (BLUE)

```
static void key_RGB(void)
{
  switch(Virtual_key)
  {
    case 1:setRGBColor(GREEN);break;
    case 3:setRGBColor(RED);break;

    case 2:
    case 4:setRGBColor(BLUE); ;break;
  }
}
```

- The `key_goto_state` function performs different operations depending on the state of the key (short press or long press). The `key_state` function is called to get the state of the key, and the operation to be performed is determined based on the state of the key and the system operation mode (runmode).
  - If you long-press KEY1 and call the key_RGB function to change the color of the LED, the value of `Virtual_key` will be incremented. If the color recognition mode is running and

the value of Virtual_key exceeds 4, Virtual_key will be reset to 1;
    - If you short-press KEY1 and run the color recognition mode, call the `send_key` function to send the corresponding command.

```
void key_goto_state(void)
{
  uint8_t key_oo = key_state();
    if(key_oo == 2)//长按 Long press
    {
      Virtual_key ++ ;//改变发送的命令 Change the command sent
      if(runmode == COLOR_AI) //只有颜色识别才发送KEY_DOWN Only send KEY_DOWN for
color recognition
      {
         if(Virtual_key > 4)
        {
          Virtual_key = 1;
         }
      }
      else //人物识别 Person Recognition
      {
         if(Virtual_key > 2)
        {
          Virtual_key = 1;
         }
      }
          key_RGB();
     }

     else if(key_oo == 1)//短按 Short press
     {
        if(runmode == COLOR_AI || runmode==REFACE_AI )//Only color recognition
and face recognition interact
        {
          send_key();//发送数值 Sending Values
        }
     }
}
```

3. bsp_RGB

- Include libraries such as `Adafruit_NeoPixel.h` and motor_car.hpp header files

```
#include <stdio.h>
#include <string.h>
#include <Arduino.h>
#include <Adafruit_NeoPixel.h>
#include "motor_car.hpp
```

- Enumerate RGB light status, off, red, green, blue, yellow, etc.

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

- Define RGB light control pins

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

- Create an instance of `Adafruit_NeoPixel`

```
// 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);
```

- Initialize RGB light control

```
void RGB_init()
{
    RGB.begin();  // 初始化RGB Initialize RGB
    RGB.show();   // 刷新RGB显示 Refresh RGB display
    if(runmode == COLOR_AI || runmode==REFACE_AI )
    {
        setRGBColor(GREEN); //默认关闭 Default off

     }
     else
     {
        setRGBColor(BLACK); //默认关闭 Default off
     }

}
```

- Set the color of the RGB light

```
/**
 * @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;
  }
}
```

4. color_decect_esp32.ino

- Import header files, including the implementation of ESP32's WiFi function, motor control, button management, and RGB light control.

```
#include <stdio.h>
#include "esp32_wifi.hpp"
#include "motor_car.hpp"
#include "esp_key.hpp"
#include "bsp_RGB.hpp"
```

- A macro `AI_set_mode` is defined here, indicating that the current AI mode is `COLOR_AI` (color recognition).

`cmd_flag` is an external variable used to indicate the current command status.

```
#define AI_set_mode COLOR_AI  //设置AI模式 Setting AI Mode
#define BUZZER_PIN 10

extern uint8_t cmd_flag;
```

- `mode_chage()` is a mode switching function that sets the value of `cmd_flag` according to the current operating mode to indicate different processing modes.

```
void mode_chage()
{
  if(runmode == Nornal_AI)
  {
    cmd_flag = 2;//进入透传模式 Enter transparent mode
  }
  else if(runmode == REFACE_AI )
  {
    cmd_flag = 3;//解析人脸识别数据 Parsing facial recognition data
  }
  else if(runmode == QR_AI )
  {
    cmd_flag = 4;//解析二维码数据  Parsing QR code data
  }
  else
  {
    cmd_flag = 5;//其它AI模式数据  Other AI mode data
   }
}
```

