QR code instructions

We can view the WiFi camera screen through the APP, and the car will complete the corresponding instructions according to the different results.

```
Since the previous tutorials have 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. QRcommand.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"
```

- Here a macro `AI_set_mode` is defined, indicating that the current AI mode is `QR_AI` (QR code recognition).

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

```
#define AI_set_mode QR_AI  //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 `QRcommand()` function is used to recognize a QR code. Based on the recognized information, the car performs corresponding actions at a certain speed.
    - When the recognized information is "go forward", the car moves forward;
    - When the recognized information is "go back", the car moves backward;
    - When the recognized information is "turn left", the car turns left;
    - When the recognized information is "turn right", the car turns right.

```
//The instructions here are changed according to your own QR code
void QRcommand()
{
    if(strcmp(QR_msg.QR_msg,"go forward")==0)
    {
        setCarMove(FORWARD,55);
        delay(700);
        setCarMove(STOP,0);
    }
   else if(strcmp(QR_msg.QR_msg,"go back")==0)
   {
        setCarMove(BACKWARD,55);
        delay(700);
        setCarMove(STOP,0);
   }
    else if(strcmp(QR_msg.QR_msg,"turn left")==0)
   {
        setCarMove(LEFT_ROTATE,55);
        delay(700);
        setCarMove(STOP,0);
   }
    else if(strcmp(QR_msg.QR_msg,"turn right")==0)
   {
        setCarMove(RIGHT_ROTATE,55);
        delay(700);
        setCarMove(STOP,0);
   }

   memset(QR_msg.QR_msg,0,strlen(QR_msg.QR_msg));

}
```

- 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 operation mode is color recognition or face recognition, the state detection of the virtual button is called. If new data is received, the `QRcommand()` method is called to identify the QR code command to control the movement of the robot.

```
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;
     //识别到二维码 Recognize the QR code
      QRcommand();
  }

