**Lesson 16 APP Controlled Robot**

**Table**

[1. ESP8266-01 Module 1](#_Toc32497)

[2. Install and learn the Smart Robot Car APP 6](#_Toc23448)

[3. Upload the code 11](#_Toc922)

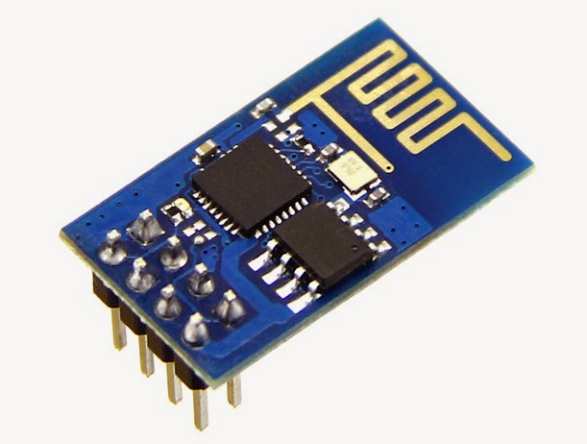
[4. Troubleshooting 13](#_Toc14690)

[5. Code 14](#_Toc6033)

[6. Any questions and suggestions are welcome 31](#_Toc9118)

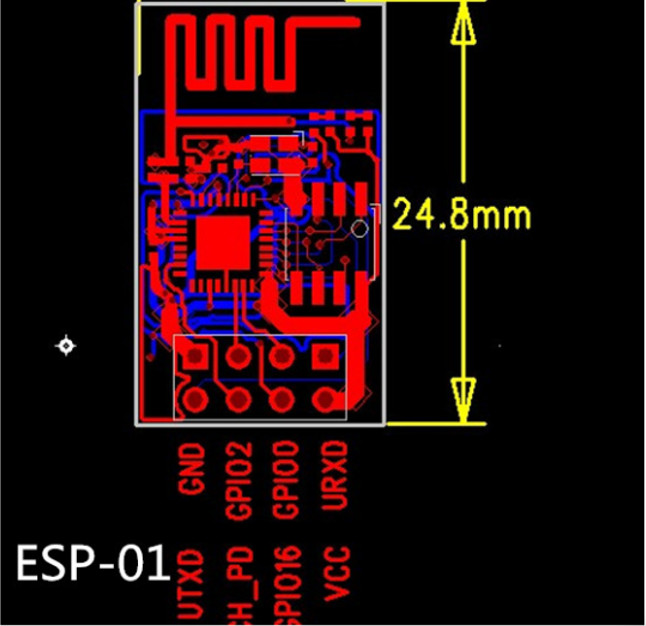
# 1. ESP8266-01 Module

The ESP8266 ESP-01 is a Wi-Fi module that allows microcontrollers access to a Wi-Fi network.



ESP8266-01 MODULE(hereinafter referred to as ESP-01)

ESP-01 pins



| **PIN** | **Description** |
| --- | --- |
| URXD | UART\_RXD，receive |
| UTXD | UART\_TXD，send |
| GPIO 16 | External Reset signal, reset when low level, work when high level (default high) |
| GND | GND |
| VCC | 3.3V |
| GPIO 0 | Working mode selection: floating: FlashBoot, working mode; pull down: UARTDownload, downloading mode |
| CH\_PD | Work at high level; power off at low level |
| GPIO 2 | (1) It must be high level when powering on, and the hardware pull-down is prohibited; (2) Internal default pull-up |

## 1.1 Introduction to AT commands of ESP-01 module

Basic instructions

|  |  |
| --- | --- |
| **command** | **description** |
| AT | Test AT boot |
| AT+GMR | View version information |
| AT+CWMODE | Select WIFI application mode |
| AT+RST | restart module |
| **client mode** | |
| AT+CWLAP | List currently available router access points |
| AT+CWJAP | Join access point |
| AT+CWQAP | exit access point |
| AT+CIPSTART | Establish TCP, connect to server |
| AT+CIPCLOSE | Close TCP |
| AT+CIFSR | Get local IP address |
| AT+CIPMODE | Set module transfer mode |
| AT+CIPSEND | Send data |
| **server mode** | |
| AT+ CWSAP | Query and set the WIFI name, password and encryption method in AP (server) mode |
| AT+ CWLIF | View the IP address of the connected device |
| AT+CIPMUX | Start multiple connections |
| AT+CIPSERVER | Configured as server default port 333 |
| AT+CIPSTO | Set server timeout |
| AT+ CIPSTATUS | Get connection status |

## 1.2 Working mode and commands of ESP-01 module

**ESP-01 module acts as a client (transparent transmission)**

1. AT: Test AT development mode start
2. AT+GMR: View firmware version information
3. AT+CWMODE=1: Set WIFI application mode

（1） Station mode

（2） AP mode

（3） AP and Station mode, AP refers to as an access point, station refers to as a client station

4. AT+RST：restart

5. AT+CWLAP：list available access points

6. AT+CWJAP="wifiname","wifi passport"：join wifi

7. AT+CIFSR ：get local IP address

8. The PC connects to the router, and the network debugging assistant uses the computer IP address to create a server

9. AT+CIPSTART="TCP","192.168.101.110",8080    Establish a TCP connection with the server

10. AT+CIPMODE=1：Set the transparent transmission mode (you can send it all the time, otherwise you have to use AT+CIPSEND=4 to send the number of bytes; as a server mode, you cannot use the transparent transmission mode)

11. AT+CIPSEND ：Start transparent transmission, the serial port debugging assistant sends data, and the network debugging assistant sends data

12. Received data format: serial port debugging: +IPD, n:xxxxxxxxxx The length of the received data is n bytes, xxxxx is the data; network debugging: [Tcp client 192.168.1.108 2872] 123, TCP mode, client IP address, port number, 123 is the data