- When the `decect_color()` function is used to recognize the learned color, the RGB light displays 7 colors.

```
void decect_color(void)
{
  for (uint8_t i = 1;i<=7;i++)
  {
    setRGBColor(i);
    delay(200);
  }
   setRGBColor(BLACK); //关掉 Turn off
}
```

- The `setup()` function is the entry point for initializing the program, including the initialization of the buzzer, button, serial port, WiFi, AI mode, motor and RGB light, and calls the `mode_chage` function to set the command flag.

```
void setup()
{
  pinMode(BUZZER_PIN, OUTPUT); //蜂鸣器初始化 Buzzer initialization
  init_key();//按键初始化 Button initialization
  serial_init();//串口初始化 Serial port initialization
  SET_ESP_WIFI_MODE();//设置wifi模式 Set Wi-Fi mode

  SET_ESP_AI_MODE(AI_set_mode);//设置AI模式 Setting AI Mode

  Motor_init();//电机初始化 Motor initialization
  RGB_init();//RGB初始化 RGB initialization

  mode_chage();//根据模式选择解析数据办法 Choose the method to parse data according
to the mode

  //setCarMove(FORWARD,35); //35-255
}
```

- In the main loop, when the running mode is color AI or face AI, the state detection of the virtual button is called. If new data is received, newlines is reset to 0, and the `decect_color()` function is called for color recognition.

```
void loop()
{
  //Key logo and RGB light color Only these two modes will use key interaction
  if(runmode == COLOR_AI || runmode==REFACE_AI )
  {
    key_goto_state();//虚拟按键 Virtual buttons
  }

  if(newlines == 1)//接收到新数据 New data received
  {
     newlines = 0;
     if(esp32_ai_msg.cx != 160)//因为协议，此值过滤就好 Because of the protocol,
this value is filtered
     {
         //识别学习的颜色 Color recognition learning
        decect_color();
     }
  }
}
```

Experimental results

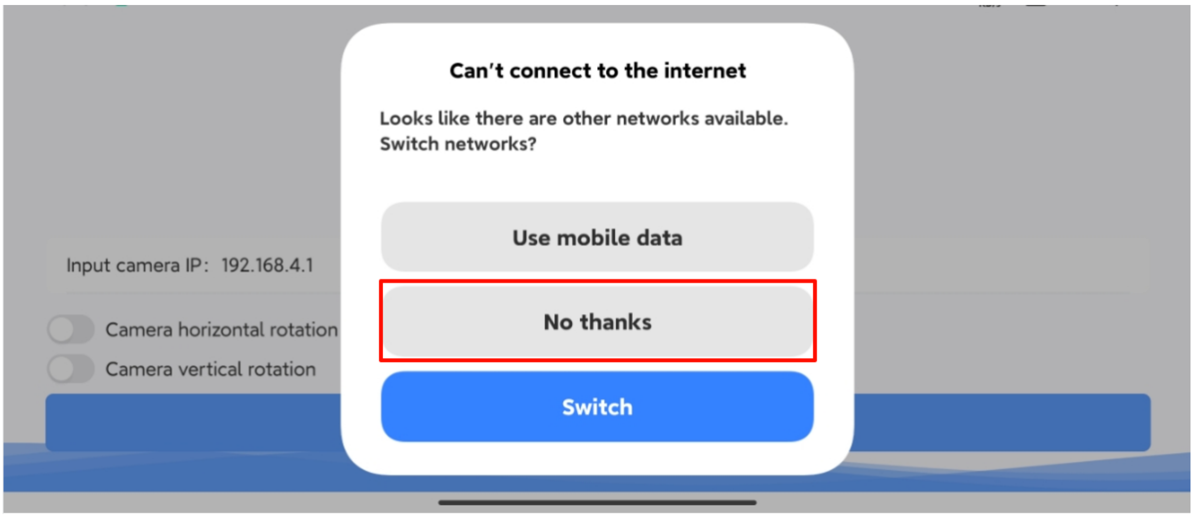
After compiling the program successfully, upload the code to the Arduino Uno development board, disconnect the car from the computer, and connect the WiFi camera to the serial port on the expansion board.

