Welcome

Thank you for choosing Freenove products!

About Battery

First, read the document **About_Battery.pdf** in the unzipped folder.
If you did not download the zip file, please download it and unzip it via link below:
https://github.com/Freenove/Freenove_4WD_Car_Kit_for_ESP32

Get Support and Offer Input

Freenove provides free and responsive product and technical support, including but not limited to:

- Product quality issues
- Product use and build issues
- Questions regarding the technology employed in our products for learning and education
- Your input and opinions are always welcome
- We also encourage your ideas and suggestions for new products and product improvements

For any of the above, you may send us an email to:

support@freenove.com

Safety and Precautions

Please follow the following safety precautions when using or storing this product:

- Keep this product out of the reach of children under 6 years old.
- This product should be used only when there is adult supervision present as young children lack necessary judgment regarding safety and the consequences of product misuse.
- This product contains small parts and parts, which are sharp. This product contains electrically conductive parts. Use caution with electrically conductive parts near or around power supplies, batteries and powered (live) circuits.
- When the product is turned ON, activated or tested, some parts will move or rotate. To avoid injuries to hands and fingers, keep them away from any moving parts!
- It is possible that an improperly connected or shorted circuit may cause overheating. Should this happen, immediately disconnect the power supply or remove the batteries and do not touch anything until it cools down! When everything is safe and cool, review the product tutorial to identify the cause.
- Only operate the product in accordance with the instructions and guidelines of this tutorial, otherwise parts may be damaged or you could be injured.
- Store the product in a cool dry place and avoid exposing the product to direct sunlight.
- After use, always turn the power OFF and remove or unplug the batteries before storing.

About Freenove

Freenove provides open source electronic products and services worldwide.

Freenove is committed to assist customers in their education of robotics, programming and electronic circuits so that they may transform their creative ideas into prototypes and new and innovative products. To this end, our services include but are not limited to:

- Educational and Entertaining Project Kits for Robots, Smart Cars and Drones
- Educational Kits to Learn Robotic Software Systems for Arduino, Raspberry Pi, micro: bit and ESP32.
- Electronic Component Assortments, Electronic Modules and Specialized Tools
- **Product Development and Customization Services**

You can find more about Freenove and get our latest news and updates through our website:

http://www.freenove.com

Copyright

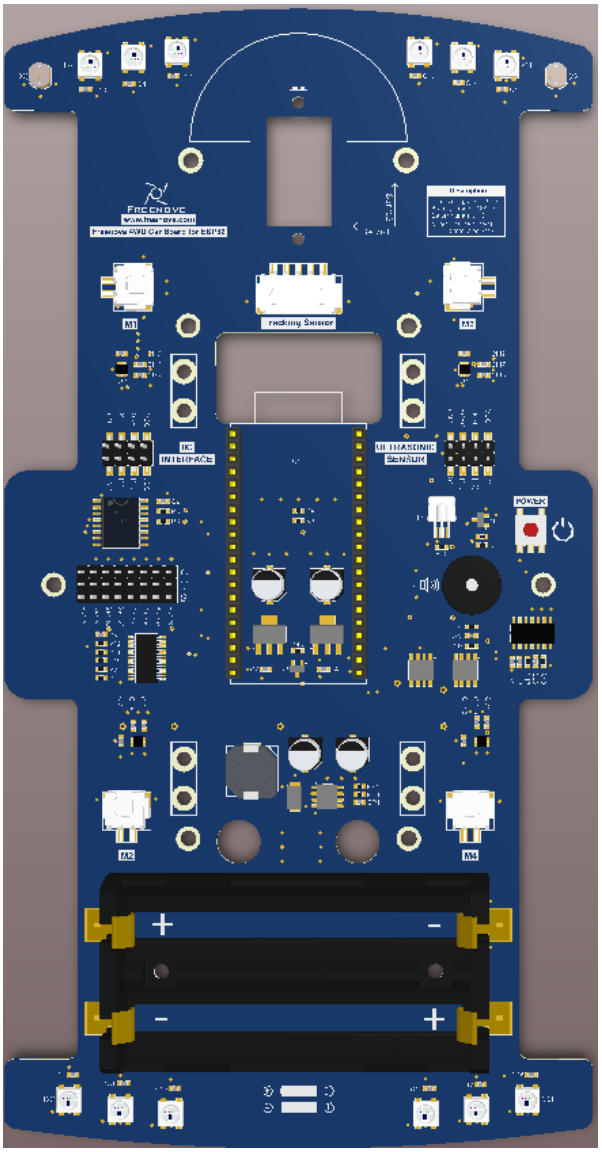
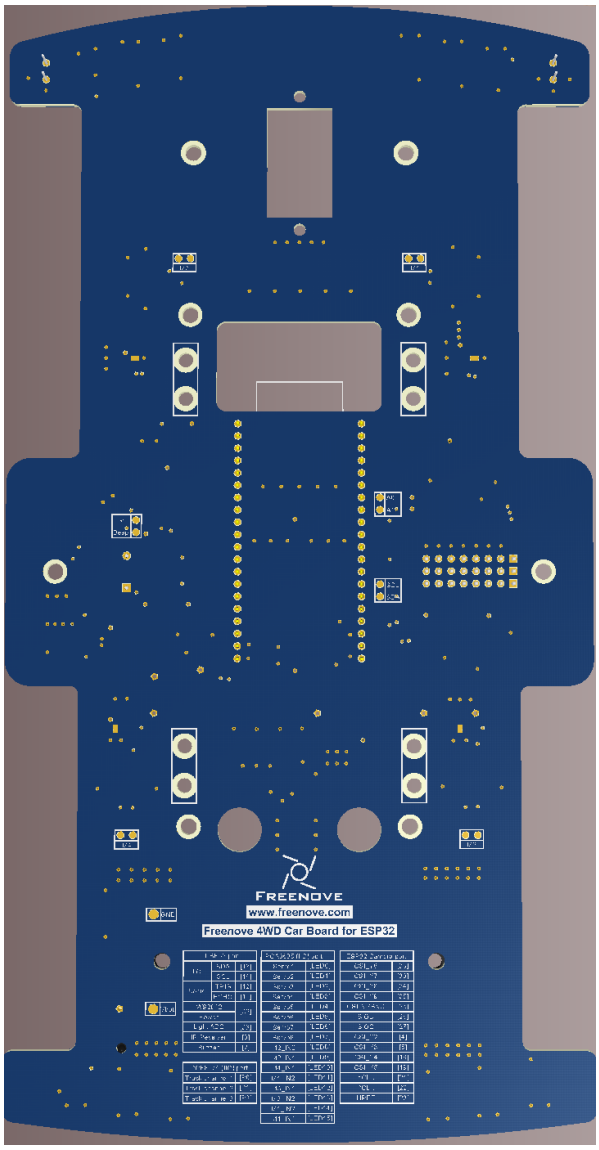
FREENOVE™

Contents

List

If you have any concerns, please feel free to contact us via support@freenove.com

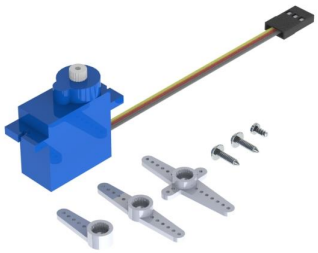
ESP32 Car Shield

Top	Bottom
	

Machinery Parts

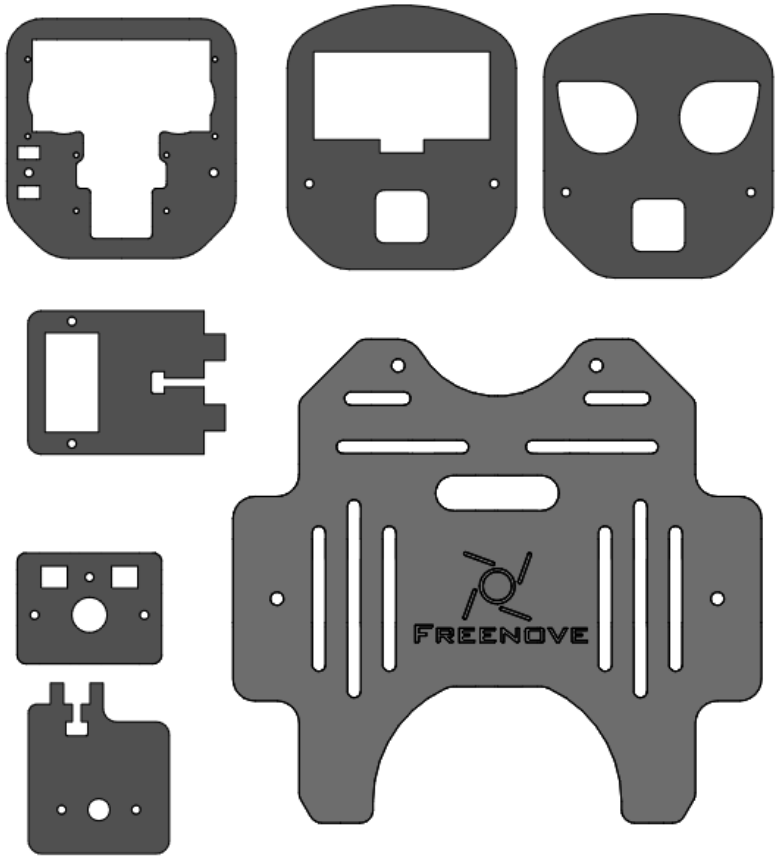
				
				

Transmission Parts

Servo package x2 	Driven wheel x4 
DC speed reduction motor x4 	Motor bracket package x4 

Acrylic Parts

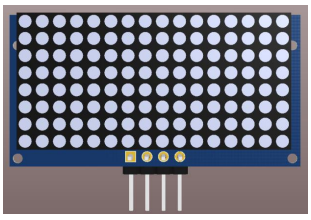
Acrylic List

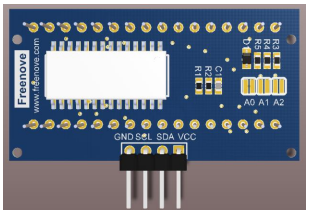


Electronic Parts

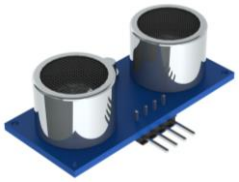
Line tracking module x1

Dot Matrix Module x1

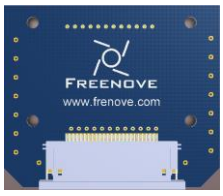


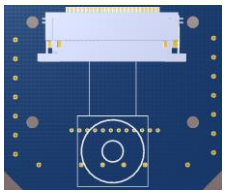


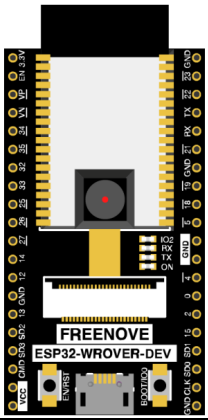
HC-SR04 Ultrasonic Module x1



Camera connector x1



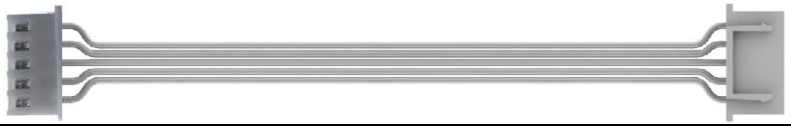


ESP32 x1	Infrared emitter x1
	

Wires

Jumper Wire F/F(4) x1

FPC Wire x1

XH-2.54-5Pin cable x1


Tools

Cross screwdriver (3mm) x1	Black tape x1	Cable Tidy x15cm
		

Required but NOT Contained Parts

2 x 3.7V 18650 lithium **rechargeable** batteries with continuous discharge current >3A.
It is easier to find proper battery on eBay than Amazon. Search "18650 high drain" on eBay.

