5.6 WIFI camera control car

1. Learning goal:

Learn about the serial communication method, analyze the communication protocol, use the APP to control the movement of the car, and the state machine switches the movement state of the car.

2. Experimental phenomena:

After we open the power of switch, the mobile phone connects to the WIFI signal of the car, then opens the APP to control the car forward, backward, left translation, right translation, spin left and spin right.

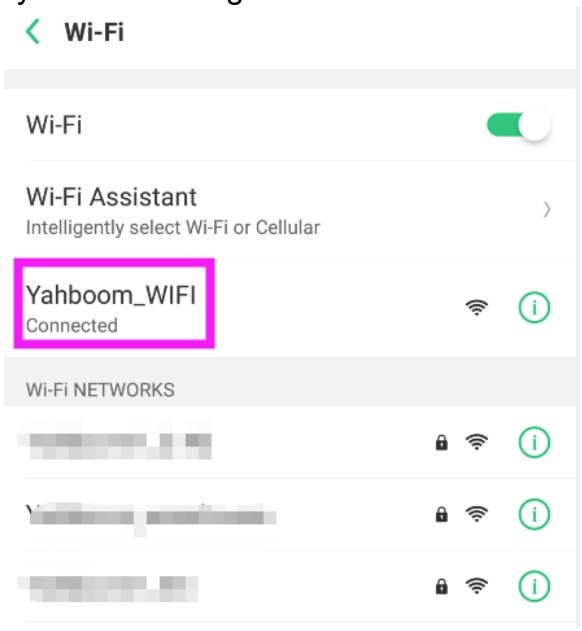
3. Analysis:

3.1 We can send data to WIFI camera by mobile APP.

3.1.1 Please use your mobile phone scan QR code on the cover of the Instruction manual, download and install the YahboomRobot APP.

Note:During installation, If you find any prompts on your phone (for example: location permissions of your phone). You must select "Allow".

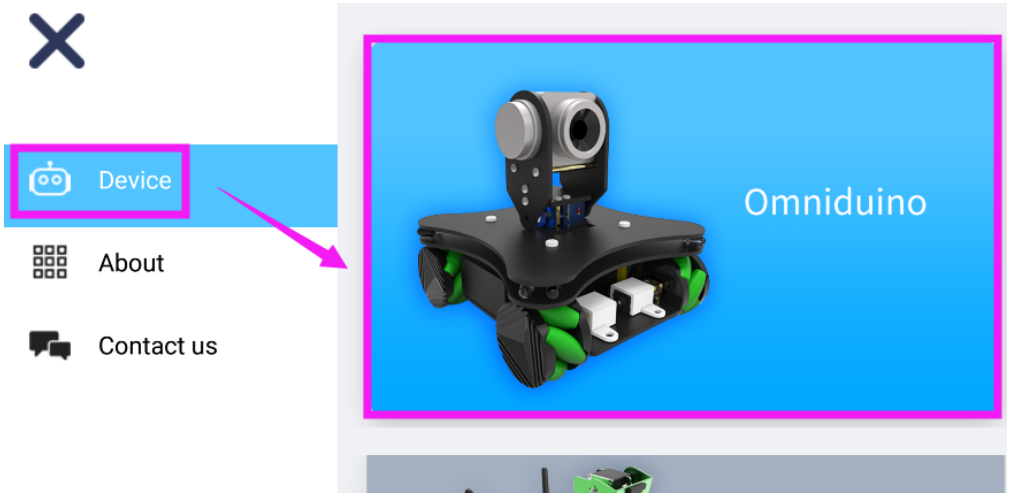
3.1.2 Turn on the phone to search for the WIFI signal and connect the Yahboom_WIFI, no need password, just click on the connection. Some mobile phones may promptly disconnect without network data, and you can click again to connect.



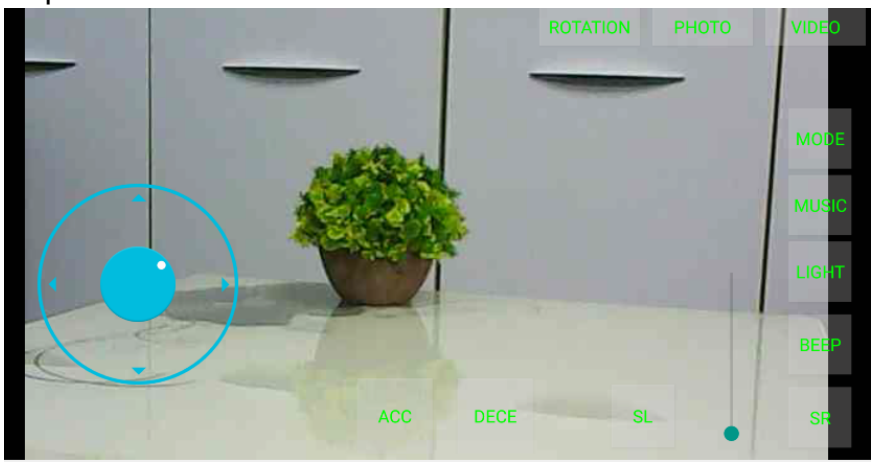
3.1.3 Open the YahboomRobot APP.