After the program starts, we can observe the camera screen through the YahboomCAM APP.

APP connection

Connect the mobile phone to the hotspot of the WiFi camera (the name of the built-in hotspot: Yahboom_ESP32_WIFI), and then open the YahboomCam software.

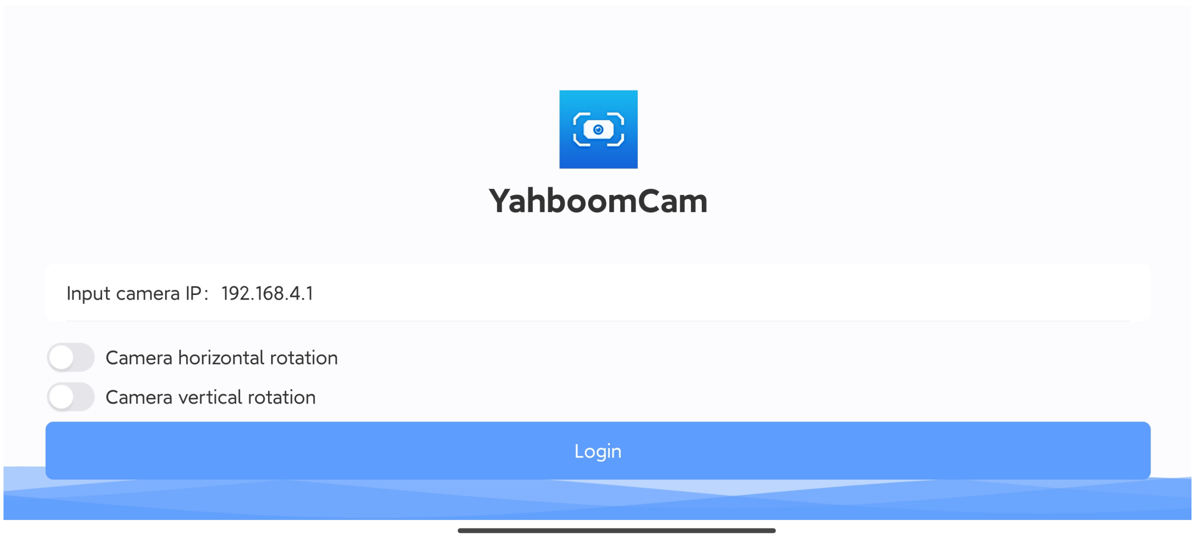
```
Some mobile phones will prompt for connecting to a hotspot without a network, and
we need to click to keep connected!
```


Can't connect to the internet

Looks like there are other networks available. Switch networks?

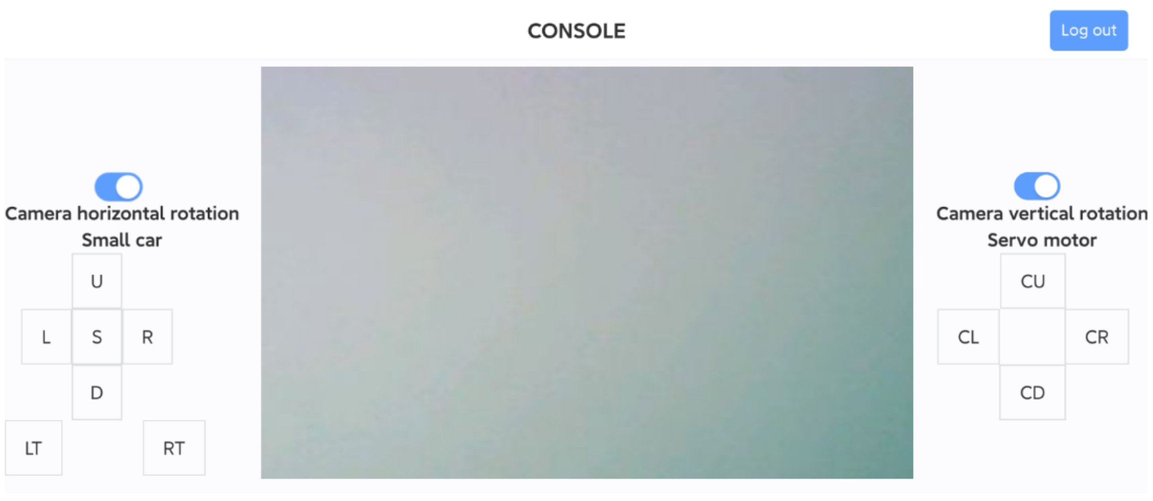
Enter IP:192.168.4.1 in the YahboomCam software, then click Login to enter the APP control interface.

```
The IP of the WiFi camera's built-in hotspot is 192.168.4.1
```