Preface

Welcome to use Freenove 4WD Car Kit for ESP32. Following this tutorial, you can make a very cool car with many functions.

Based on the ESP32 development board, a popular IoT control board, this kit uses the very popular Arduino IDE for programming, so you can share and exchange your experience and design ideas with enthusiasts all over the world. The parts in the kit include all electronic components, modules, and mechanical components required for making the vcar. They are all individually packaged. There are detailed assembly and debugging instructions in this book. If you encounter any problems, please feel free to contact us for fast and free technical support.

support@freenove.com

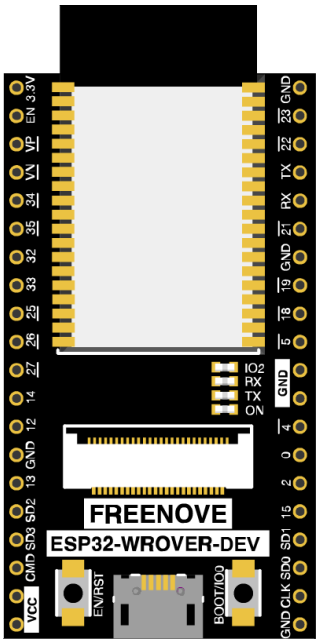
This car does not require a high threshold for users. Even if you know little professional knowledge, you can make your own smart car easily with the guidance of the tutorial. If you're really interested in ESP32 and hope to learn how to program and build circuits, please visit our website: www.freenove.com or contact us to buy our kit designed for beginners: **Freenove Ultimate Kit for ESP32**.

ESP32

ESP32-WROVER has launched a total of two antenna packages, PCB on-board antenna and IPEX antenna respectively. The PCB on-board antenna is an integrated antenna in the chip module itself, so it is convenient to carry and design. The IPEX antenna is a metal antenna derived from the integrated antenna of the chip module itself, which is used to enhance the signal of the module.

PCB on-board antenna	IPEX antenna
	

In this tutorial, the ESP32-WROVER is designed based on PCB on-board antenna packaged ESP32-WROVER module. All projects in this tutorial are based on this board.

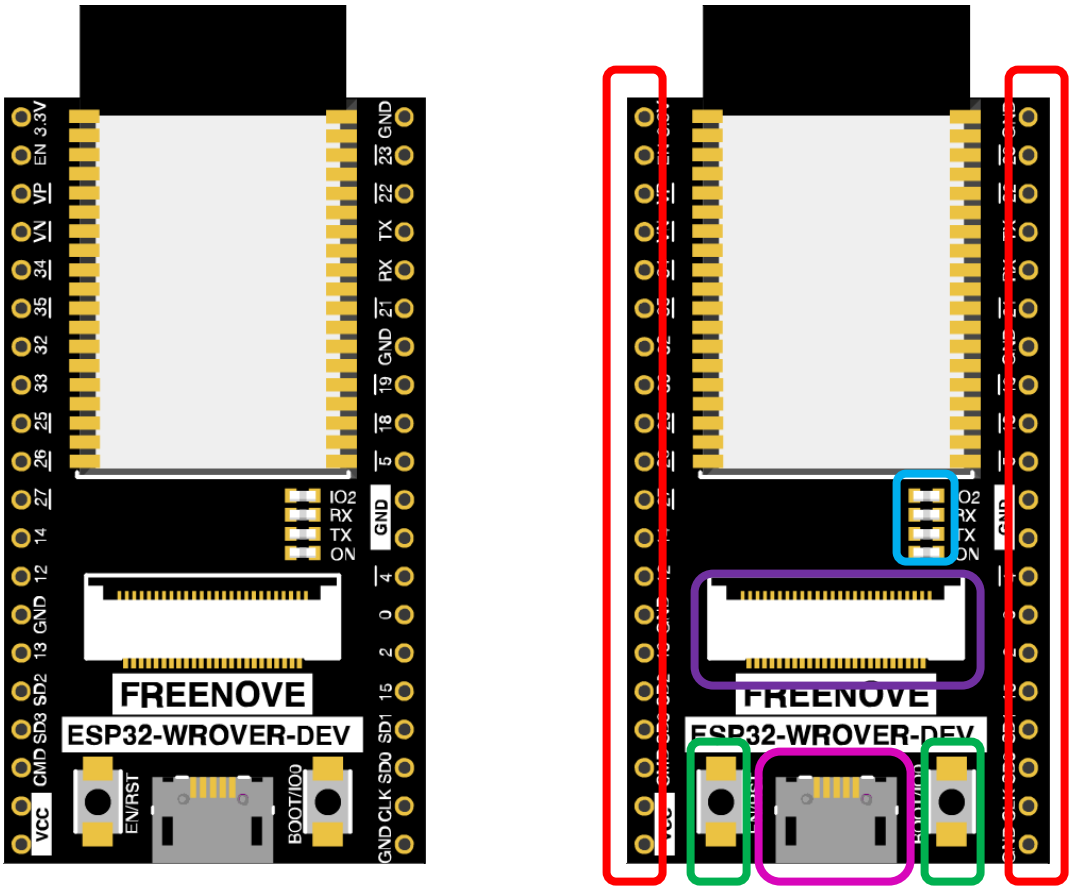


Pins of the Car

If you don't know the what each GPIO corresponds to, you can refer to the following table.
The functions of the pins are allocated as follows:

Pins of ESP32	Funtions	Description
GPIO36	Camera interface	CSI_Y6
GPIO39		CSI_Y7
GPIO34		CSI_Y8
GPIO35		CSI_Y9
GPIO25		CSI_VYSNC
GPIO26		SIOD
GPIO27		SIOC
GPIO4		CSI_Y2
GPIO5		CSI_Y3
GPIO18		CSI_Y4
GPIO19		CSI_Y5
GPIO21		XCLK
GPIO22		PCLK
GPIO23		HREF
GPIO13	I2C port	SDA
GPIO14		SCL
GPIO32	Battery detection / WS2812	A6
GPIO33	Search light ADC port	A7
GPIO12	Ultrasonic module interface	Trig
GPIO15		Echo
GPIO2	Buzzer port	Buzzer
GPIO0	Infrared receiver port	IR
GPIO1	Serial port	TX
GPIO3		RX

The hardware interfaces of ESP32 are distributed as follows:

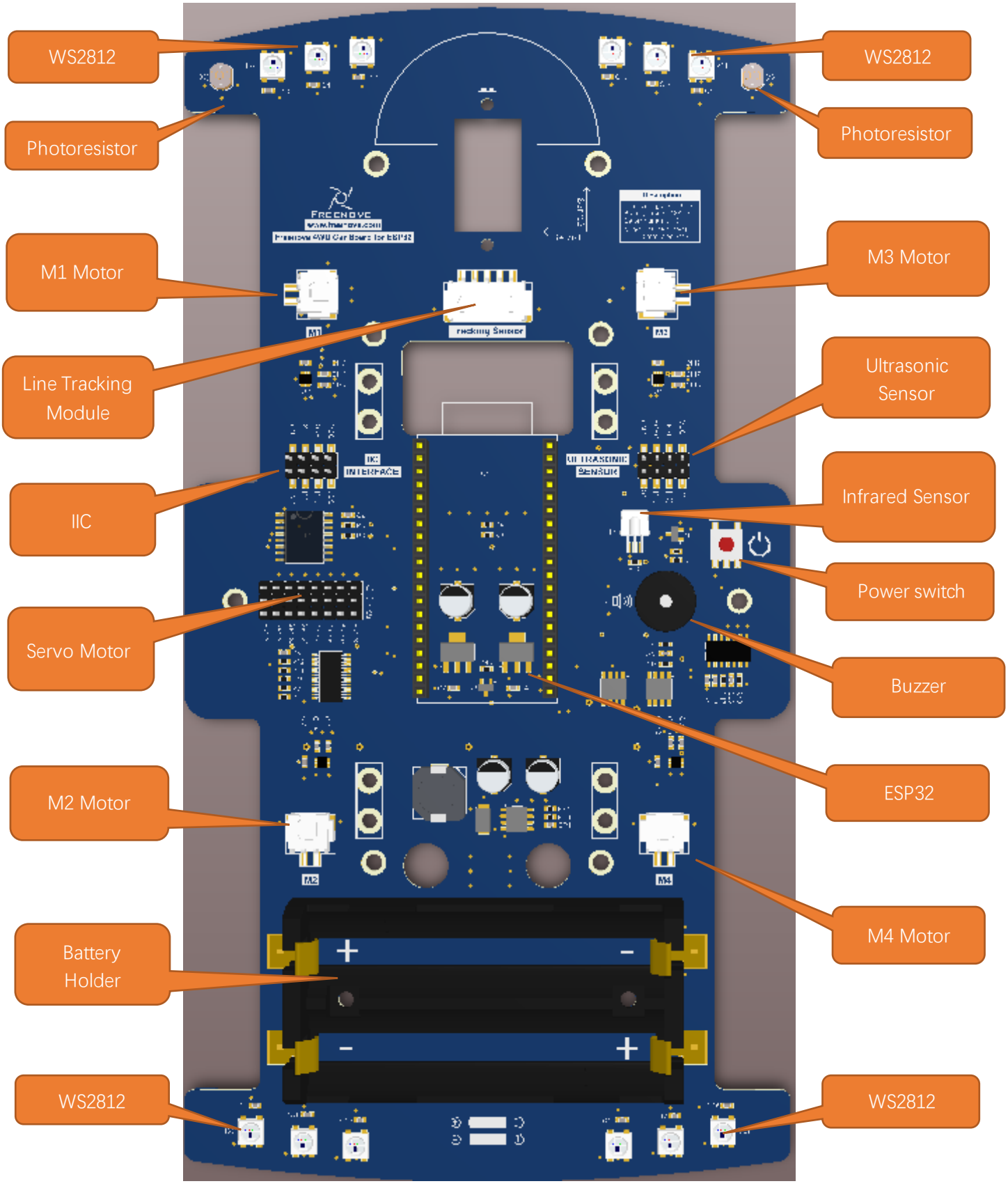


Compare the left and right images. We've boxed off the resources on the ESP32 in different colors to facilitate your understanding of the ESP32.