**ESP-01 WIFI module as client (single connection)**

1. AT+CWMODE=1: set WIFI application mode

(1) Station mode

(2) AP mode

(3) AP and Station mode,

AP refers to as an access point, station refers to as a client station

1. AT+RST：restart
2. AT+CWJAP="wifiname","wifi password": join wifi
3. PC is connected to the router, and the network debugging assistant uses the computer IP address to create a server, and the IP setting is shown in Figure 2 above;
4. AT+CIPSTART="TCP","192.168.101.110",8080    ：Establish a TCP connection with the server
5. AT+CIPSEND=4: Serial port debugging sends four bytes of data, input the content of the four bytes to be sent, no need to press Enter. If the number of bytes sent exceeds the length n set by the command, it will respond busy, and send the first n bytes of data, and respond SEND OK after completion. Network debugging can be sent arbitrarily.

**ESP-01 WIFI module as server**

1.AT+CWMODE=2 ：set WIFI application mode

(1) Station mode

(2) AP mode

(3) AP and Station mode,

AP refers to as an access point, station refers to as a client station

2.AT+RST：restart

3.AT+CWSAP? ：Query and display the parameters in AP mode，+CWSAP:"ESP\_8266","12345678",11,3,4,0

4.AT+CWSAP="ESP\_8266","12345678",11,3 ：access point name, password, channel number, encryption method. 11 is the channel number, it needs to be restarted after modification, and 3 is the encryption method

|  |  |  |  |
| --- | --- | --- | --- |
| <ecn> | Encryption | 0 | OPEN，if set to open, it will not work even if a password is set |
| 1 | WEP |
| 2 | WPA\_PSK |
| 3 | WPA2\_PSK |
| 4 | WPA\_WPA2\_PSK |

1. AT+CIPMUX=1:start multiple connections
2. AT+CIPSERVER=1:create a server, the default port is 333
3. AT+CIPSTO=300：set the server timeout from 0 to 28800, the unit is s, and the client will be kicked out when the timeout expires.
4. AT+CIFSR :obtain the local IP address in order to set up the network assistant. First, the PC needs to be connected to the hotspot of the WIFI module, and the network debugging assistant on the PC connects to the AP as a client.
5. AT+CWLIF:view connected devices

This lesson uses the ESP-01 module as the working mode of the server:

1.AT+RST\r\n //In the Arduino code, the AT command must end with a carriage return and line feed character "\r\n"

2. AT+CWMODE=3\r\n //set to soft AP+station mode

3. AT+CWSAP=\"Cokoino\_ESP8266-01\",\"12345678\",11,0\r\n

//. Cokoino\_ESP8266-01 -----------WIFI access point name

//. 12345678 -----------WIFI password

//. 11 ------------Channel number

//. 0 -------------Encryption mode 0-OPEN

4. AT+CIPMUX=1\r\n //start multiple connections

5. AT+CIPSERVER=1,3001\r\n //Create a server, the default port is 333, modify the port to 3001, consistent with APP

6. AT+CIPSTO=7000\r\n // Example Set the server timeout period to 7000 seconds

# Install and learn the Smart Robot Car APP

## 2.1 Install APP

The APK file of the Robot APP is stored in this folder：E:\CKK0002-master\Robot apk\app-release.apk

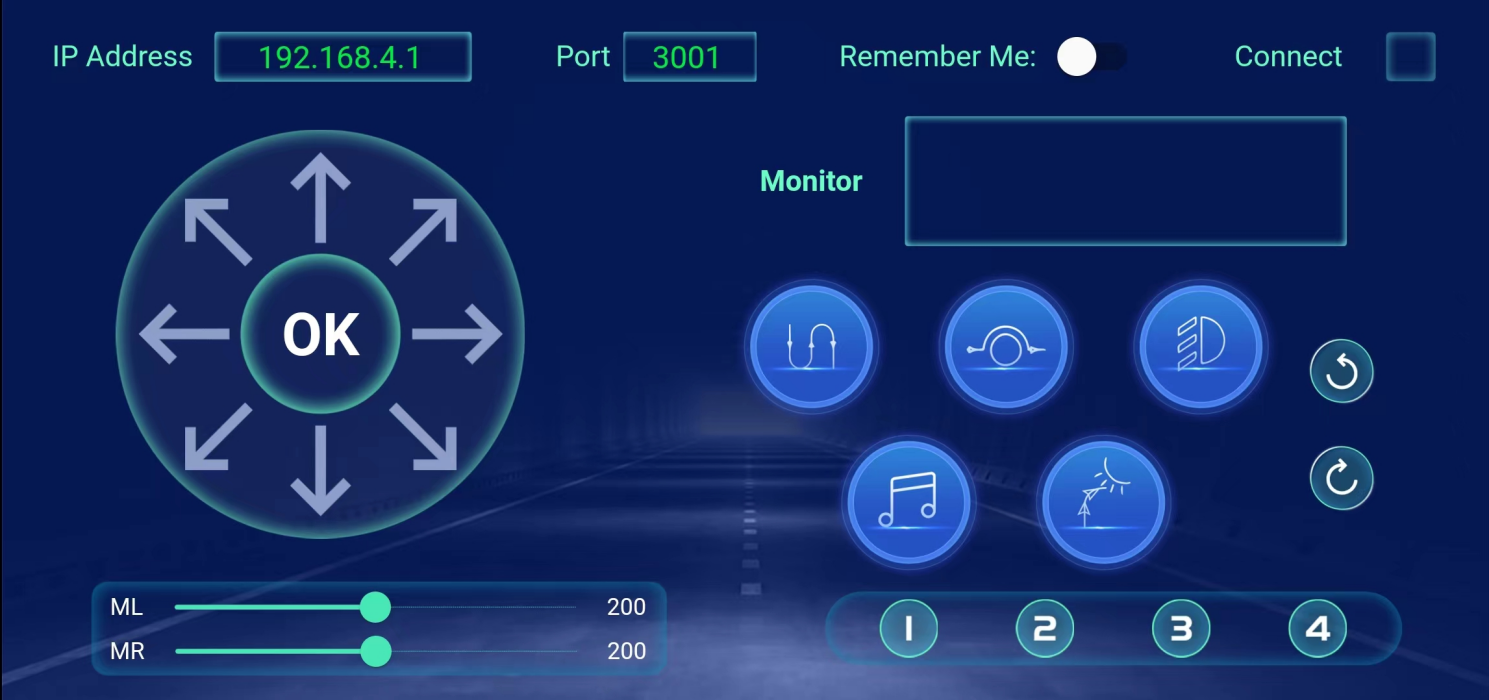
Send the .apk file to the mobile phone for installation. Note: this APP is only compatible with Android phones