!Note: When you open it for the first time, it will be prompted to get some permissions, you must select “Allow”.

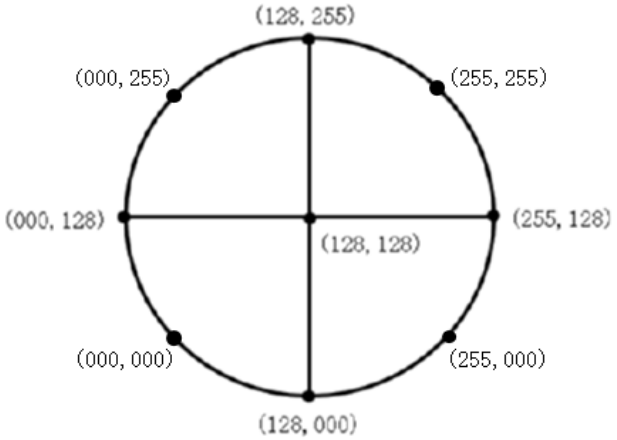
3.1.4 Click on the menu icon in the upper left corner of the APP  . In the device column, you need to select Omniduino. As shown below.



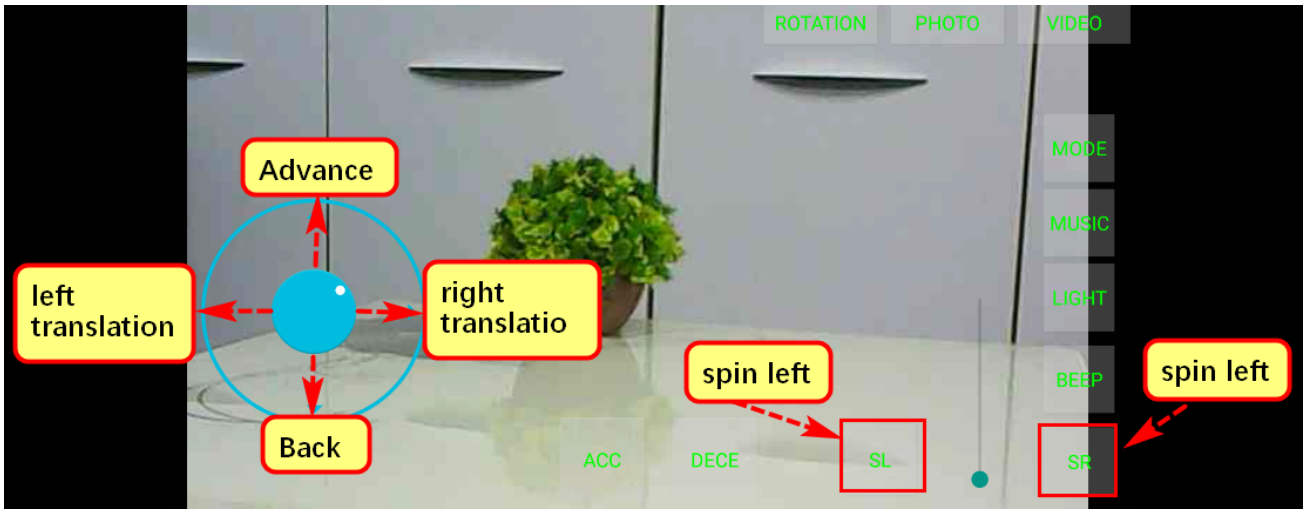
3.1.5 Click on the picture to enter the remote control interface of the omniduino car.



3.2 When the APP simulation joystick is sent, the X axis gradually increases from left to right (0~255), the Y axis gradually increases from down to up (0~255), and the middle position is (128, 128).