Box color	Corresponding resources introduction
	GPIO pin
	LED indicator
	Camera interface
	Reset button, Boot mode selection button
	USB port

Introduction of the Car

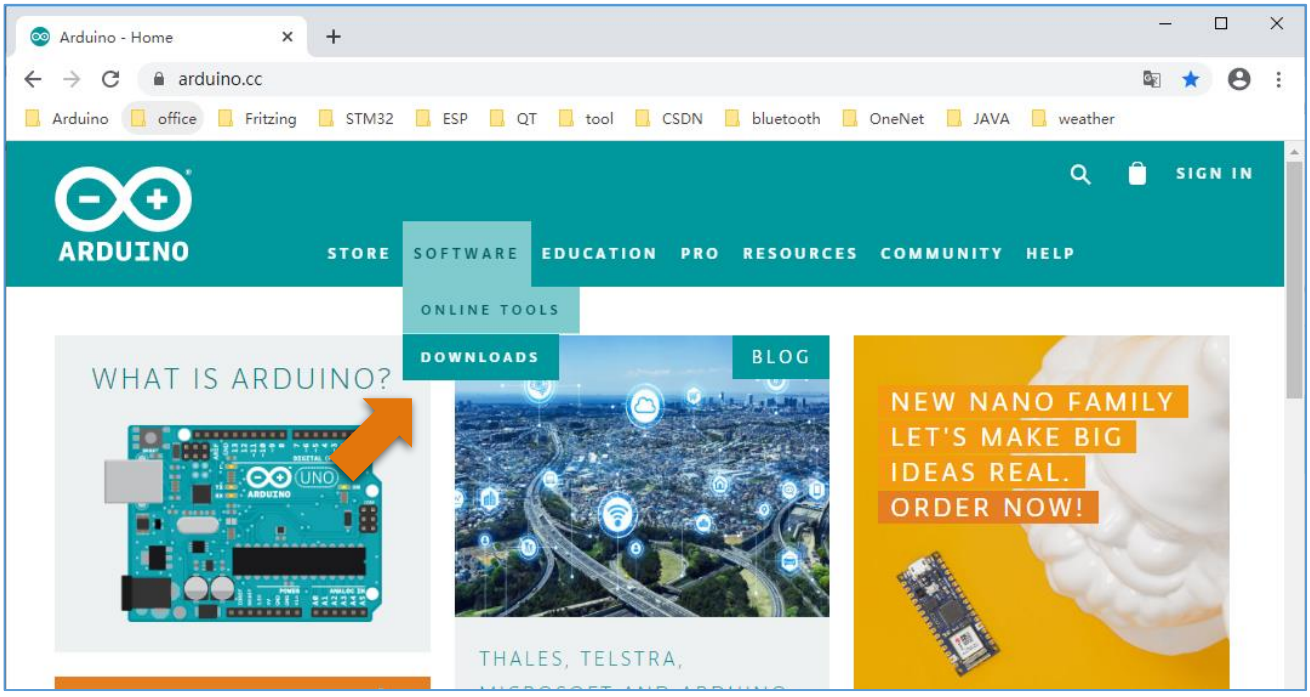
The function diagram of the ESP32 car is as follows:



Chapter 0 Installation of Arduino IDE

Arduino Software

Arduino Software (IDE) is used to write and upload the code for Arduino Board.
First, install Arduino Software (IDE): visit https://www.arduino.cc, click "Download" to enter the download page.

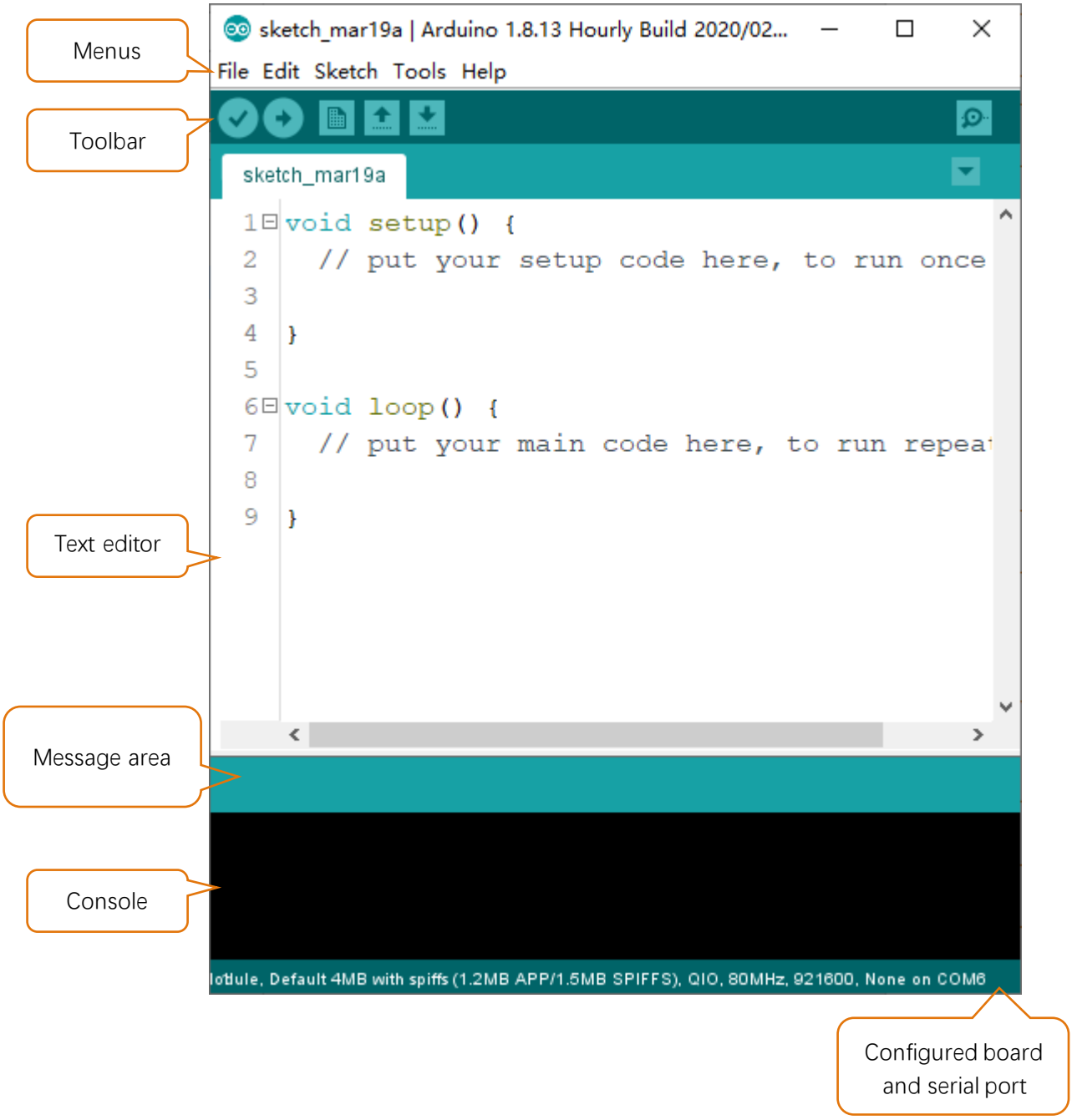


Select and download corresponding installer according to your operating system. If you are a Windows user, please select the "Windows Installer" to download and install the driver correctly.

After the download completes, run the installer. For Windows users, there may pop up an installation dialog during the installation. When it popes up, please allow the installation.

After installation is completed, an Arduino Software shortcut will be generated in the desktop. Run the Arduino Software.

The interface of Arduino Software is as follows:



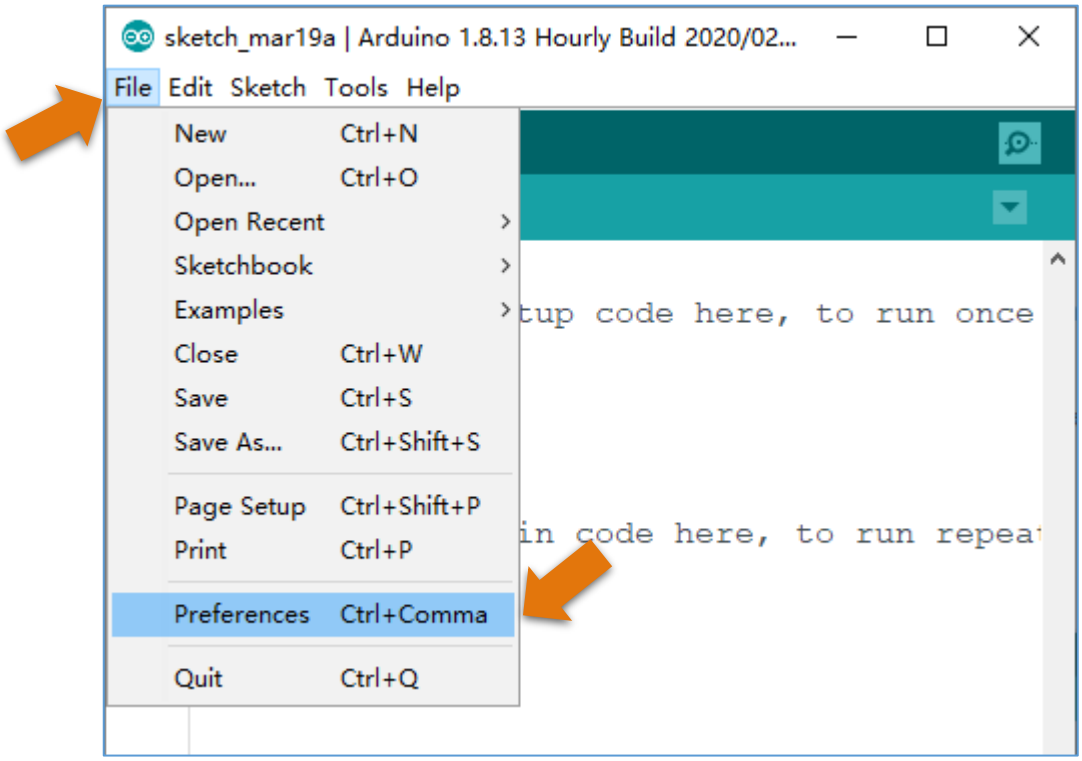
Programs written with Arduino Software (IDE) are called **sketches**. These sketches are written in the text editor and saved with the file extension **.ino**. The editor has features for cutting/pasting and searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino Software (IDE), including complete error messages and other information. The bottom right-hand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.

