WiFi Control Car

WiFi Control Car
- Download Yahboom CAM
- Device connection
  - Hardware connection
  - Software connection
- Implementation ideas
    - Data format (communication protocol)
- Code Analysis
- Experimental results
- APP remote control steps
  - APP connection
  - APP control

We can use the APP to view the WiFi camera screen, control the car's motion status, and control the servo rotation.

```
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!
```

Download Yahboom CAM

You can scan the QR code below to download the latest APP.

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

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

Data format (communication protocol)

We only parse data in two data formats.

Data format	Description	Example
$	Data header	
#	Data tail	
Data,Data,Data	Data type 1	Data represents only numbers: 1,0,0,0
Alphabet+Data	Data type 2	Alphabet+Data represents a letter plus the following three digits: A100

Example:

Data format 1: $1,0,0,0#, $2,0,0,0#, $3,0,0,0#

Data format 2: $A100#, $A090#, $B120#

Note:

- Data format 1 is mainly used to control the car status
- Data format 2 is mainly used to control the servo angle

Click the function button on Yahboom CAM, the serial port will send data in the above format, our program parses the data and controls the car status.

Code Analysis

Here we only briefly introduce the code content. For detailed code, please refer to the corresponding code file, which is provided in the download area!

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

```
#include <Wire.h>                     // 包含Wire(I2C)通讯库 Include Wire 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 data such as arrays, flag bits, etc

```
// 定义数据，如数组，标志位等 Define data such as arrays, flag bits, etc
#define BUFFER_SIZE 20
volatile uint8_t Rx_Index = 0;
volatile uint8_t Data_Flag = 0;
```

```
char* Data_Array[BUFFER_SIZE];
char Data_Servo[BUFFER_SIZE];

int DataArraySize = 0;

// 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

// 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

// 定义舵机相关参数 Define servo related parameters
#define SERVO_MIN_PULSE_WIDTH 500
#define SERVO_MAX_PULSE_WIDTH 2500
#define SERVO_PULSE_WIDTH_CYCLE 20000
#define SERVO_MIN_ANGLE 0
#define SERVO_MAX_ANGLE 180

// 定义舵机控制引脚 Define the servos control pin
#define Servo1_PIN 0
#define Servo2_PIN 1
#define Servo3_PIN 2
#define Servo4_PIN 3

unsigned int uMotorSpeed = 75;
unsigned int uCurCarState = 0;
unsigned int uPreCarState = 0;
unsigned int uCurRgbState = 0;
unsigned int uPreRgbState = 0;
unsigned int uCurBuzState = 0;
unsigned int uPreBuzState = 0;
unsigned int uCurServoAngle = 90;
unsigned int uPreServoAngle = 0;
```

- 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,
};
```

- Enumerate common colors

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

- Enumerate buzzer state

```
// 枚举蜂鸣器状态 Enumerate buzzer state
enum BuzzerState {
  BuzzerOff,
  BuzzerOn,
};
```

- 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;
    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;
  }
}
```

- 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 BLACK:
      RGB.setPixelColor(0, RGB.Color(0, 0, 0));
      RGB.show();
      break;
    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 State: 发声开关 Whether to make a sound
 * @retval 无 None
 */
void setBuzzerSound(BuzzerState State) {
  switch (State) {
    case BuzzerOn:
      tone(BUZZER_PIN, 1000);
      delay(5);
      break;
    default:
      noTone(BUZZER_PIN);
      break;
  }
}
```

- Set the servo rotation angle

```
/**
 * @brief 设置舵机旋转角度 Set the servo rotation angle
 * @param servoPin: 舵机控制引脚 Servo control pin
 * @param angle: 舵机旋转角度 Servo rotation angle
 * @retval 无 None
 */
void setServoAngle(uint8_t servoPin, int angle) {
  angle = 180 - angle;
  if (angle > 145) {
    angle = 145;
  }
  if (angle < 35) {
    angle = 35;
  }
  long pulseWidth = map(angle, SERVO_MIN_ANGLE, SERVO_MAX_ANGLE, SERVO_MIN_PULSE_WIDTH,
SERVO_MAX_PULSE_WIDTH);
  long PWM_Value = pulseWidth * 4096 / SERVO_PULSE_WIDTH_CYCLE;
  pwm.setPWM(servoPin, 0, PWM_Value);
}
```