If a risk warning pops up during installation, please ignore the risk and choose to continue the installation. We guarantee that the APP is virus-free and risk-free. After the installation is complete, we will see the Robot APP icon as shown below on the phone



## 2.2 Introduction of Smart Robot Car APP

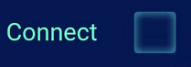
Click the Robot APP icon on the mobile phone, the APP interface is as follows



**Introduction to APP UI**

 It is the address of the ESP-01 module as a server. It is a fixed value and cannot be changed on the APP interface.

 It is the port number of the ESP-01 module as a server, it is set to be a fixed value of 3001 in the app, so when you writing the Arduino code, be sure to use the AT command to set the port number of the ESP-01 module to 3001.

 Connect Button. Click "Connect", you can connect the wifi to the ESP-01 module, when the connection is completed, the Monitor will display a successful connection message

This is the data monitoring window, which can simultaneously display the operation instructions and status of the APP

 Function button: “following the line”. When the APP is connected to the ESP-01 module of the car, press this button, and the car will follow the line. control command:“trk”.

 Function button:“Avoid obstacles”. When the APP is connected to the ESP-01 module of the car, press this button, and the car will drive automatically and avoid obstacles. Control command“aod”.

 Function button:“Light Show”，When the APP is connected to the ESP-01 module of the car, press this button, and the led light on the car will be turned on and change various colors. control command:“lgt”.

 Function button:“Music”. When the APP is connected to the ESP-01 module of the car, press this button, the buzzer on the car will start playing music with different melodies. control command:“muc”.

 Function button:“Follow Light”. If you add a photosensitive sensor to the car, press this button, the robot will follow the light source in a dark environment. control command:“flt”. Note that our robot car is not equipped with a photosensitive sensor, and this function will not be used in this lesson.

Function button:“Rotaiton Left”，when the APP is connected to the ESP-01 module of the car, press this button, the car will turn left in a circle. control command:“rtl”.

Function button:“Rotaiton Right”，when the APP is connected to the ESP-01 module of the car, press this button, the car will turn right in a circle. control command:“rtr”.

 Function button:“Button 1”，it is defined as the rotation of the servo. After the APP is connected to the ESP-01 module on the car, press this button, and the servo on the car will start to work. control command:“bt1”.

Function button:“Button 2”、 “Button 3”、 “Button 4”， control command: “bt2”、 “bt3” 、 “bt4”.Functions are undefined because they are not used in this lesson.

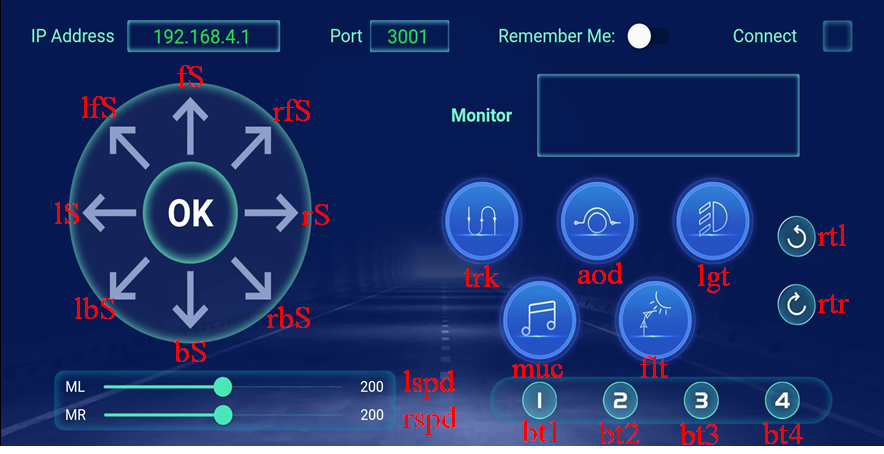
Drag-and-drop button:“Left Speed”、 “Right Speed”，when the APP is connected to the ESP-01 module of the car, dragging these two buttons will change the rotation speed of the left and right wheels of the car. control commands:“lspd”、 “rspd”

 Move direction buttons,the arrow is the control button for the driving direction of the car, and there are 8 control directions in total, namely "forward", "left forward", "left", "left backward", "backward", "right backward", "right", "right forward" ", the middle "OK" button is defined as the stop button.When the APP is connected to the ESP-01 module of the car,You can control the direction of movement of the car with these arrow buttons.

## 2.3 Introduction to button commands on the APP interface

All function buttons on the APP interface have a fixed command, which is unique and invariable. Therefore, when writing Arduino codes, you need to pay attention to matching the judgment commands in the code with the commands of the APP function button, otherwise the APP will not control the car correctly.

