Color Follows

We can view the WiFi camera screen through the APP, and the car will respond accordingly based on the recognition results.

```
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
implementation ideas of the car functions!
```

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. motor_car

- Contains libraries such as `Wire`, `Adafruit_PWMServoDriver` and esp32_wifi.hpp header file

```
#include <stdio.h>
#include <string.h>
#include <Arduino.h>
#include <Wire.h>                       // Include Wire (I2C) communication
library Include Wire library
#include <Adafruit_PWMServoDriver.h>    // Include Adafruit PWMServoDriver
library Include Adafruit PWMServoDriver library
#include "esp32_wifi.hpp"
```

- Enumerate the car status: stop, move forward, move backward, turn left, turn right, etc.

```
// 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,
};
```

- Define the motor control pins, the underlying chip I2C address and the motor frequency

```
// 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

// 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
```

- Create an instance of `Adafruit_PWMServoDriver`

```
// 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);
Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver(Bottom_Layer_Driver_ADDR);
```

- Motor control initialization

```
void Motor_init() {
  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
  setCarMove(STOP, 0);             // 设置小车停止状态 Set the car to stop state
}
```

- Setting the speed of a single motor

```
/**
 * @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's 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;
  }
}
```

3. follow_pid

- Define the gain parameters of the PID controller on the X and Y axes, and declare floating-point and integer variables used to store the variables required for PID control on these axes.

```
#define KPx  (0.6) //0.6
#define KDx  (0.001) //0.001

#define KPy  (0.008) //0.005
#define KIy  (0.000001)
#define KDy  (0.0001) //0.0001

float KP_x,KD_x; //PID in x-axis direction
float KP_y,KI_y,KD_y; //PID in y-axis direction

int error_x,error_last_x;
int error_y,error_last_y,integral_y;
```

- Define two functions `init_x_PID` and `init_y_PID` to initialize the PID controllers in the x-axis and y-axis directions.

```
void init_x_PID(void)
{
    KP_x = KPx;
    KD_x = KDx;

    error_x = 0;
    error_last_x = 0;

    esp32_ai_msg.cx = 160;

}

void init_y_PID(void)
{
    KP_y = KPy;
    KI_y = KIy;
    KD_y = KDy;

    error_y = 0;
    error_last_y = 0;
    integral_y = 0;

    esp32_ai_msg.cy = 120;
```

```
    esp32_ai_msg.area = 1000; //This value should be suitable for the distance,
the area is reduced by 10 times

}
```

- Define two functions `PID_count_x` and `PID_count_y` to calculate the PID control output in the x-axis and y-axis directions respectively.

```
int PID_count_x(void)
{
    int pwmoutx = 0;
    error_x =esp32_ai_msg.cx - 160;

     if(myabs(error_x)<35)
    {
        //重置 Reset
        error_last_x = 0;
        return 0;
    }

    //位置式 Position
    pwmoutx = error_x * KP_x + error_last_x *KD_x;

    error_last_x = error_x;

    return pwmoutx;
}


int PID_count_y(void)
{
    int pwmouty = 0;
    if(esp32_ai_msg.area >6000)
    {
      return 0; //面积最大抛弃  Largest abandoned area
     }

    error_y = 1000 - esp32_ai_msg.area;

    if(myabs(error_y)<600)
    {
        //重置 Reset
        error_last_y = 0;
        integral_y = 0;

        return 0;
    }

    integral_y += error_y;
    if(integral_y > 1000)
    {
        integral_y = 1000;
    }
    else if(integral_y < -1000)
    {
        integral_y = -1000;
    }
```

```
    //位置式 Position
    pwmouty = error_y * KP_y + integral_y *KI_y +error_last_y *KD_y;

    error_last_y = error_y;

    return pwmouty;
}
```