- Receive data

Note: Receive data between `$` and `#`

```
/**
 * @brief 接收指定格式的数据 Receives data in the specified format
 * @param 无 None
 * @retval 接收指定数据 Received specified data
 */
char* recData() {
  static char Data_Buffer[BUFFER_SIZE];
  static char Rx_Buffer[BUFFER_SIZE];

  if (Serial.available() > 0) {
    char SerialData = Serial.read();
    if (SerialData == '$') {
      memset(Rx_Buffer, 0, sizeof(Rx_Buffer));
      memset(Data_Buffer, 0, sizeof(Data_Buffer));
      Data_Flag = 1;
      Rx_Index = 0;
    } else if (Data_Flag == 1 && SerialData == '#') {
      if (Rx_Index < BUFFER_SIZE) {
        memcpy(Data_Buffer, Rx_Buffer, Rx_Index);
        memset(Rx_Buffer, 0, sizeof(Rx_Buffer));
        Rx_Index = 0;
        Data_Flag = 0;
        // Serial.println(Data_Buffer);
        return Data_Buffer;
      } else {
        // Serial.println("Rx_Buffer is full!");
        memset(Rx_Buffer, 0, sizeof(Rx_Buffer));
        Rx_Index = 0;
        Data_Flag = 0;
      }
    } else {
      if (Rx_Index < BUFFER_SIZE) {
        Rx_Buffer[Rx_Index++] = SerialData;
      } else {
        // Serial.println("Rx_Buffer is full!");
        memset(Rx_Buffer, 0, sizeof(Rx_Buffer));
        Rx_Index = 0;
        Data_Flag = 0;
      }
    }
  }
  return nullptr;
}
```

- Separate the data into a new array

```
/**
 * @brief 将数据分离到一个新的数组中 Separate the data into a new array
 * @param dataBuffer: 要分离的数据 The data to be separated
 * @param dataArray: 分离后的数据 The data after separation
 * @param dataArraySize: 分离数据的大小 The size of the separated data
 * @retval 无 None
 */
void splitDataAndSave(char* dataBuffer, char** dataArray, int& dataArraySize) {
  char* token = strtok(dataBuffer, ",");
  int index = 0;
  while (token != NULL) {
    dataArray[index] = token;
    token = strtok(NULL, ",");
    index++;
  }
  dataArraySize = index;
}
```

- Parse the serial port data after processing

```
Dump format 2 data and print format 1/2 data
```

```
/**
 * @brief 处理后解析串口数据 Parse the serial port data after processing
 * @param dataBuffer: 分离后的数据 The data after separation
 * @retval 无 None
 */
void parseSerialData(char* dataBuffer) {
  if (dataBuffer != nullptr) {
    // Serial.println(dataBuffer);
    if (isalpha(dataBuffer[0])) {
      if (dataBuffer[0] == 'B') {
        memcpy(Data_Servo, dataBuffer + 1, 3);
        uCurServoAngle = strtol(Data_Servo, NULL, 10);
      }
      // memcpy(Data_Servo, dataBuffer + 1, 3);
      // uCurServoAngle = strtol(Data_Servo, NULL, 10);
      // Serial.print("Servo angle:");
      // Serial.println(strtol(Data_Servo, NULL, 10));
    } else {
      splitDataAndSave(dataBuffer, Data_Array, DataArraySize);
      // Serial.print("Car state:");
      // Serial.println(strtol(Data_Array[0], NULL, 10));
      uCurCarState = strtol(Data_Array[0], NULL, 10);  // Control the car status
      // Serial.print("RGB state:");
      // Serial.println(strtol(Data_Array[1], NULL, 10));
      uCurRgbState = strtol(Data_Array[1], NULL, 10);  // Control the RGB status
      // Serial.print("Buzzer state:");
      // Serial.println(strtol(Data_Array[2], NULL, 10));
      uCurBuzState = strtol(Data_Array[2], NULL, 10);  // Control the Buzzer status
      // Serial.print("No define:");
      // Serial.println(strtol(Data_Array[3], NULL, 10));
    }
    memset(Data_Servo, 0, sizeof(Data_Servo));
    memset(Data_Array, 0, sizeof(Data_Array));
  }
}
```