	Verify	Check your code for compile errors .
	Upload	Compile your code and upload them to the configured board.
	New	Create a new sketch.
	Open	Present a menu of all the sketches in your sketchbook. Clicking one will open it within the current window and overwrite its content.
	Save	Save your sketch.
	Serial Monitor	Open the serial monitor.

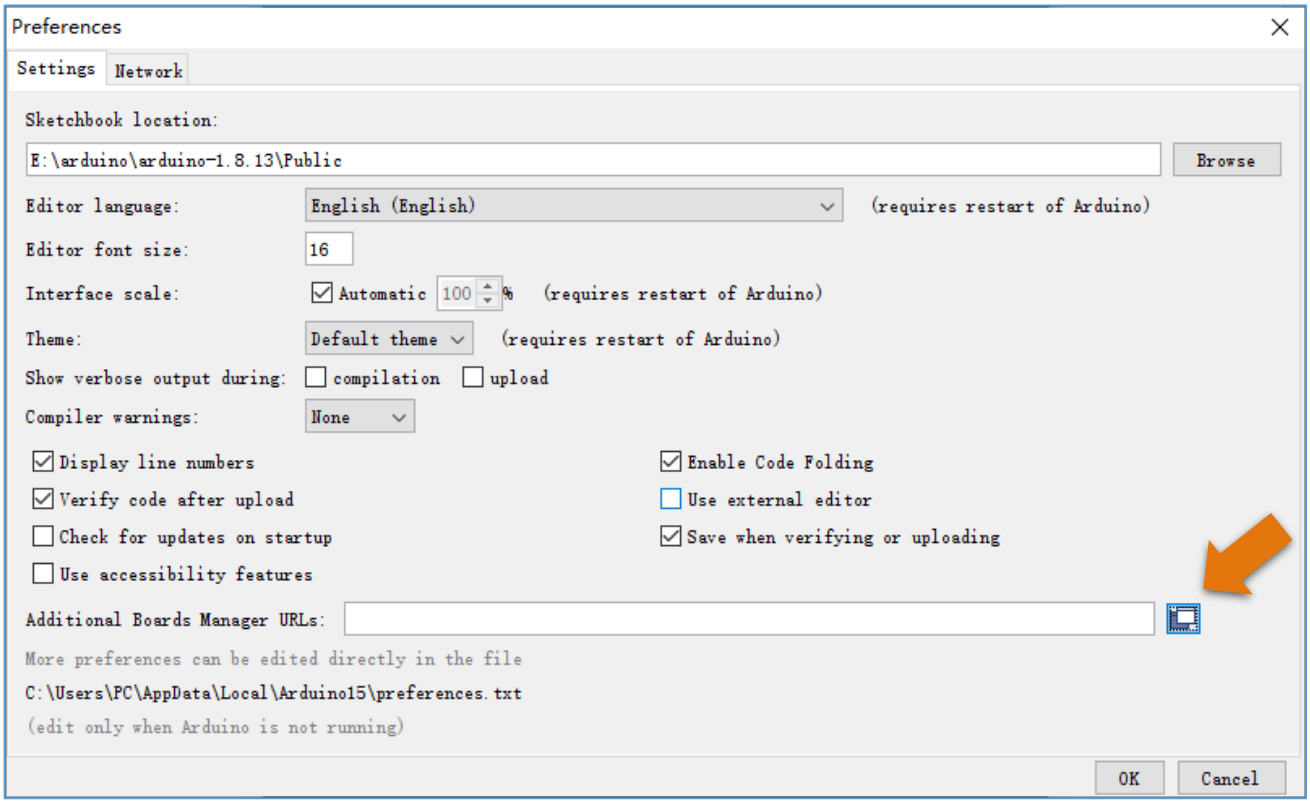
Additional commands are found within the five menus: File, Edit, Sketch, Tools, Help. The menus are context sensitive, which means only those items relevant to the work currently being carried out are available.

Environment Configuration

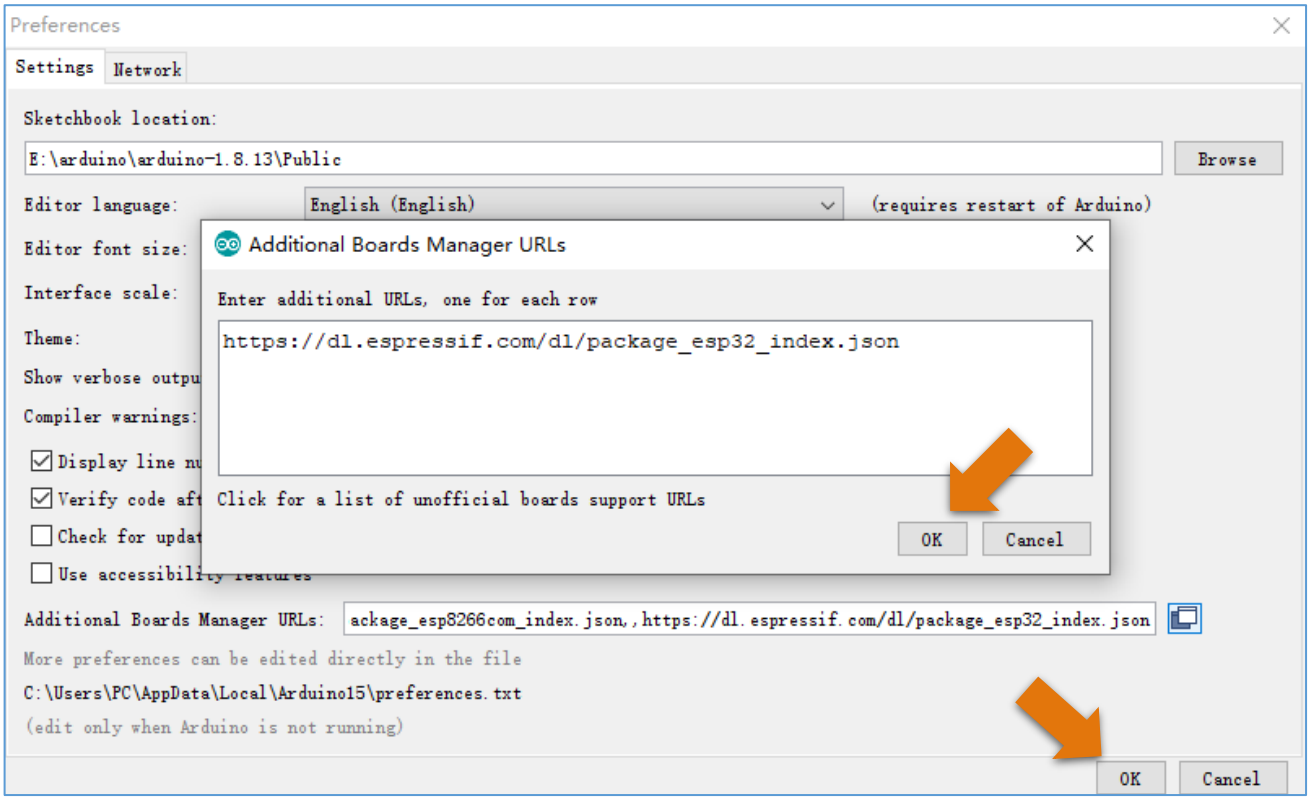
First, open the software platform arduino, and then click File in Menus and select Preferences.



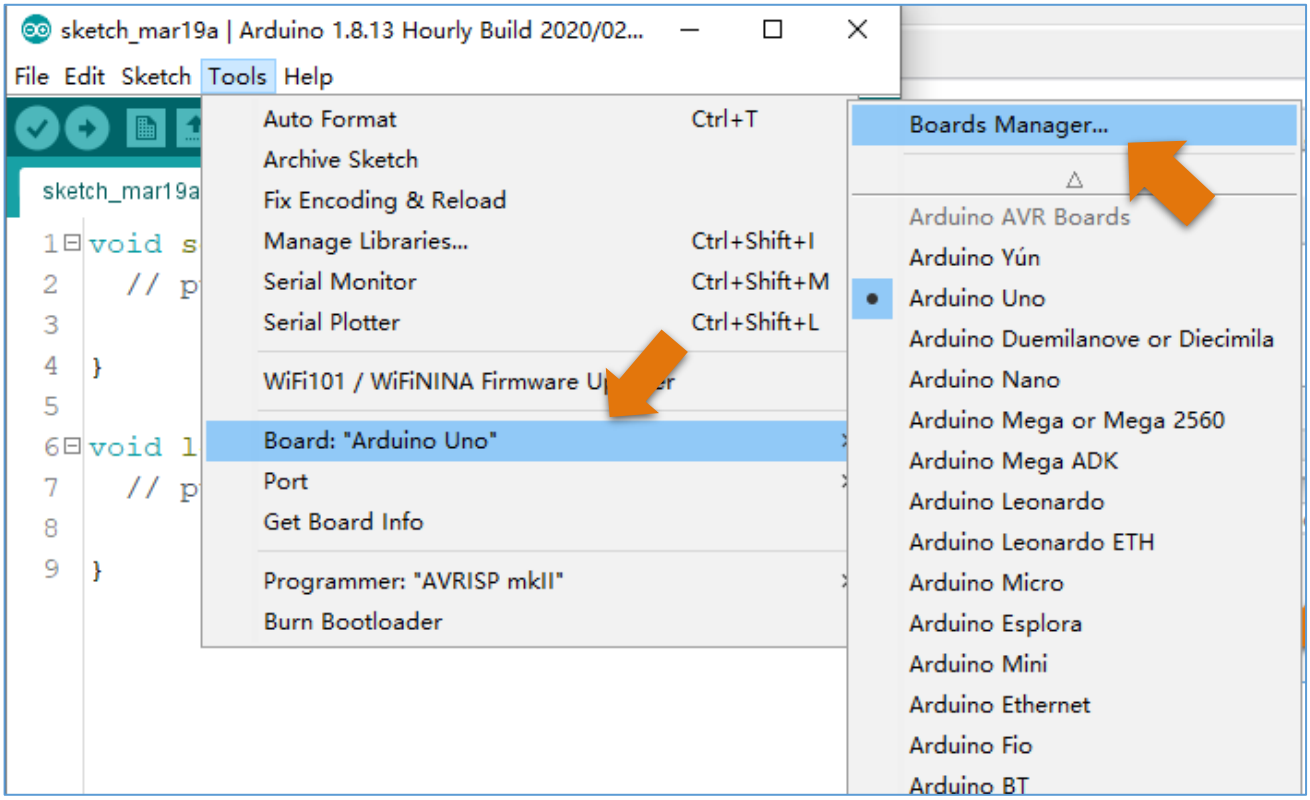
Second, click on the symbol behind "Additional Boards Manager URLs"