The commands of the function button on the APP interface are as follows:

After you successfully upload the code and connect the APP to the ESP-01 module of the car, press the button in the app, and the ESP-01module will receive the command sent by the APP button and convert it into a string signal. String signals are as follows:

/// Car driving direction control button on the app interface, a total of 8 direction buttons.

const String phone1 = "fS";  // forwardStart:

const String phone1\_5 = "lfS";  // forward\_left\_Start

const String phone2 = "lS";    // leftStart

const String phone2\_5 = "lbS";  // left\_backward\_Start

const String phone3 = "bS";    // backwardStart

const String phone3\_5 = "rbS";  // backward\_right\_Start

const String phone4 = "rS";    // rightStart

const String phone4\_5 = "rfS";  // right\_forward\_Start

/// The other function buttons on the app interface

const String phone5 = "OK";//stop

const String phone6 = "rtl";//rotation left

const String phone7 = "rtr";//rotation right

const String phone8 = "trk";//track line running

const String phone9 = "aod";//Avoid obstacles

const String phone10 = "lgt";//light show

const String phone11 = "muc";//buzzer

const String phone12 = "flt";//fllow light

const String phone13 = "bt1";//button1

# 3. Upload the code

3.1 The code used in this lesson is placed in this folder：

E:\CKK0002-master\Tutorial\sketches\14\_1\_Wifi\_Controlled\_Car

3.2 Install Regexp library

For the installation method, please refer to the method of installing the library Servo.h in Lesson 4

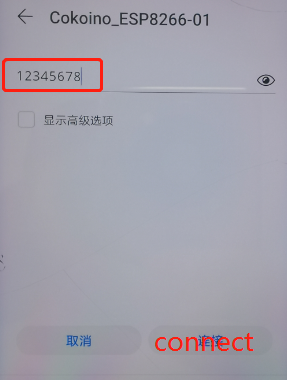
3.3 Before uploading the code, turn the ESP-01 switch on the control board to the side away from the "ESP-01" silk screen.

3.4 After uploading the code, unplug the USB cable, put the Smart Robot Car on the ground, turn on the power switch on the control board.turn the ESP-01 switch on the control board to the "ESP-01" silk screen.The LCD on the car will display"WIFI Prepared!"

3.5 Click "Settings" on the mobile phone, and click "WLAN" on the setting interface to enter the WLAN interface. Then look for the "Cokoino\_ESP8266-01" signal in the list of available WLANs

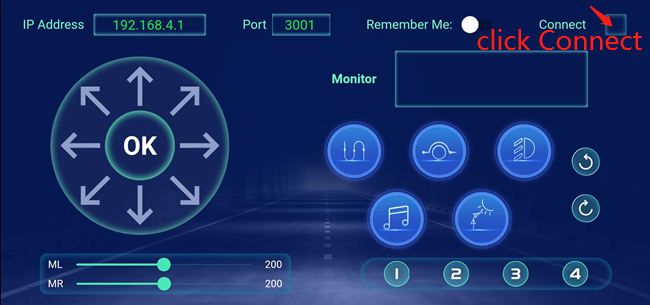


3.6 Click “Cokoino\_ESP8266-01”WLAN，enter password 12345678，then click "connect"



After the connection is successful, if the mobile phone pops up a window prompting that the current WLAN cannot access the Internet, whether to continue to use this WLAN, click "Use"

3.7 Open the Robot APP on the mobile phone and click "Connect"



* 1. After the connection is successful, the Monitor box will display "Received: Welcome to connect to ESP8266-01, ID: 0""or "Connect Finished".

If unable to connect, please exit the app, power on the control board again, reconnect the phone to the "Cokoino-ESP8266-01" WLAN signal, and then open the app and click on "Connect".Generally, a re operation can be successful.





3.9 Congratulations, the APP has been successfully connected to the ESP-01 module, and you can start to control the car on the APP interface of the mobile phone.

# 4. Troubleshooting

4.1 Unable to upload code successfully

Before uploading the code, please check whether the ESP-01 switch on the control board is turned to the side away from the "ESP-01" silk screen.

If it still fails, please plug and unplug the USB cable again, and then upload the program. Generally, a re operation can be successful.

4.2 Cannot find the WLAN signal of ESP-01 module

After the code is successfully uploaded, the switch of the ESP-01 needs to be turned to the side of the "ESP-01" silk screen.

Check if the ESP-01 module is plugged into the correct position on the control board

4.3 The car moves slowly or does not move, and the "hum" sound of the motor can be heard

Check the power of the 18650 battery, if the battery level is low and below 7V, it is recommended to charge it before use. If the battery level is between 7~8V, you can try increasing the motor speed in the code. The battery level matched with the motor speed in the example code is between 8V and 8.4V.

# 5. Code

Usually, there are two basic main functions for Arduino code, void setup() and void loop().

void setup(){ }

The setup() function is called when a sketch starts, which is used to initialize variables, pin modes, start

using libraries, etc.

The setup() function will **only run once**, after each **power up** or **reset** of the Arduino board.

void loop(){ }

This function will loop consecutively. Code in this function will be executed again and again…

There are some other functions integrated by Arduino.

pinMode(pin, mode)

Configures the specified pin to behave either as an input or an output.

Parameters

pin: Arduino pin number to set the mode of.

mode: INPUT, OUTPUT, or INPUT\_PULLUP.

digitalWrite(pin, value)

Write a HIGH or a LOW value to a digital pin.

If the pin has been configured as an OUTPUT with pinMode(), its voltage will be set to the corresponding

value: 5V (or 3.3V on 3.3V boards) for HIGH, 0V (ground) for LOW.