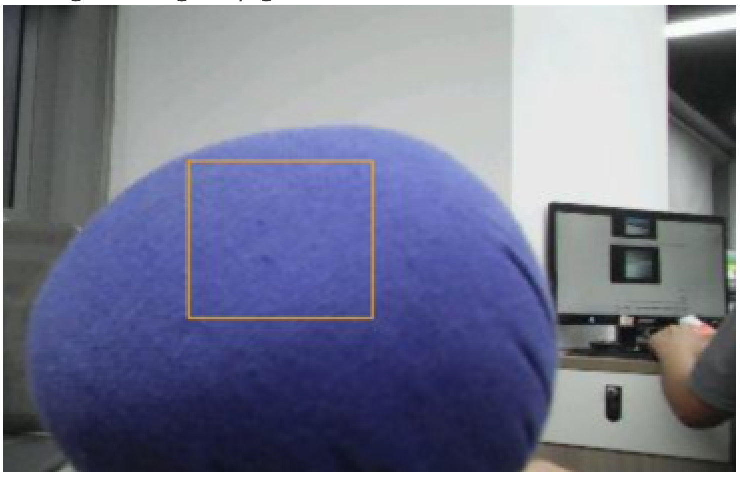
After entering the APP interface, the APP will display the camera screen.

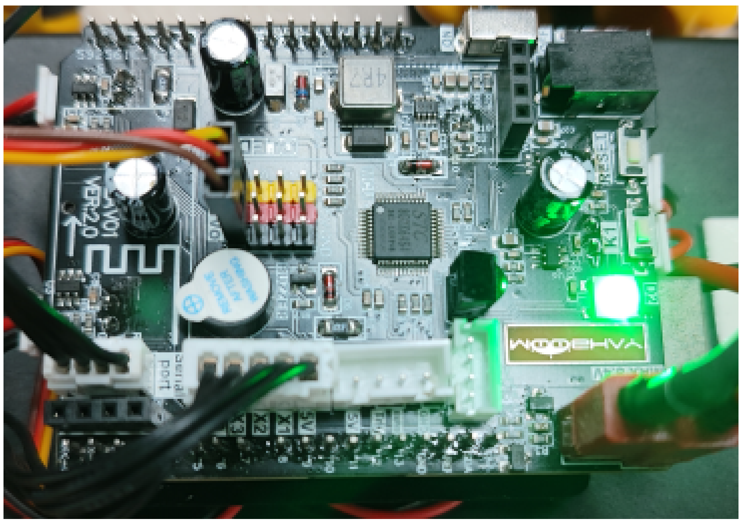
```
If there is no display, check whether the phone is connected to the WiFi camera
hotspot normally.
```