- The `Car_Follow_Start()` function starts the car following function, obtains the control signal by calling the PID calculation function, and sets the motor speed.

```
void Car_Follow_Start(void)
{
    int motorL,motorR;
    int PWMx,PWMy;
    PWMx = PID_count_x();
    PWMy = PID_count_y();

    Set_speed(PWMy,PWMx); //传入值即可 Just pass in the value

}
```

4. 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;
  }
}
```

5. color_follow.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
   }
}
```

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

```
void setup()
{
  // PID Initialization
  init_x_PID();
  init_y_PID();

  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


}
```

- In the main loop, when the operation mode is color recognition or face recognition, the state detection of the virtual button is called. If new data is received, newlines is reset to 0, and the `Car_Follow_Start()` function is called to control the speed of the car.

```
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;
  }

  //pid calculate wheel speed
  Car_Follow_Start();



}
```

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.

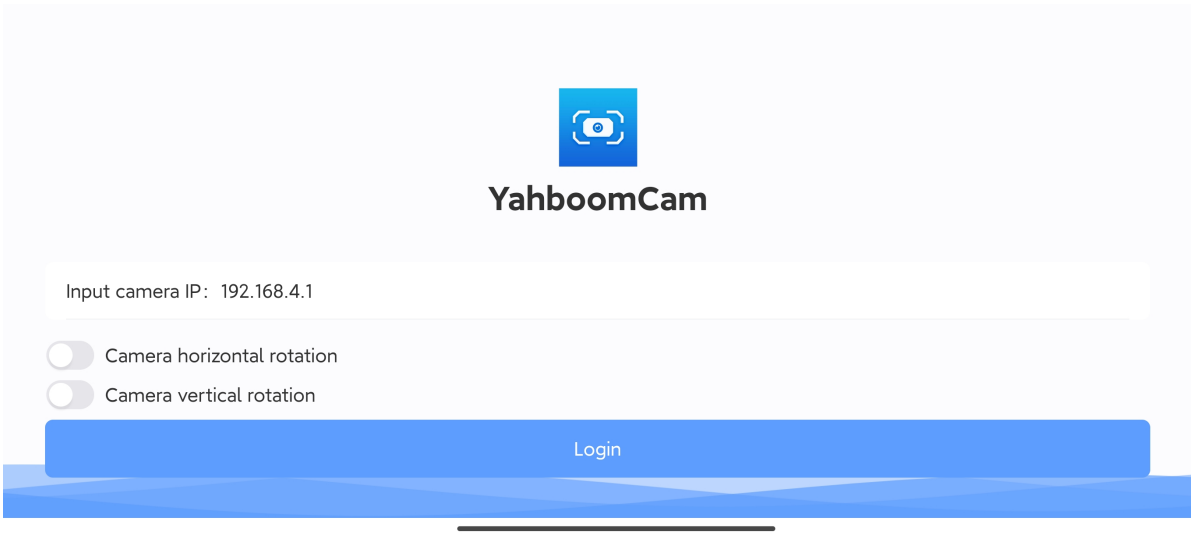
APP connection

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!
```

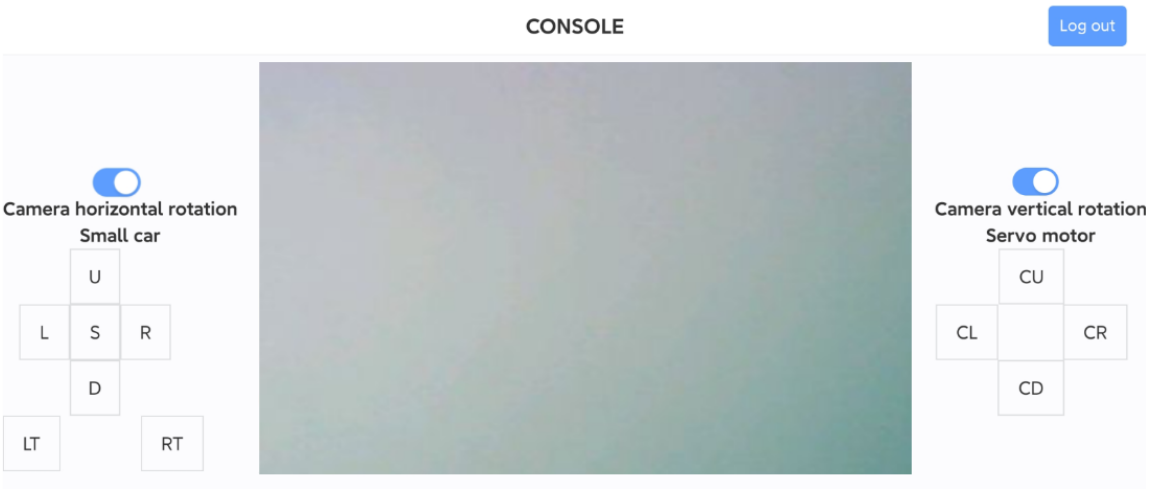
Enter IP:192.168.4.1 in the YahboomCam software, then click Login to enter the APP control interface.