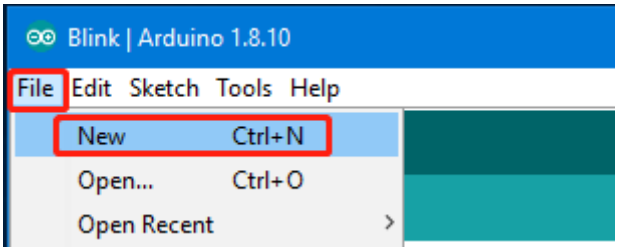


3.3 **“SL”** button in APP, press send: $Spin, 11#, release send: $Spin, 00#.
3.4 **“SR”** click button in APP, press send: $Spin, 21#, release send: $Spin, 00#.
As shown below:

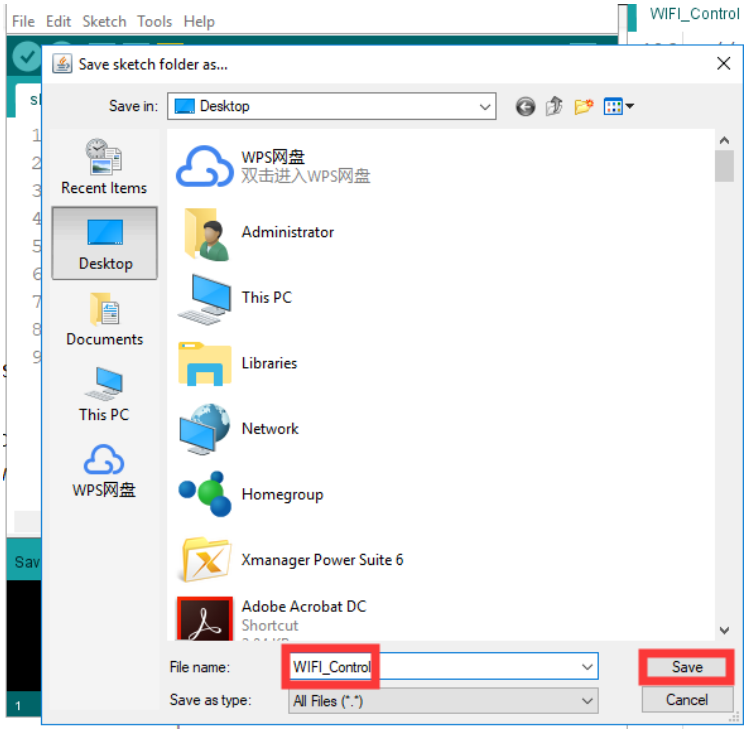


4. Create new project

4.1 Click【File】-->【New】.



4.2 Press **Ctrl+S** to save and rename WIFI_Control. As shown below.



4.3 We can see that there is a Serial folder with **WIFI_Control.ino** on the computer desktop.

4.4 We will **WIFI_Control1.ino** as shown below.

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

The setup() function only runs once when the car is turned on or when the reset button is pressed, and the program for initializing the relevant content can be written;

The loop() function is the main loop function of the car and most of the data processing and logic processing are done in this function.

5. **Programming**

5.1 Add a global variable that controls the car.

```
//Car control parameters
int CarSpeedControl = 60;
int g_CarState  = 0;

/*Car running status enumeration */
const typedef enum {
  enSTOP = 0,
  enRUN,
  enBACK,
  enLEFT,
  enRIGHT,
  enSPINLEFT,
  enSPINRIGHT
} enCarState;
```

5.2 Add serial port data setting variables.

```
/*serial data setting*/
int IncomingByte = 0;            //Received data byte
int Receive_Length = 0;          //Receive data length
String InputString = "";         //Used to store received content
boolean NewLineReceived = false; //Previous data end mark
boolean StartBit = false;        //Agreement start sign
```

5.3 New create serialEvent() function,which to receive the information transmitted by the serial port. After filtering, save the contents of the omniduino omnidirectional car communication protocol to the InputString variable, and finally set NewLineReceived = true.

```
void serialEvent()
{
  while (Serial.available())
  {
    //One byte is read one byte, and the next sentence is read into the string array to form a completed packet.
    IncomingByte = Serial.read();
    if (IncomingByte == '$')        //$Indicates that the agreement begins
    {
      StartBit = true;
    }
    if (StartBit == true)          //Start receiving content
    {
      InputString += (char)IncomingByte;
      Receive_Length++;
      if (Receive_Length >= 15)
      {
        StartBit = false;
        Receive_Length = 0;
        InputString = "";
      }
    }
    // #indicates the end of the agreement
    if (StartBit == true && IncomingByte == '#')
    {
      StartBit = false;
      Receive_Length = 0;
      int len = InputString.length();
      if (len >= 8 && len <= 13)
      {
        NewLineReceived = true;
      }
      else
      {
        NewLineReceived = false;
        InputString = "";
      }
    }
  }
}
```