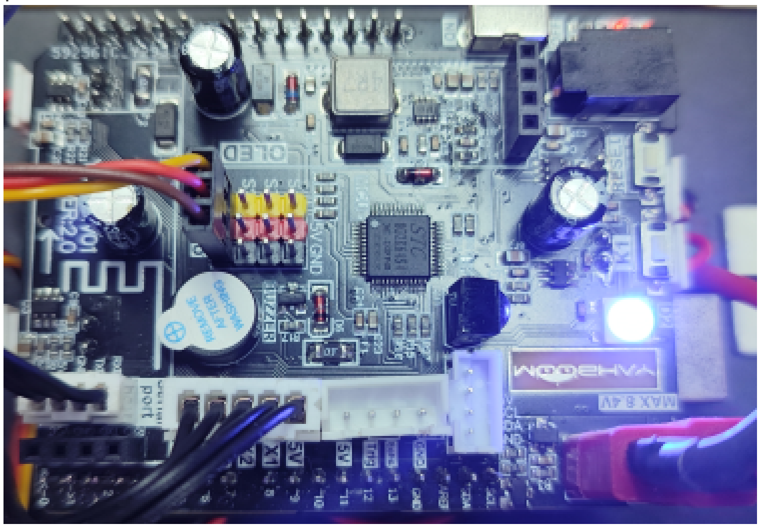
Effect

First, short press KEY1, and the admission box will appear. Select the color to be learned, and the RGB light will light up green;

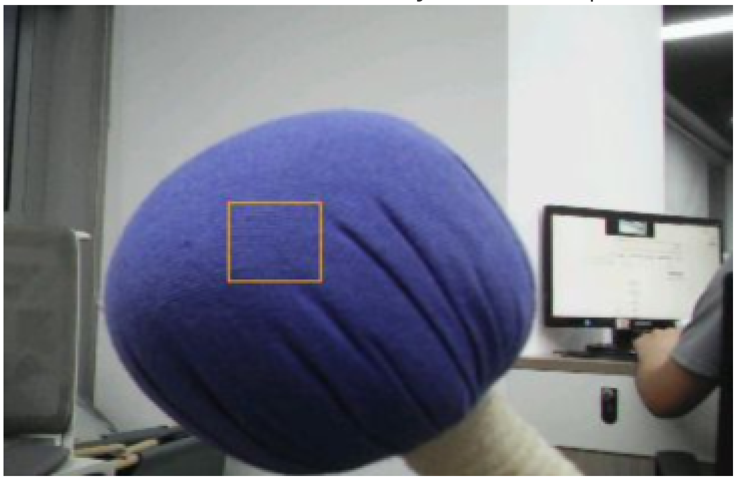


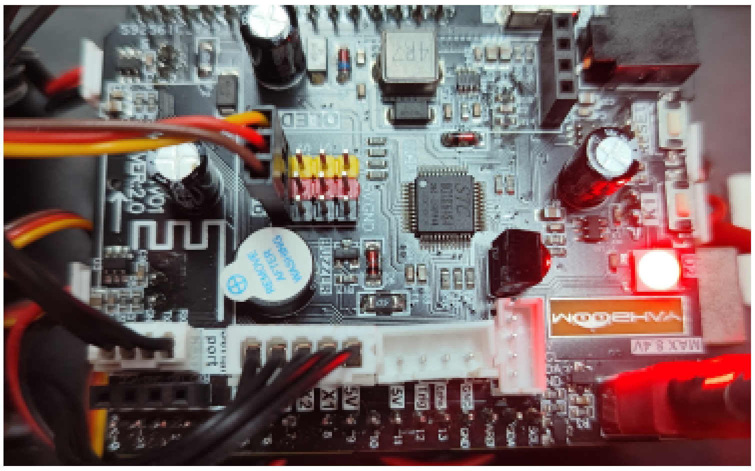


Then long press KEY1, and after releasing it, the color of the RGB light will turn blue, then short press KEY1 to confirm the color to be learned;



If you need to reduce the size of the recognition box, long press KEY1 (about 1s) and observe whether the RGB is red. If it is red, you can short press KEY1 to reduce the box size;





When the learned color is recognized, the RGB light displays 7 colors of running lights.

Notes

1. The screen needs to be turned on to start the recognition. If the screen is turned off, the recognition will also be turned off.