- WiFi control car function

```
/**
 * @brief WiFi遥控小车 WiFi control car function
 * @param 无 None
 * @retval 无 None
 */
void WiFi_Control() {
  parseSerialData(recData());  // Parse the serial port data after processing
  // Print the car status and control the car
  if (uCurCarState != uPreCarState) {
    setCarMove(uCurCarState, uMotorSpeed);
    uPreCarState = uCurCarState;
  }
  // Print the RGB status and control the RGB
  if (uCurRgbState != uPreRgbState) {
    setRGBColor(uCurRgbState);
    uPreRgbState = uCurRgbState;
  }
  // Print the buzzer status and control the buzzer
  if (uCurBuzState != uPreBuzState) {
    setBuzzerSound(uCurBuzState);
    uPreBuzState = uCurBuzState;
  }
  // Print and control the Servo roate angle
  if (uCurServoAngle != uPreServoAngle) {
    setServoAngle(Servo1_PIN, uCurServoAngle);
    uPreServoAngle = uCurServoAngle;
  }
}
```

- Initialization Code

```
void setup() {
  Serial.begin(115200);                                     // 初始化I2C通讯 Initialize I2C 
communication
  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
}
```

- Looping code

```
void loop() {
  // WiFi遥控小车 WiFi remote control car
  WiFi_Control();
}
```

Experimental results

After successfully compiling the program and uploading 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 control it through the Yahboom CAM APP.

```
The drive motor needs an external battery pack and the expansion board switch is turned on to
drive it normally.

The burning program cannot use other programs to occupy the serial port or an external serial
communication module (for example: WiFi camera module), otherwise the program cannot be
burned or an error message will be prompted!
```

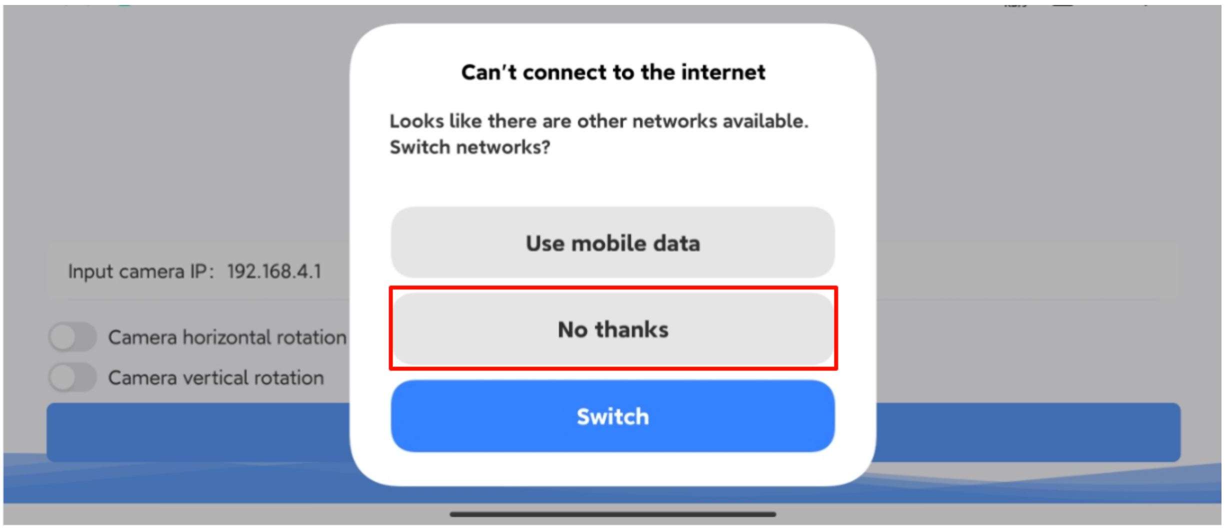
APP remote control steps

It is recommended to use the hotspot built into the WiFi camera module for control.