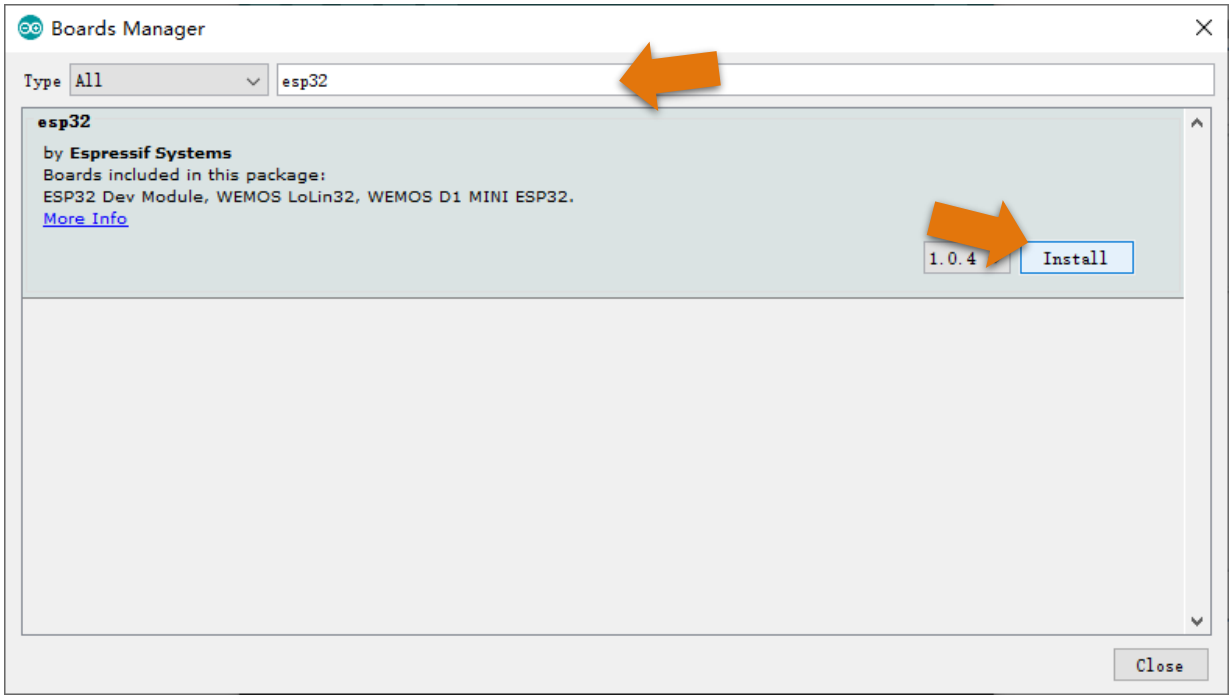
Third, fill in https://dl.espressif.com/dl/package_esp32_index.json in the new window, click OK, and click OK on the Preferences window again.



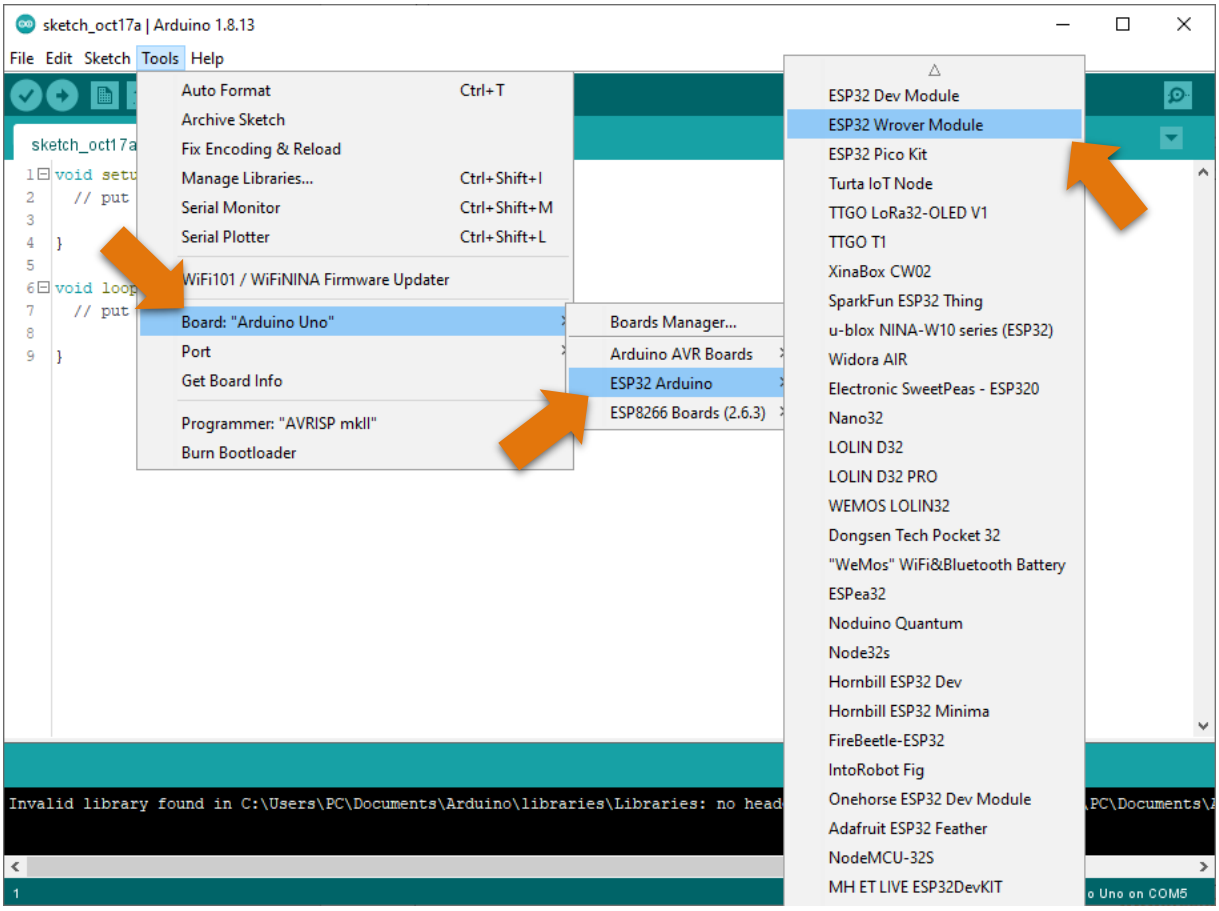
Fourth, click Tools in Menus, select Board:"ArduinoUno", and then select "Boards Manager".



Fifth, input "esp32" in the window below, and press Enter. click "Install" to install.



When finishing installation, click Tools in the Menus again and select Board: "Arduino Uno", and then you can see information of ESP32-WROVER. click "ESP32-WROVER" so that the ESP32 programming development environment is configured.



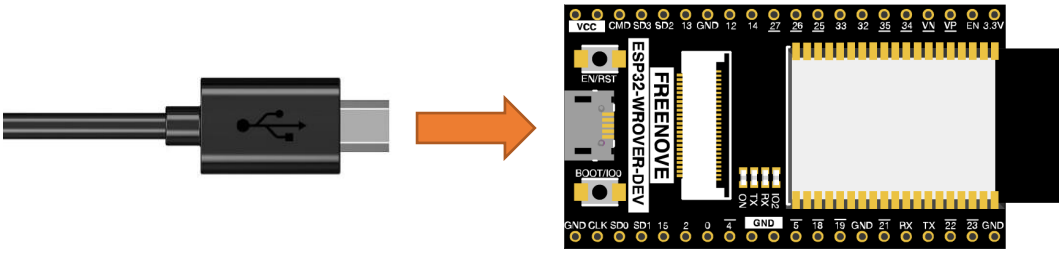
CH340

ESP32 uses CH340 to download codes. So before using it, we need to install CH340 driver in our computers.