Parameters

pin: Arduino pin number.

value: HIGH or LOW.

analogWrite(pin, value)

Writes an analog value (PWM wave) to a pin.

You do not need to call pinMode() to set the pin as an output before calling analogWrite().

Parameters

pin: Arduino pin to write to. Allowed data types: int.

value: the duty cycle: between 0 (always off) and 255 (always on). Allowed data types: int.

For more details, please refer to: https://www.arduino.cc/reference/

14\_1\_Wifi\_Controlled\_Car.ino:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 \* This code applies to cokoino smart robot car kit

 \* Through this link you can download the source code:

 \* https://github.com/Cokoino/CKK0002

 \* Company web site:

 \* http://cokoino.com/

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <Regexp.h>

#include <LiquidCrystal\_I2C.h>

LiquidCrystal\_I2C lcd(0x27, 16, 2);

#include <Wire.h>

#include <Adafruit\_PWMServoDriver.h>

Adafruit\_PWMServoDriver pwm = Adafruit\_PWMServoDriver();

#include <Servo.h>

Servo carservo;

int pos=0;

#include <Adafruit\_NeoPixel.h>

#ifdef \_\_AVR\_\_

#include <avr/power.h> // Required for 16 MHz Adafruit Trinket

#endif

#define WS2812\_PIN 6   //WS2812 PIN

#define WS2812\_COUNT  12  // How many NeoPixels are attached to the Arduino?

#define BRIGHTNESS 10  // NeoPixel brightness, 0 (min) to 255 (max)

// Declare our NeoPixel strip object:

Adafruit\_NeoPixel strip = Adafruit\_NeoPixel(WS2812\_COUNT, WS2812\_PIN, NEO\_GRB + NEO\_KHZ800);

// Argument 1 = Number of pixels in NeoPixel strip

// Argument 2 = Arduino pin number (most are valid)

// Argument 3 = Pixel type flags, add together as needed:

//   NEO\_KHZ800  800 KHz bitstream (most NeoPixel products w/WS2812 LEDs)

#define Trig\_Pin 13 //trig PIN

#define Echo\_Pin 12  //echo PIN

#define Buzz 11     //buzzer PIN

#define led\_R 9     //right green led PIN

#define led\_L 5     //left green led PIN

#define Line\_L A0   //left line PIN

#define Line\_M A1   // middle line PIN

#define Line\_R A2   //right line  PIN

#define distance\_minimum 30 //The minimum obstacle distance is defined as 30cm

float distance,distance\_0,distance\_130;//Import the middle, right, and left distance variables

int randNumber=0;

int L\_Distance=0;

int M\_Distance=0;

int R\_Distance=0;

float cm;

// regular

MatchState ms;

/// Car driving direction control button on the app interface, a total of 8 direction buttons.

const String phone1 = "fS";  // forwardStart:

const String phone1\_5 = "lfS";  // forward\_left\_Start

const String phone2 = "lS";    // leftStart

const String phone2\_5 = "lbS";  // left\_backward\_Start

const String phone3 = "bS";    // backwardStart

const String phone3\_5 = "rbS";  // backward\_right\_Start

const String phone4 = "rS";    // rightStart

const String phone4\_5 = "rfS";  // right\_forward\_Start

/// The other function buttons on the app interface

const String phone5 = "OK";//stop

const String phone6 = "rtl";//rotation left

const String phone7 = "rtr";//rotation right

const String phone8 = "trk";//track line running

const String phone9 = "aod";//Avoid obstacles

const String phone10 = "lgt";//light show

const String phone11 = "muc";//buzzer

const String phone12 = "flt";//fllow light

const String phone13 = "bt1";//button1

String comdata = "";//import the comdata string

char judge = 0;//init the judge

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*Set the melody and rhythm of the music\*\*\*\*\*\*\*\*\*\*

//Tenor NTF 0 is an empty beat

#define NTF0 -1

#define NTF1 350

#define NTF2 393

#define NTF3 441

#define NTF4 495

#define NTF5 556

#define NTF6 624

#define NTF7 661

//High pitch NTFH

#define NTFH1 700

#define NTFH2 786

#define NTFH3 882

#define NTFH4 935

#define NTFH5 965

#define NTFH6 996

#define NTFH7 1023

//Low pitch NTFH

#define NTFL1 175

#define NTFL2 196

#define NTFL3 221

#define NTFL4 234

#define NTFL5 262

#define NTFL6 294

#define NTFL7 330

//Note frequency array

int tune[]=

{

  NTF3,NTF3,NTF3,NTF3,NTF3,NTF3,

  NTF3,NTF5,NTF1,NTF2,NTF3,NTF0,

  NTF4,NTF4,NTF4,NTF4,NTF4,NTF3,NTF3,NTF3,NTF3,

  NTF5,NTF5,NTF4,NTF2,NTF1,NTF0,

  NTFL5,NTF3,NTF2,NTF1,NTFL5,NTF0,NTFL5,NTFL5,

  NTFL5,NTF3,NTF2,NTF1,NTFL6,NTF0,

  NTFL6,NTF4,NTF3,NTF2,NTFL7,NTF0,

  NTF5,NTF5,NTF4,NTF2,NTF3,NTF1,NTF0,

  NTFL5,NTF3,NTF2,NTF1,NTFL5,NTF0,

  NTFL5,NTF3,NTF2,NTF1,NTFL6,NTF0,NTFL6,

  NTFL6,NTF4,NTF3,NTF2,NTF5,NTF5,NTF5,NTF5,

  NTF6,NTF5,NTF4,NTF2,NTF1,NTF0

};