APP connection

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

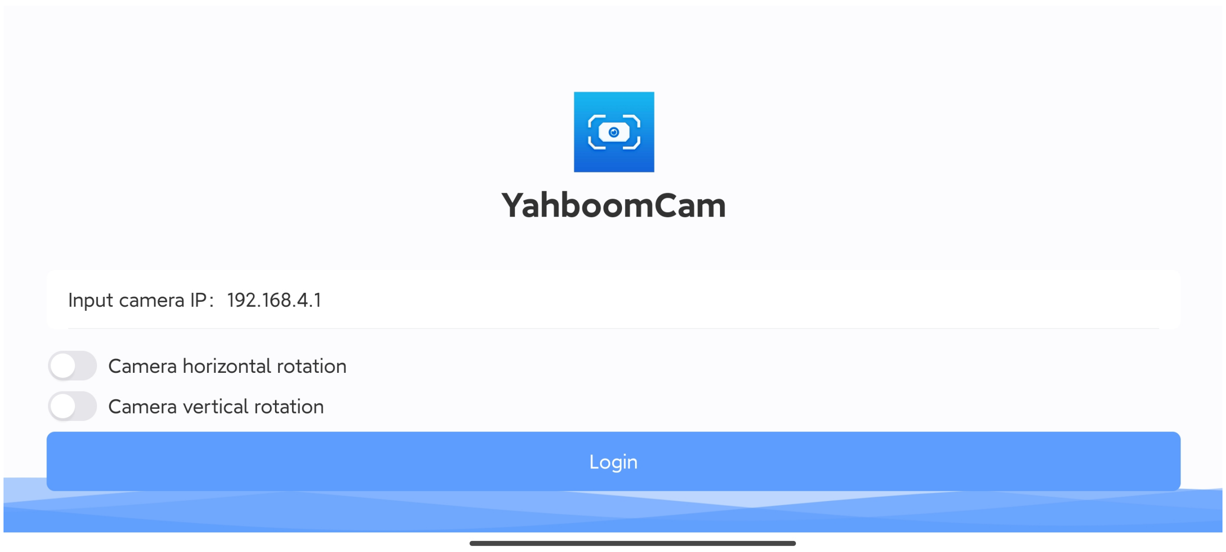
```
Some mobile phones will prompt for connecting to a hotspot without a network. We need to
click to keep connected!
```


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 to log in and enter the APP control interface.

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



Input camera IP： 192.168.4.1

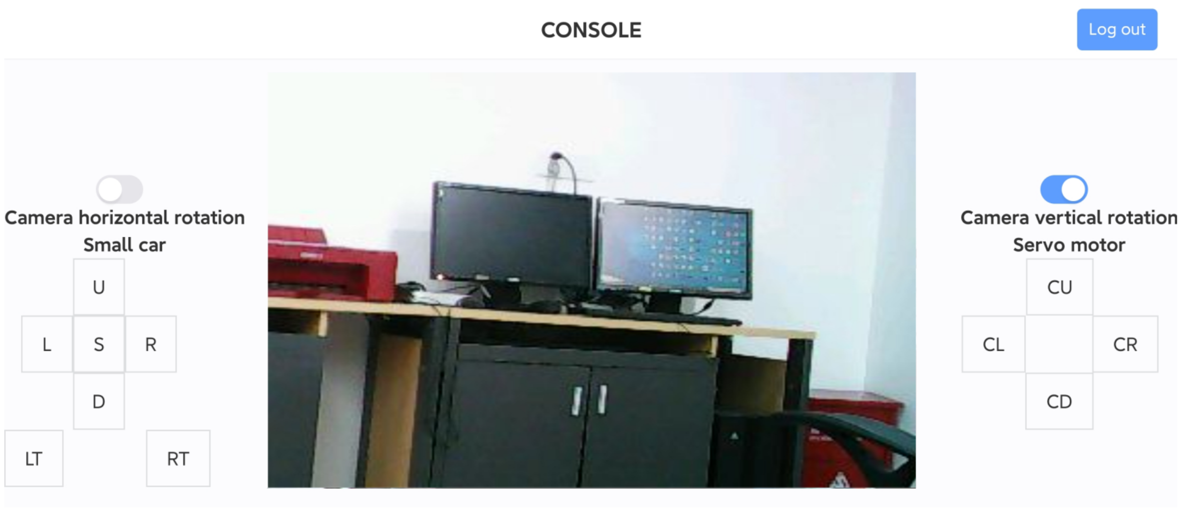
Camera horizontal rotation

Camera vertical rotation

APP control

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
```



- Left button

Control the movement of the car: U (forward), D (backward), L (left move), R (right move), S (stop), LT(left rotation), RT(right rotation)

- Right button

Control the rotation angle of the servo: CL(rotate left), CR (rotate right), CU (rotate upward), CD (rotate downward)

Note:

Image problem: Due to the installation problem of the WiFi camera on our car, we need to check the horizontal flip and vertical rotation of the camera, so that the displayed image will be normal!

Servo control problem: Our car is only equipped with one servo, so it can only be controlled by the left and right of the servo, and the rotation range is controlled in front of the car [35°, 145°].

```
The car program limits the rotation angle of the servo sent by the APP, and the control
rotation range is [35°, 145°], which is to avoid collision or squeezing between the WiFi
camera and the car expansion board.
```