}
```

Experimental results

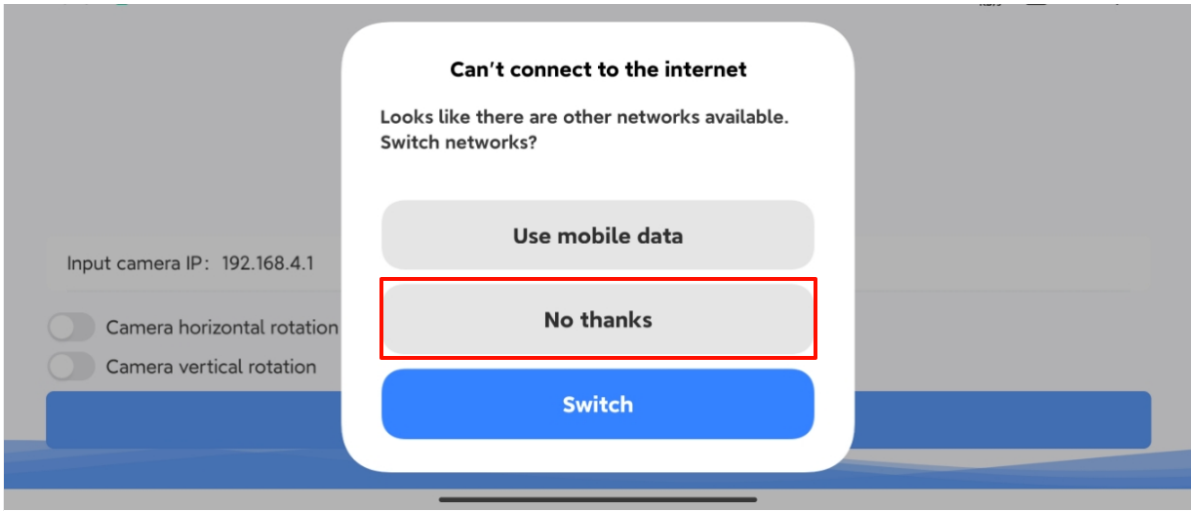
After compiling the program successfully, upload the code to the Arduino Uno development board, disconnect the car from the computer, and connect the WiFi camera to the serial port on the expansion board.

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

APP connection

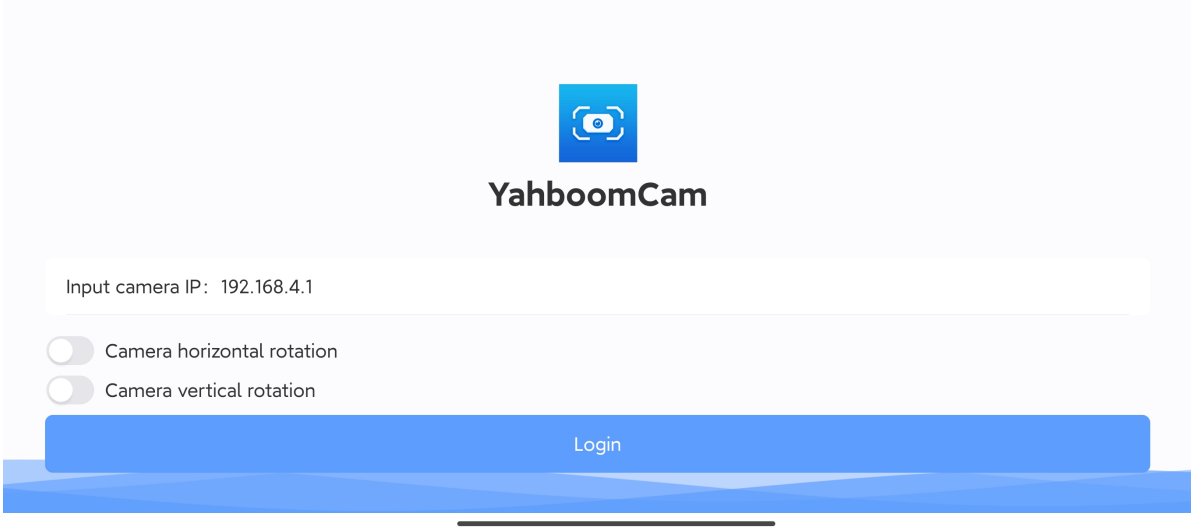
Connect the mobile phone to the hotspot of the WiFi camera (the name of the built-in hotspot is: Yahboom_ESP32_WIFI), and then open the YahboomCam software.

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



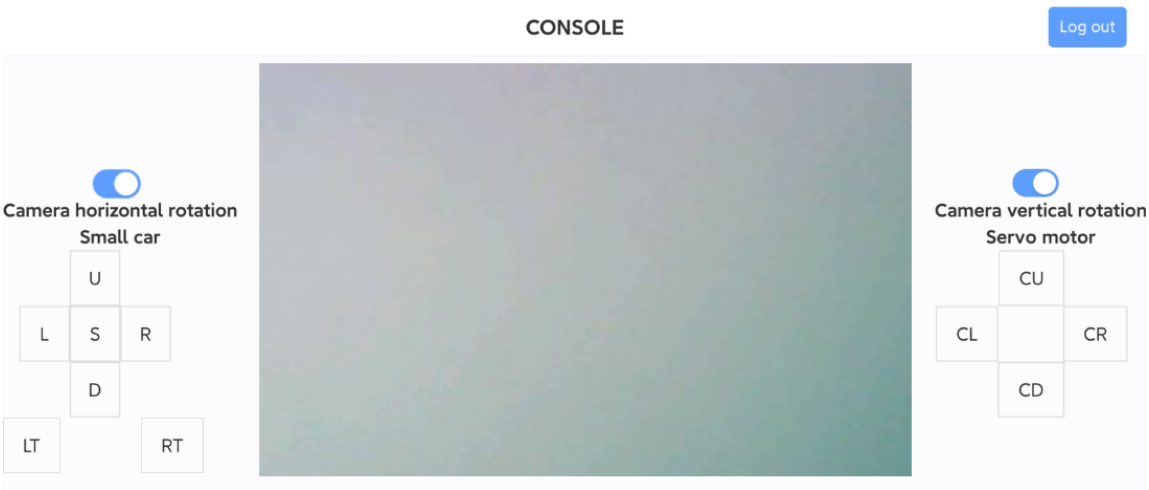
Enter IP:192.168.4.1 in the YahboomCam software, then click login to enter the APP control interface.

```
The IP address 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 picture displayed, check whether the phone is connected to the
WiFi camera hotspot normally.
```



Effect

Aim the camera at the QR code to be recognized. After recognition, the car will perform corresponding actions according to the command, and the command icon will be displayed synchronously on the microbit mainboard.

When the forward command is recognized, the car will move forward;

When the backward command is recognized, the car will move backward;

When the left command is recognized, the car will move left;

When the right command is recognized, the car will move right.

QR code:

go forward

go back

turn left

turn right

Notes

1. The screen needs to be turned on to start recognition. If the screen is turned off, the recognition will also be turned off.
2. Make sure the direction of the camera screen is correct. You need to turn on the vertical rotation and horizontal rotation switches on the APP.