//Note beat array

float durt[]=

{

  0.5,0.5,1,0.5,0.5,1,

  0.5,0.5,0.75,0.25,1.5,0.5,

  0.5,0.5,1,0.5,0.5,0.5,0.5,0.25,0.25,

  0.5,0.5,0.5,0.5,1.5,0.5,

  0.5,0.5,0.5,0.5,1,0.5,0.25,0.25,

  0.5,0.5,0.5,0.5,1,1,

  0.5,0.5,0.5,0.5,1,1,

  0.5,0.5,0.5,0.5,1,0.75,0.25,

  0.5,0.5,0.5,0.5,1,1,

  0.5,0.5,0.5,0.5,1,0.5,0.5,

  0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,

  0.5,0.5,0.5,0.5,0.75,0.25

};

//Define the buzzer pin, note length variable

int length;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void setup() {

  Serial.begin(115200);

  pwm.begin();

  pwm.setPWMFreq(50); // Set the PWM frequency as 50

  ///Initialize the motor state

  pwm.setPWM(2, 0, 0);

  pwm.setPWM(3, 0, 0);

  pwm.setPWM(4, 0, 0);

  pwm.setPWM(5, 0, 0);

  delay(100);  // If the information printed out of the serial port is garbled, extend the delay time to solve the problem.

  while (Serial.read() >= 0)

    continue;

  Serial.flush();

  ESP8266\_ATCOMMAND();//esp-01 module AT instruction function

  lcd.init();

  lcd.backlight();

  lcd.clear();

  pinMode(Buzz, OUTPUT);

  pinMode(led\_R, OUTPUT);

  pinMode(led\_L, OUTPUT);

  pinMode(Trig\_Pin, OUTPUT);

  pinMode(Echo\_Pin, INPUT\_PULLUP);

  carservo.attach(10);//servo PIN

  carservo.write(65);//Initialize the car head in the middle position

  length = sizeof(tune)/sizeof(tune[0]);

  strip.begin();

  strip.show();

  strip.setBrightness(BRIGHTNESS);

}

void loop() {

  while (Serial.available() > 0) {

    comdata += char(Serial.read());

    delay(1);

  }

  judgement();

}

// ESP8266 set the AT instructionS

void ESP8266\_ATCOMMAND() {

  Serial.print(F("AT+RST\r\n"));  //F(): Store string constants in Flash flash to avoid memory depletion due to SRAM usage.

  delay(3000);

  Serial.print(F("AT+CWMODE=3\r\n"));  //set to softAP+station mode

  delay(300);

  Serial.print(F("AT+CWSAP=\"Cokoino\_ESP8266-01\",\"12345678\",11,2\r\n"));//wifiname:Cokoino\_ESP8266-01,wifipassword:12345678

 //channnel:11 Encryption mode:2 ;Encryption mode should not set to 1,otherwise the wifi can't set successed

  delay(200);

  Serial.print(F("AT+CIPMUX=1\r\n"));//Enable multiple connections

  delay(200);

  Serial.print(F("AT+CIPSERVER=1,3001\r\n"));//Create the server. The default port is 333. Change the port to 3001, which is consistent with the APP

  delay(200);

  Serial.print(F("AT+CIPSTO=7000\r\n"));//Example Set the server timeout period to 7000 seconds

  delay(2000);

}

void judgement() {

  if (comdata.length() > 0) {

    comdata += "\n";  //This sentence must be added, otherwise the matched command character is one less, and the newline is used to assist in the complete match.

    char buf[comdata.length()];

    comdata.toCharArray(buf, comdata.length());

    ms.Target(buf);

    char result = ms.Match("%c\*%+IPD, ?[0-9]+, ?[0-9]+: ?([^%c]+)%c\*$");

    if (result > 0) {

      ms.GetCapture(buf, 0);

      comdata = String(buf);

      lcd.clear();

      lcd.setCursor(0, 0);

      lcd.print(comdata);

      delay(100);

    } else {

      result = ms.Match("%c\*%s?([0-9]),%s?([^%c]+)%c\*$");  // esp8266 Multi-channel supports up to 5 connections (id:0-4)

      if (result > 0) {

        char buf0[1];  // esp8266 In multi-channel mode. id of the connection at this time

        ms.GetCapture(buf0, 0);

        ms.GetCapture(buf, 1);

        comdata = String(buf);

        if (comdata == "CONNECT")//The APP successfully connects to the wifi of ESP-01 module

        {

          String receiveOkMs = "Welcome to connect to ESP8266-01, ID: " + String(buf0) + " .";//A successful connection message is displayed

          Serial.println("AT+CIPSEND=" + String(buf0) + "," + receiveOkMs.length() + "\r\n");

          delay(10);

          Serial.print(receiveOkMs);

          lcd.clear();

          lcd.setCursor(0, 0);

          lcd.print(String(buf0) + ",CONNECT ");

          delay(1500);

          lcd.setCursor(0, 1);

          lcd.print("MSG\_Len:");

          lcd.setCursor(9, 1);

          lcd.print(String(receiveOkMs.length()) + "Bytes");  // If the combined variable is a non-string, it needs to be converted to a string for normal display.

          delay(2000);

          lcd.clear();

          lcd.setCursor(0, 0);

          lcd.print("Available Memory");

          lcd.setCursor(0, 1);

          lcd.print(": " + String(availableMemory()));  // If the combined variable is a non-string, it needs to be converted to a string for normal display.

          delay(2000);

        }

      }

      lcd.clear();

      lcd.setCursor(1, 0);

      lcd.print("Not a APP\_CMD! ");

    }

    //comdata = "";

    //return; // When debugging communication with the APP, it needs to be commented out when normal use

     if (comdata == phone1) {

       judge = 1;

     }

      else if (comdata == phone1\_5) {

       judge = 2;

     } else if (comdata == phone2) {

       judge = 3;

     } else if (comdata == phone2\_5) {

       judge = 4;