```
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.

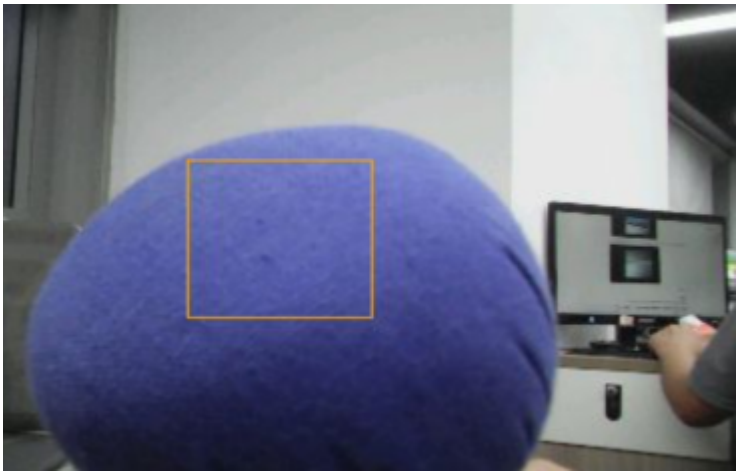
```
If no screen is displayed, check whether the phone is connected to the WiFi camera hotspot normally
```

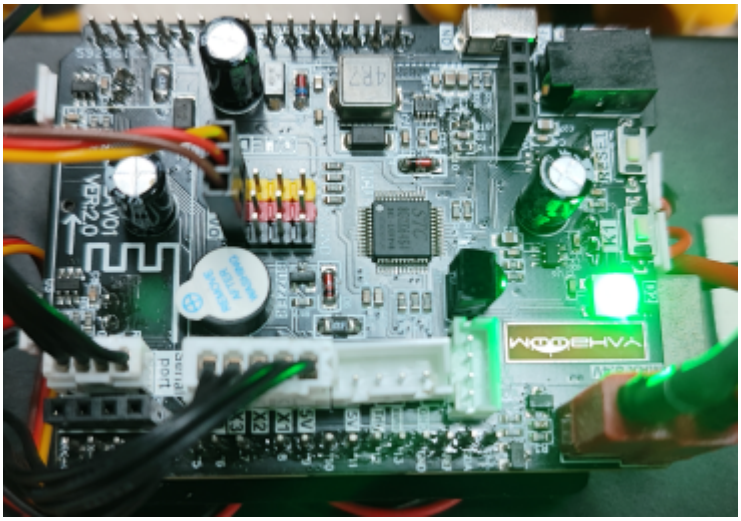


Effect

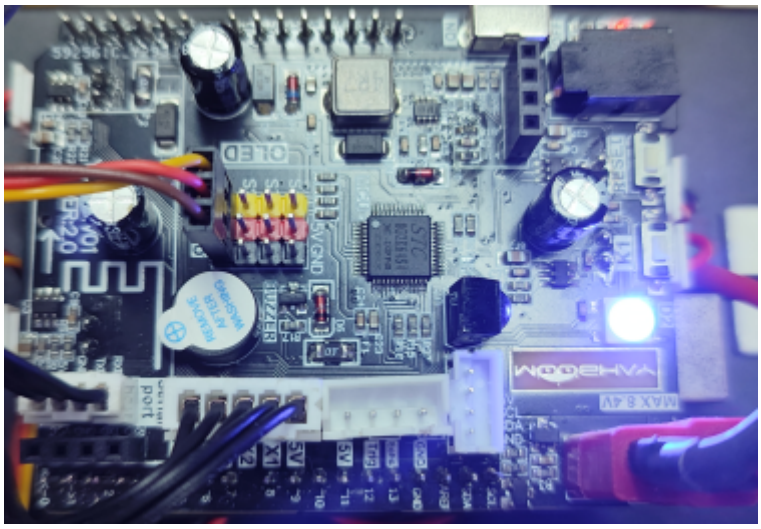
The color admission process is consistent with the color recognition case. After recognizing the learned color, the car follows the color movement.

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

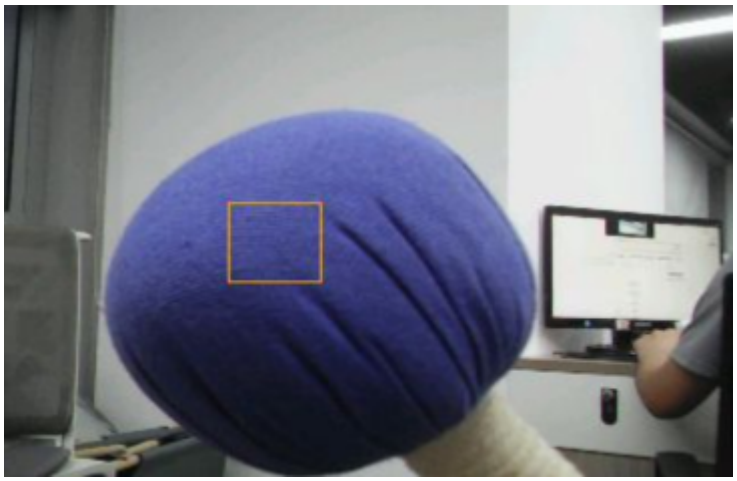


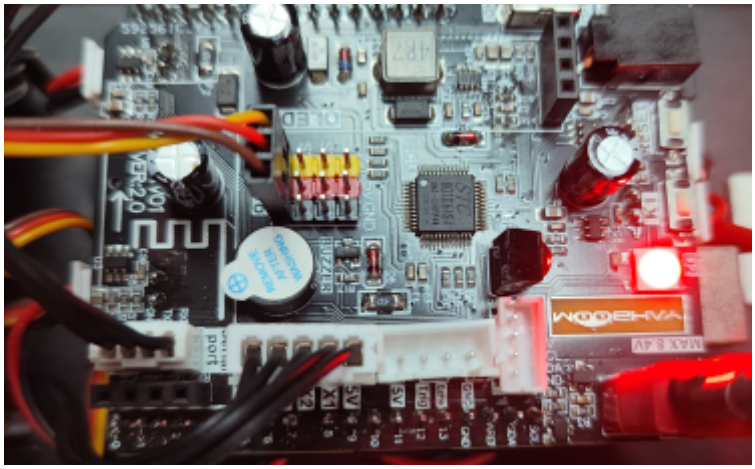


Then long press KEY1, after releasing, the color of the RGB light will be blue, then short press KEY1 to confirm the color to be learned;



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 car follows the color.

Notes

1. You need to turn on the screen to start recognition. If the screen is turned off in the middle, the recognition will also be turned off.
2. Make sure the direction of the camera screen is positive. You need to turn on the vertical rotation and horizontal rotation switches on the APP, otherwise the following direction will be opposite.