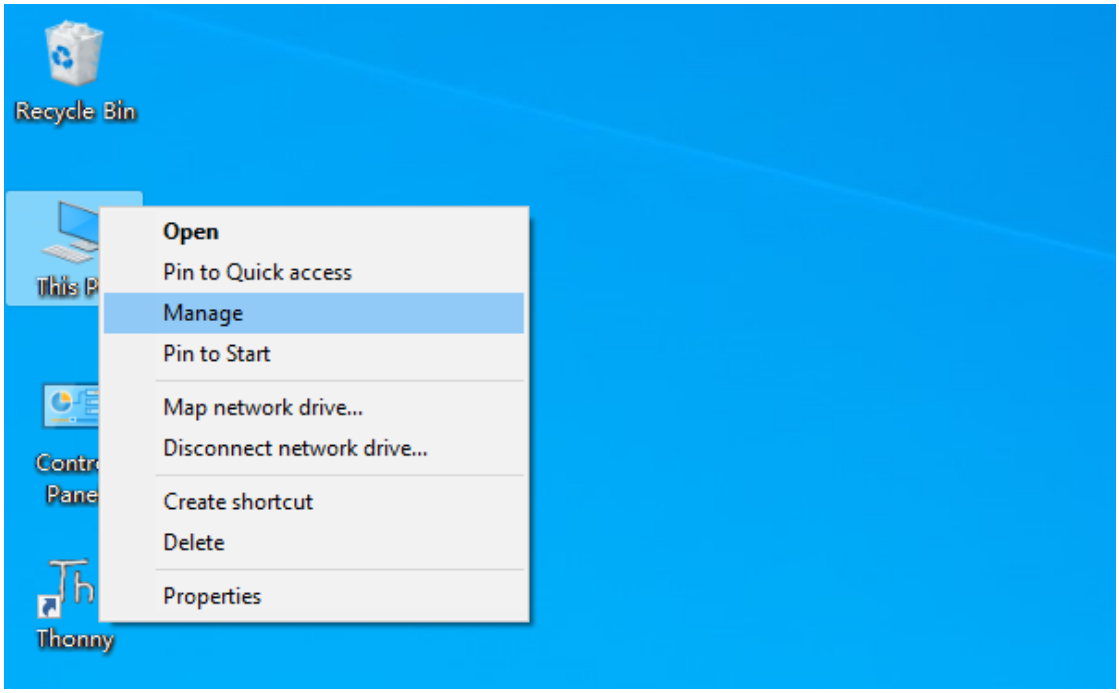
Windows

Check whether CH340 has been installed

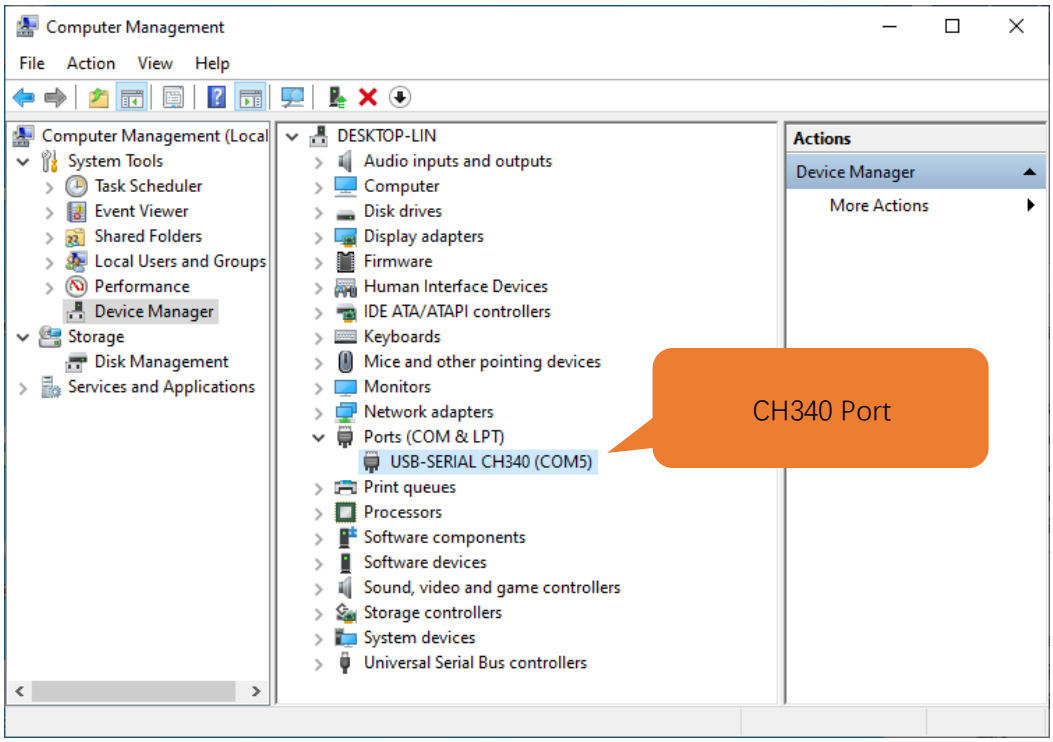
1. Connect your computer and ESP32 with a USB cable.



2. Turn to the main interface of your computer, select "This PC" and right-click to select "Manage".



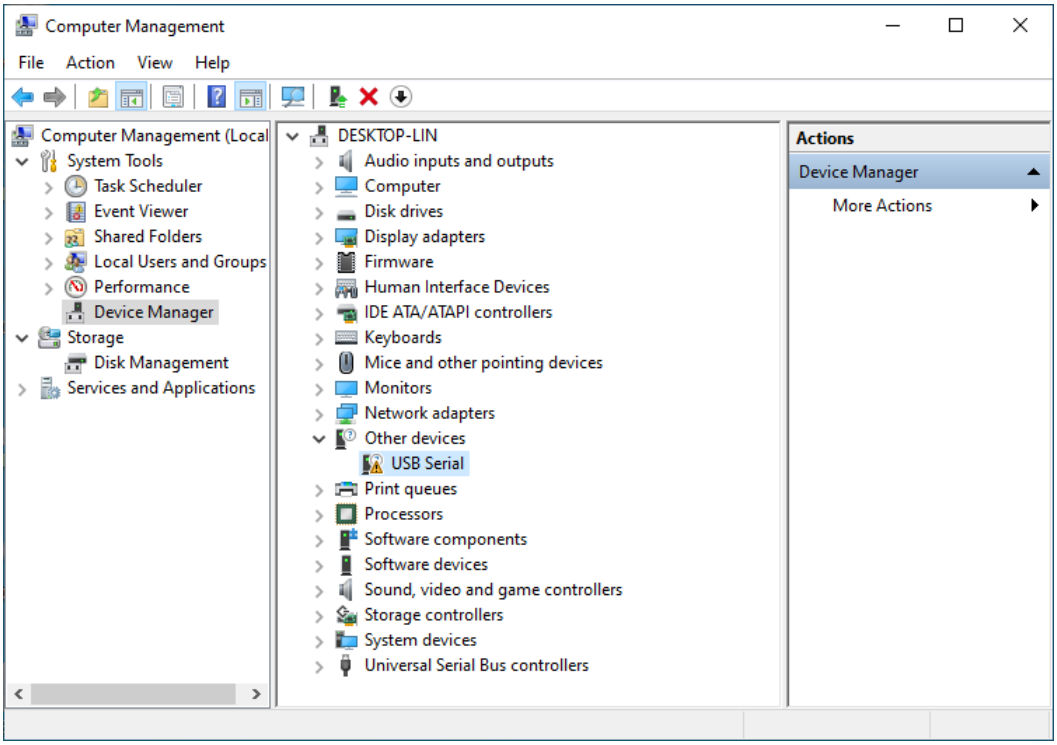
3. Click "Device Manager". If your computer has installed CH340, you can see"USB-SERIAL CH340 (COMx)". And you can click here to move to the next step.