     } else if (comdata == phone3) {

       judge = 5;

     } else if (comdata == phone3\_5) {

       judge = 6;

     } else if (comdata == phone4) {

       judge = 7;

     } else if (comdata == phone4\_5) {

       judge = 8;

     } else if (comdata == phone5) {

       judge = 9;

     } else if (comdata == phone6) {

       judge = 10;

     } else if (comdata == phone7) {

       judge = 11;

     } else if (comdata == phone8) {

       judge = 12;

     } else if (comdata == phone9) {

       judge = 13;

     } else if (comdata == phone10) {

       judge = 14;

     } else if (comdata == phone11) {

       judge = 15;

     } else if (comdata == phone12) {

       judge = 16;

     } else if (comdata == phone13) {

       judge = 17;

     }

     else {

       judge = 9;

     }

    comdata = "";

  }

  switch (judge) {

    case 1:

      forward();

      break;

    case 3:

      turnLeft();

      break;

    case 5:

      backward();

      break;

    case 7:

      turnRight();

      break;

    case 9:

      Stopcar();

      break;

    case 10:

      left\_rotation();

      break;

    case 11:

      right\_rotation();

      break;

    case 12:

      track\_line();

      break;

    case 13:

      obstacle\_avoidance();

      break;

    case 14:

      light\_show();

      delay(2000);

      judge = 9;

      break;

    case 15:

      music();

      delay(2000);

      judge = 9;

      break;

    case 17:

      shake\_head();

      delay(2000);

      judge = 9;

      break;

    default: break;

  }

}

//pwm.setPWM(pwmnum,on, off);

     //(pwmnum, on, off) function is mainly to adjust the output PWM duty cycle.

     // Usually, on is set to 0 and off can be changed.

     // Because the PCA9685 is a 12-bit resolution

     // the value of 0 to 4096 off represents a duty cycle of 0 to 100.

void forward()

{

  ////The off value is based on the battery level ranging from 8V to 8.4V. /////////////////////////////////

 ////If your battery level is below 8V, you can increase the off value to allow the car to move normally.//

 // drive M1 Motror forward

 pwm.setPWM(2, 0, -650);//set pwm signal to BIN2 of DRV8833

 pwm.setPWM(3, 0, 650);//set pwm signal to BIN1 of DRV8833

 //drive M4 Motror forward

 pwm.setPWM(4, 0, 650);//set pwm signal to AIN1 of DRV8833

 pwm.setPWM(5, 0, -650);//set pwm signal to AIN2 of DRV8833

}

void backward()

{

 pwm.setPWM(2, 0, 750);

 pwm.setPWM(3, 0, -750);

 pwm.setPWM(4, 0, -750);

 pwm.setPWM(5, 0, 750);

}

void turnLeft()

{

 pwm.setPWM(2, 0, -650);

 pwm.setPWM(3, 0, 650);

 pwm.setPWM(4, 0, 1200);

 pwm.setPWM(5, 0, -1200);

}

void turnRight()

{

 pwm.setPWM(2, 0, -1200);

 pwm.setPWM(3, 0, 1200);

 pwm.setPWM(4, 0, 650);

 pwm.setPWM(5, 0, -650);

}

void right\_rotation()

{

 pwm.setPWM(2, 0, -1200);

 pwm.setPWM(3, 0, 1200);

 pwm.setPWM(4, 0, -1200);

 pwm.setPWM(5, 0, 1200);

}

void left\_rotation()

{

 pwm.setPWM(2, 0, 1200);

 pwm.setPWM(3, 0, -1200);

 pwm.setPWM(4, 0, 1200);

 pwm.setPWM(5, 0, -1200);

}

void Stopcar()

{

 pwm.setPWM(2, 0, 0);

 pwm.setPWM(3, 0, 0);

 pwm.setPWM(4, 0, 0);

 pwm.setPWM(5, 0, 0);

}

void music()

{

   for(int x=0;x<length;x++)

  {

    tone(Buzz, tune[x]);

    delay(500\*durt[x]);         //The 500 here controls the length of each note to determine the rhythm of the piece

    noTone(Buzz);

  }

  delay(500);               //The interval between starting the next cycle

}

///////////////////////////////////////////////////////////Automatic obstacle avoidance

void track\_line()

{

  u8 trackingSensorVal = 0;

  trackingSensorVal = getTrackingSensorVal(); //get sensor value

  switch (trackingSensorVal)

  {

    case 0:   //000

      backward();//car backward

      delay(15);

      break;

    case 7:   //111

      forward(); //car forward

      break;

    case 1:   //001

      turnRight(); //car turn Right

      delay(120);

      break;

    case 3:   //011

      turnRight(); //car turn Right

      delay(120);

      break;

    case 2:   //010

    case 5:   //101

    case 6:   //110

      turnLeft(); //car turn left

      delay(120);

      break;

    case 4:   //100

      turnLeft(); //car turn left

      delay(120);

      break;

    default:

      break;

  }

}

u8 getTrackingSensorVal() {

  u8 trackingSensorVal = 0;

  trackingSensorVal = (digitalRead(Line\_L) == 1 ? 1 : 0) << 2 | (digitalRead(Line\_M) == 1 ? 1 : 0) << 1 | (digitalRead(Line\_R) == 1 ? 1 : 0) << 0;

  return trackingSensorVal;

}

///////////////////////////////////////////////////////////Automatic obstacle avoidance

void obstacle\_avoidance()

{

   digitalWrite(led\_R,HIGH);//light the right green LED

   digitalWrite(led\_L,HIGH);//light the left green LED

   ultrasonic\_distance();

  //Serial.println(distance);

  if(distance>distance\_minimum)

    {

    forward();   //forward

    }

  if(distance<=distance\_minimum)