5.4 New create serial_data_parse() function is mainly divided into three parts. The first part is based on the data sent by the APP simulation rocker, and the motion state of the car is modified after parsing and processing.

```
if (InputString.indexOf("X") >= 0 && InputString.indexOf("Y") >= 0)
{
  //Extract the data sent by the APP simulation joystick(eg:stop):$X128,Y128#)
  int X = InputString.substring(2, 5).toInt();
  int Y = InputString.substring(7, 10).toInt();
  if (X == 128 && Y == 128)  // stop
  {
    g_CarState = enSTOP;
  }
  else if (X <= 5)           //Left translation
  {
    g_CarState = enLEFT;
  }
  else if (X >= 255)         //Right translation
  {
    g_CarState = enRIGHT;
  }
  else if (Y <= 5)           //Back
  {
    g_CarState = enBACK;
  }
  else if (Y >= 250)         //Adavance
  {
    g_CarState = enRUN;
  }
}
```

The second part: deal the data sent by the spin button

```
if (InputString.indexOf("Spin") >= 0)
{
  if (InputString[6] == '0' && InputString[7] == '0')
  {
    g_CarState = enSTOP;
  }
  else if (InputString[7] == '1')
  {
    if (InputString[6] == '1')      //Spin left
    {
      g_CarState = enSPINLEFT;
    }
    else if (InputString[6] == '2') //Spin right
    {
      g_CarState = enSPINRIGHT;
    }
  }
}
```

The third part:clear serial port data and NewLineReceived = false.

```
InputString = ""; //clear serial port data
NewLineReceived = false;
```

5.5 In the setup () function to increase the initialization of the serial port, set the baud rate to 9600. To use serial communication, this step initialization is necessary, otherwise the serial port data cannot be transmitted.

```
Serial.begin(9600);
```

5.6 The loop() main function parses the serial protocol, and the state machine modifies the motion state of the car.

```
void loop() {
  // put your main code here, to run repeatedly:
  serialEvent();        //Receive serial port information and save protocol content
  if (NewLineReceived)
  {
    serial_data_parse(); //Call the serial port analytic function
  }

  //According to the state of the car, do the corresponding action
  switch (g_CarState)
  {
  case enSTOP:
    brake();
    break;
  case enRUN:     //Advance
    run(CarSpeedControl);
    break;
  case enLEFT:    //Left translation
    left(CarSpeedControl);
    break;
  case enRIGHT:   //Right translation
    right(CarSpeedControl);
    break;
  case enBACK:    //back
    back(CarSpeedControl);
    break;
  case enSPINLEFT://spin left
    spin_left(CarSpeedControl);
    break;
  case enSPINRIGHT://spin right
    spin_right(CarSpeedControl);
    break;

  default:
    brake();
    break;
  }
}
```

6. Compiling and downloading code

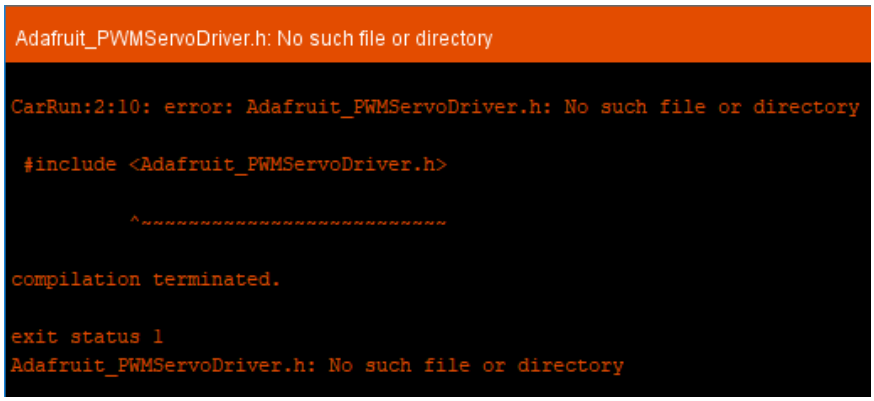
6.1 After the code is written, press Ctrl+S to save, then click the "√" button to compile. If there is no problem, click "→" to upload (the car must be connected to the computer via the USB cable).