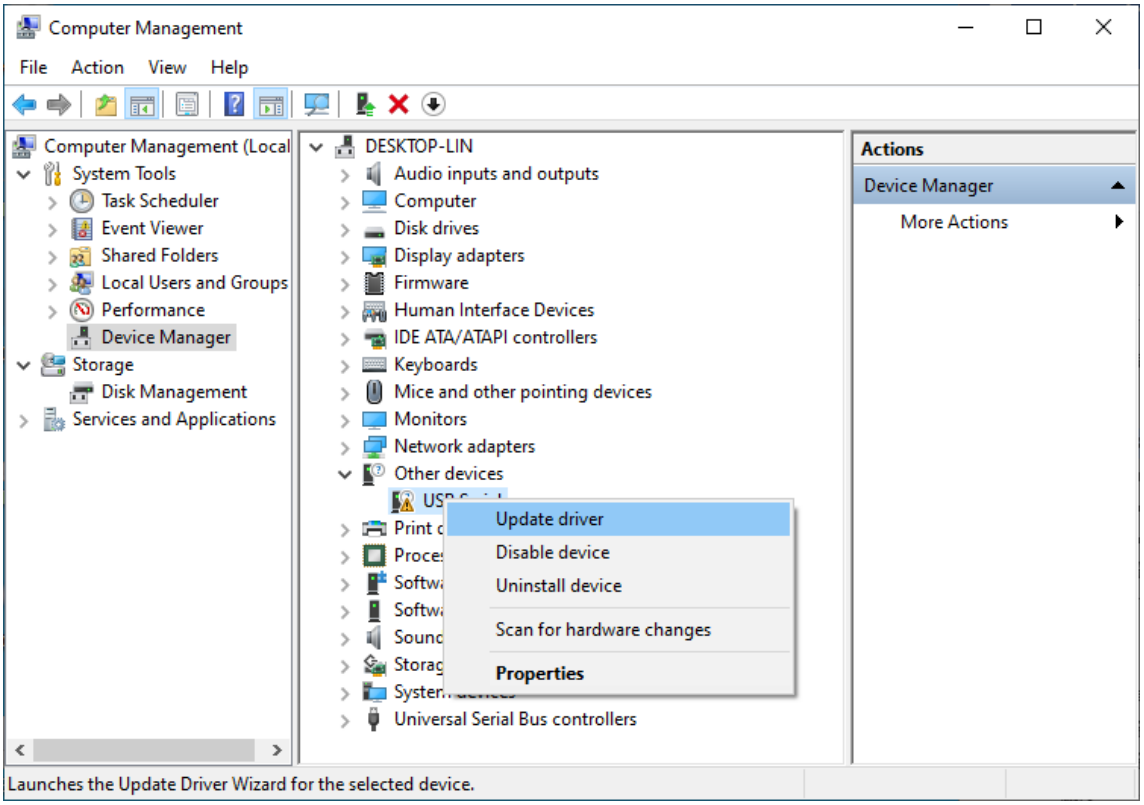
Installing CH340

Method1

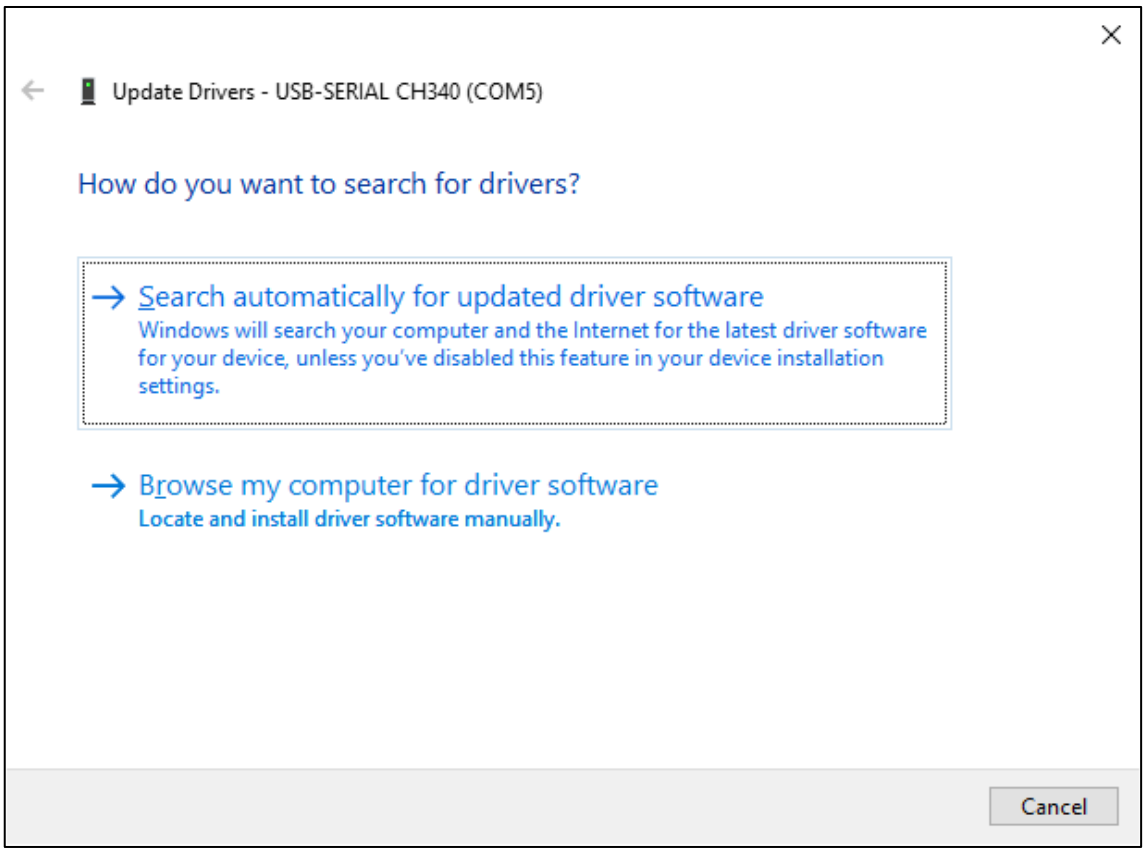
1. If you have not yet installed CH340, you will see the following interface.



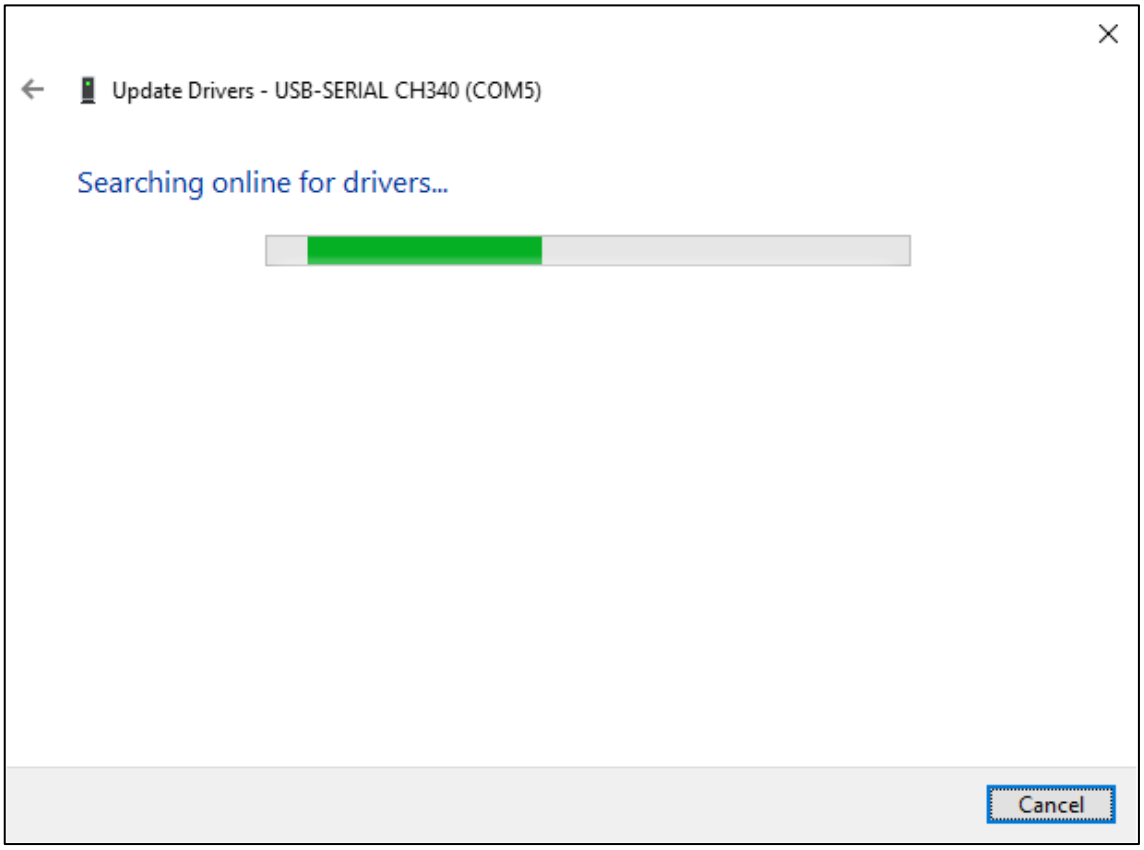
2. Click “USB Serial” and right-click to select “Update driver”.



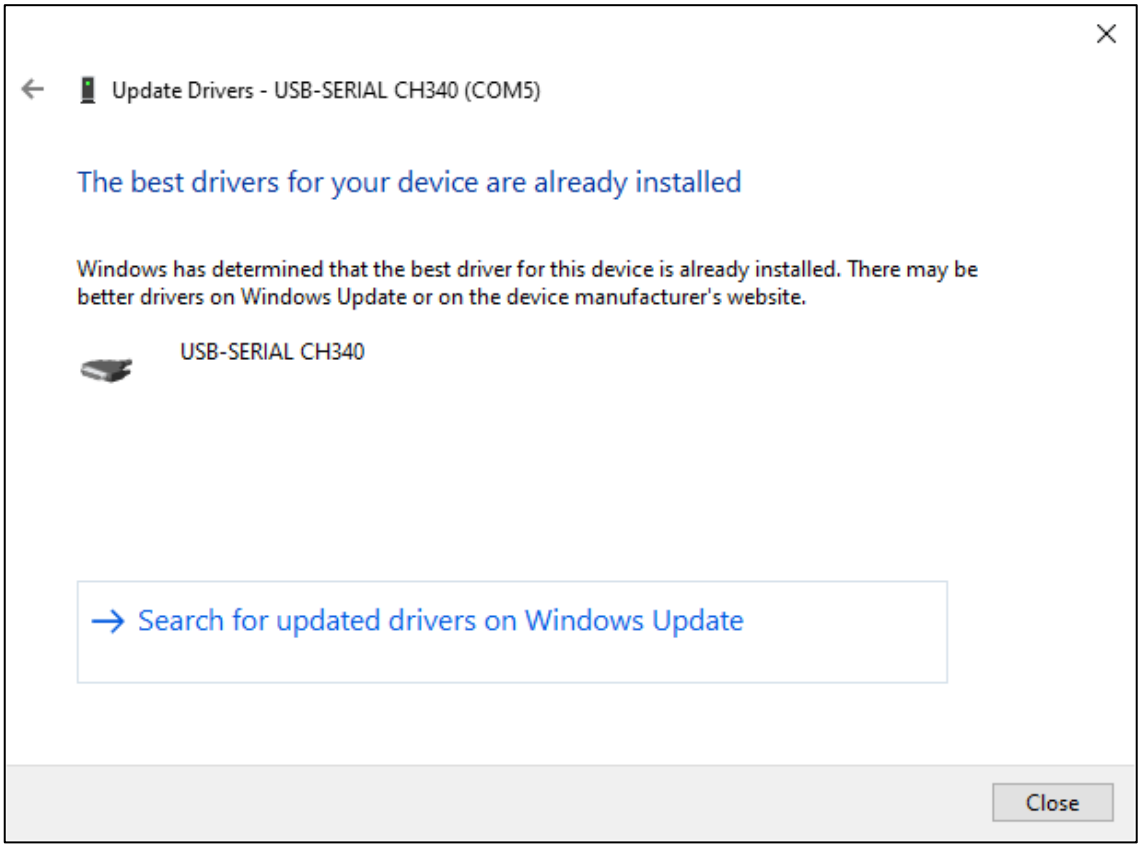
3. Click “Search automatically for updated driver software”.