    {

    Stopcar();     //stop

    buzz();       //the buzzer sounds

    servo\_wheel(); //The servo rotates, and the ultrasonic module identifies the distance

                   //of the 65-degree obstacle in the left front of the car body

    if((distance\_0<=distance\_minimum)&&(distance\_130<=distance\_minimum))

      {backward();//car backward

      delay(300); }

    if(distance\_0<distance\_130)//The right obstacle is less distant than the left

      {left\_rotation();

      delay(80);

     }

    if(distance\_0>distance\_130)//The left obstacle is less distant than the right

      {right\_rotation();

      delay(80);

      }

    if(distance\_0==distance\_130)

      {

      randNumber = random(1, 2); //randnumber

      if(randNumber==1)

        {turnLeft();delay(200);

       }

      if(randNumber==2)

        {turnRight();delay(200);

       }

      }

    delay(250);

    Stopcar();     //car stop

    }

}

///////////////////////////////////////////////////////////ultrasonic ranging

void ultrasonic\_distance()

{

delay(100);

digitalWrite(Trig\_Pin, HIGH);

delayMicroseconds(10);

digitalWrite(Trig\_Pin, LOW);

distance = pulseIn(Echo\_Pin, HIGH) \* 340 /2/ 10000.0;

if(distance==0)

  distance=300;

delay(100);

}

///////////////////////////////////////////////////////////servo rotation

void servo\_wheel()

{

carservo.write(0);

delay(250);

ultrasonic\_distance();

distance\_0=distance;

delay(250);

carservo.write(130);

delay(250);

ultrasonic\_distance();

distance\_130=distance;

delay(250);

carservo.write(65);

delay(300);

}

///////////////////////////////////////////////////////////buzzer sounds

void buzz()

{

  for(int i = 0;i < 100; i++)

  {

    digitalWrite(Buzz, HIGH);

    delay(1);

    digitalWrite(Buzz, LOW);

    delay(1);

  }

  for(int j = 0;j< 180; j++)

  {

    digitalWrite(Buzz, HIGH);

    delay(2);

    digitalWrite(Buzz, LOW);

    delay(2);

  }

}

///////////////////////////////////////////////////////////the car shake it's head

void shake\_head()

{

  for (pos = 0; pos <= 130; pos += 1) { // goes from 0 degrees to 130 degrees

    // in steps of 1 degree

    carservo.write(pos);              // tell servo to go to position in variable 'pos'

    delay(15);                       // waits 15 ms for the servo to reach the position

  }

  for (pos = 130; pos >= 0; pos -= 1) { // goes from 130 degrees to 0 degrees

    carservo.write(pos);              // tell servo to go to position in variable 'pos'

    delay(15);                       // waits 15 ms for the servo to reach the position

  }

  delay(50);

  carservo.write(65);               //Reset the head of the car

}

int availableMemory() {

  // Use 1024 with ATmega328

  int size = 2048;

  byte \*buf;

  while ((buf = (byte \*)malloc(--size)) == NULL);

  free(buf);

  return size;

}

///////////////////////////////////////////////////////////light show

void light\_show()

{

  LED\_show();

  WS2812\_show();

}

void LED\_show()

{

  for(int i = 0;i < 15; i++)

  {

    digitalWrite(led\_R,HIGH);

    digitalWrite(led\_L,HIGH);

    delay(50);

    digitalWrite(led\_R,LOW);

    digitalWrite(led\_L,LOW);

    delay(50);

    digitalWrite(led\_R,HIGH);

    digitalWrite(led\_L,HIGH);

    delay(50);

  }

}

void WS2812\_show()

{

  colorWipe(strip.Color(255, 0, 0), 10); // Red

  delay(800);

  colorWipe(strip.Color(255, 150, 0), 10); // yellow

  delay(800);

  colorWipe(strip.Color(0, 255, 0), 10); // Green

  delay(800);

  colorWipe(strip.Color(0, 255, 255), 10); // CYAN

  delay(800);

  colorWipe(strip.Color(0, 0, 255), 10); // Blue

  delay(800);

  colorWipe(strip.Color(180, 0, 255), 10); // purple

  delay(800);

  colorWipe(strip.Color(127, 127, 127), 10); // White

  delay(800);

  colorWipe(strip.Color(0, 0, 0), 30); // Clear

  Serial.println("OK");

}

void colorWipe(uint32\_t c, uint8\_t wait)

{

  for(uint16\_t i=0; i<strip.numPixels(); i++) {

    strip.setPixelColor(i, c);

    strip.show();

    delay(wait);

  }

}

//Theatre-style crawling lights.

void theaterChase(uint32\_t c, uint8\_t wait) {

  for (int j=0; j<10; j++) {  //do 10 cycles of chasing

    for (int q=0; q < 3; q++) {

      for (int i=0; i < strip.numPixels(); i=i+3) {

        strip.setPixelColor(i+q, c);    //turn every third pixel on

      }

      strip.show();

      delay(wait);

      for (int i=0; i < strip.numPixels(); i=i+3) {

        strip.setPixelColor(i+q, 0);        //turn every third pixel off

      }

    }

  }

}

# 6. Any questions and suggestions are welcome

Thank you for reading this document!

If you find any errors and omissions in the tutorial, or if you have any suggestions and questions, please feel free to contact us:

**[cokoino@outlook.com](mailto:cokoino@outlook.com)**

We will do our best to make changes and publish revisions as soon as possible.

If you want to learn more about Arduino, Raspberry Pi, Smart Cars, Robotics and other interesting products in science and technology, please continue to visit our Amazon Store by search for "**LK COKOINO**" on Amazon. We will continue to launch fun, cost-effective, innovative and exciting products.

Thank you again for choosing Cokoino products.

**LK COKOINO**