File Edit Sketch Tools Help

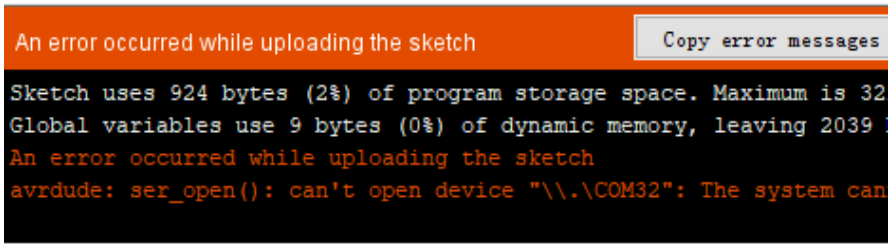
LED

```
*/
//Define LED light(D9)pin
#define LED_PIN 5

void setup() {
  // put your setup code here, to run once:
  // set LED pin to output mode
  pinMode(LED_PIN, OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(LED_PIN, LOW);       //LED is on
  delay(500);
  digitalWrite(LED_PIN, HIGH);      //LED is off
  delay(500);
}
```

6.2 If there is an error like the following, it means that the library file is missing. Please copy the library file provided by the omniduino omnibus to the library file directory compiled by arduinoIDE.
please refer to 【3.Development Environment Construction】----【3.4 Add additional library files】

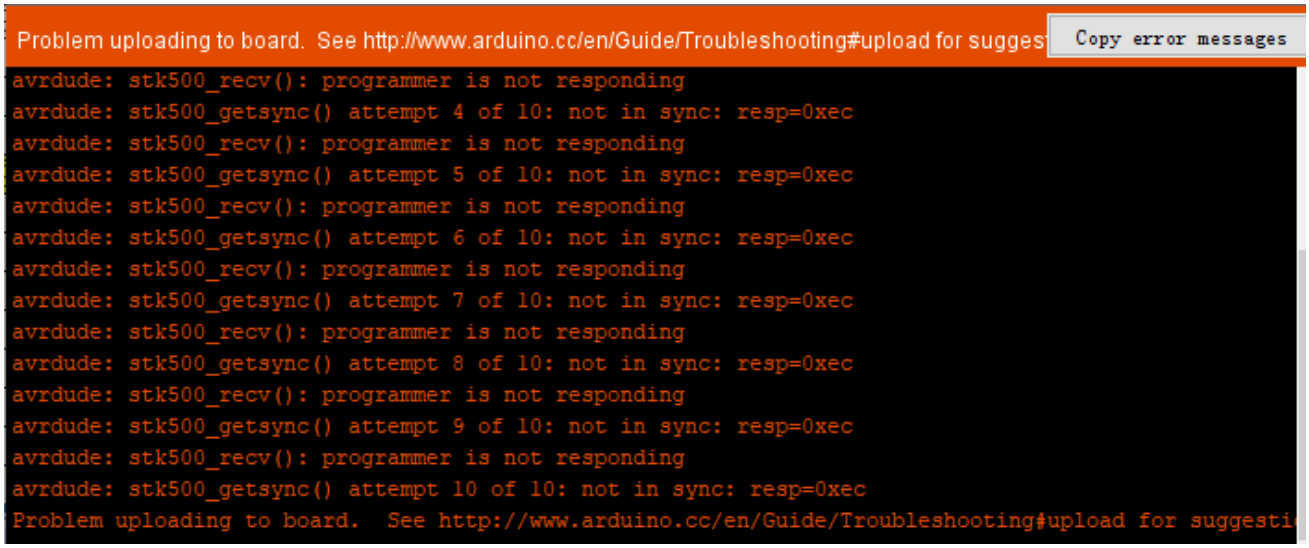


6.3 If the compilation passes normally, but the following error occurs during uploading, the reason may be that the wrong serial port or the serial port is occupied.



Solution: Open the device manager to see if there is a serial port with CH340 tag. If not, please restart the Omniduino car, then, re-plug the USB cable or replace a USB cable;
If there is a serial port number, we need to close the other serial port or assistant software,

avoid serial port occupation, and then re-select the serial port to ArduinoIDE【Tool】-->【Port】.
6.4 After clicking the upload button, the upload is always displayed, but it can't be uploaded successfully for a long time.



Because the uploading program and the WIFI camera communication is realized through the serial port, when the serial port is occupied by the WIFI camera, and the program cannot be uploaded.

Solution:

①Unplug the USB cable, turn off the power of the car, wait for the D2 indicator to go out.

②Then, plug in the USB data cable. At this time, your mobile phone should not connect the WiFi signal of the car.

③You can upload the program to the car according to the normal steps.

④After the program is successfully uploaded, unplug the USB data cable, open the power switch of the car. The corresponding experimental phenomenon will appear.

(Tip: If you upload APP control program. After the program is successfully uploaded, unplug the USB data cable, open the power switch of the car. Mobile phone connect the car to the WIFI signal, and then open the APP to control.)