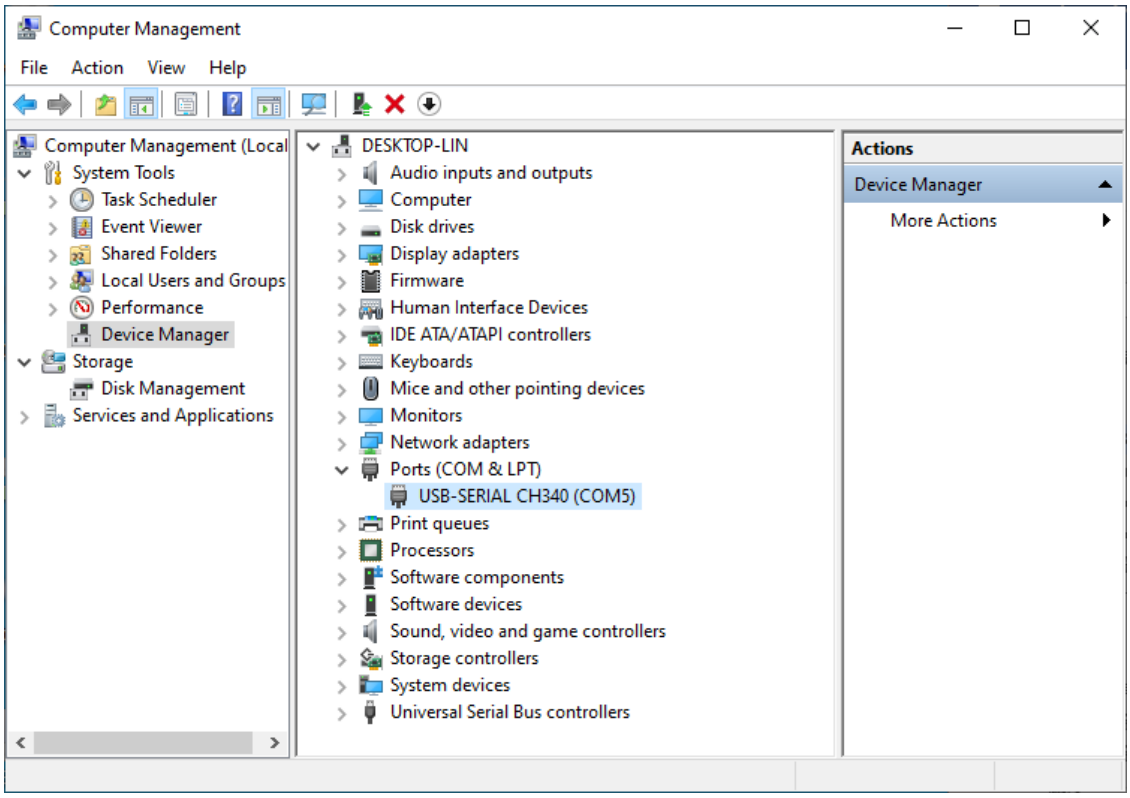
4. Wait for CH340 to finish installation.



5. When you see the following interface, it indicates that CH340 has been installed to your computer. You can close the interface.

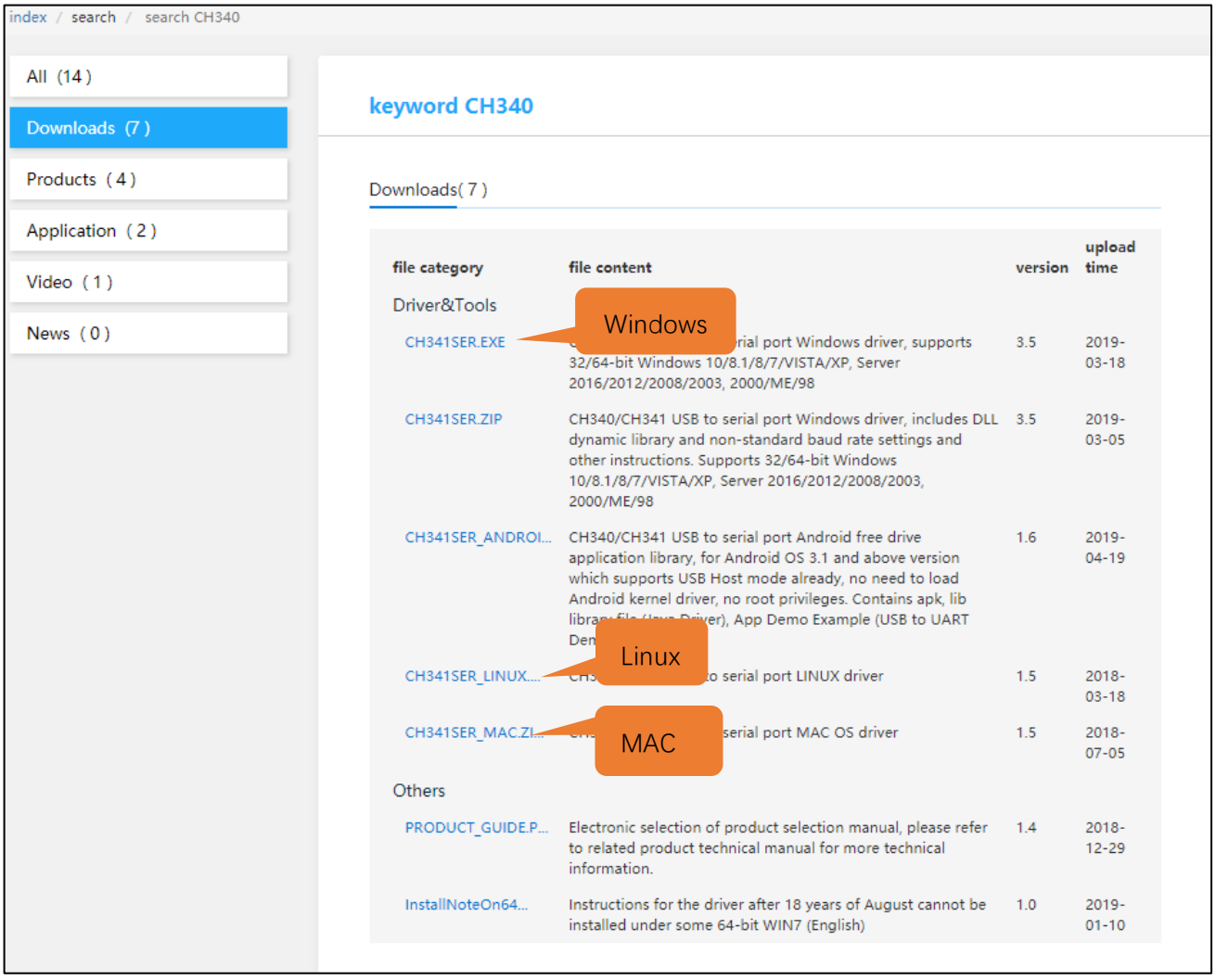


6. When ESP32 is connected to computer, you can see the following interface. Click here to move to the next step.

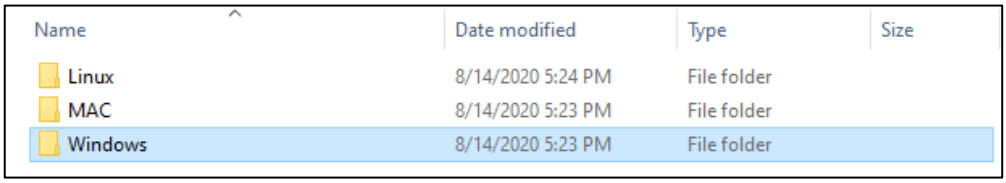


Method2

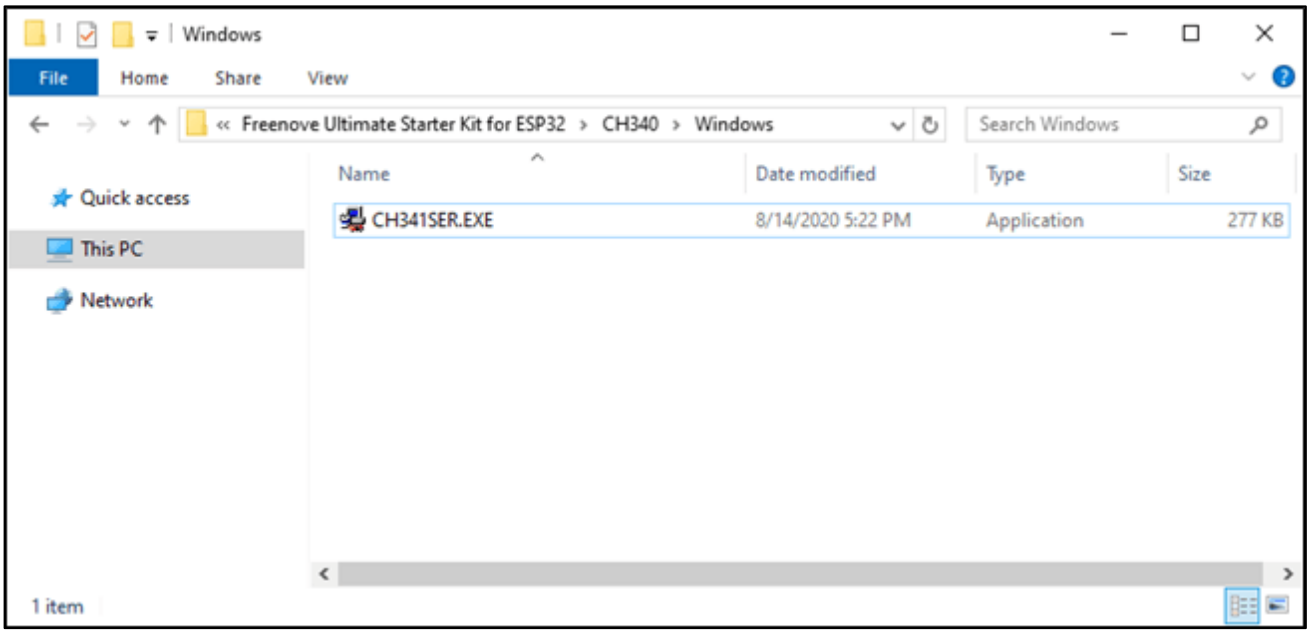
1. First, download CH340 driver, click http://www.wch-ic.com/search?q=CH340&t=downloads to download the appropriate one based on your operating system.



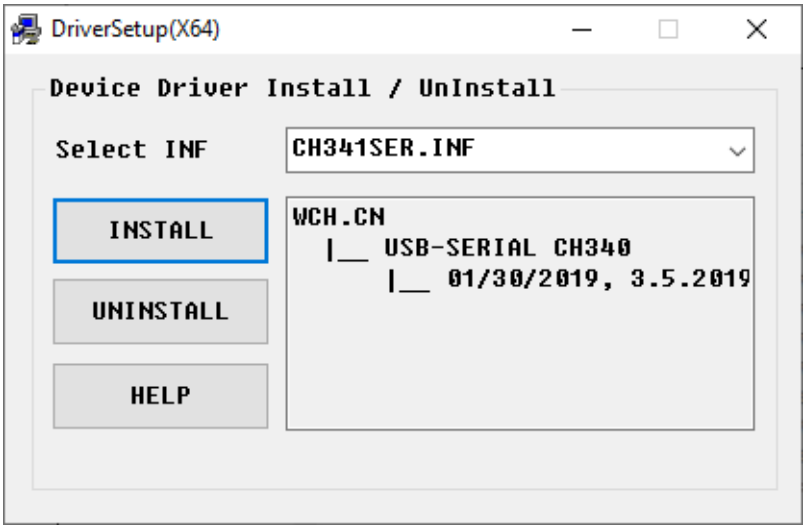
If you would not like to download the installation package, you can open **"Freenove_4WD_Car_Kit_for_ESP32/CH340"**, we have prepared the installation package.



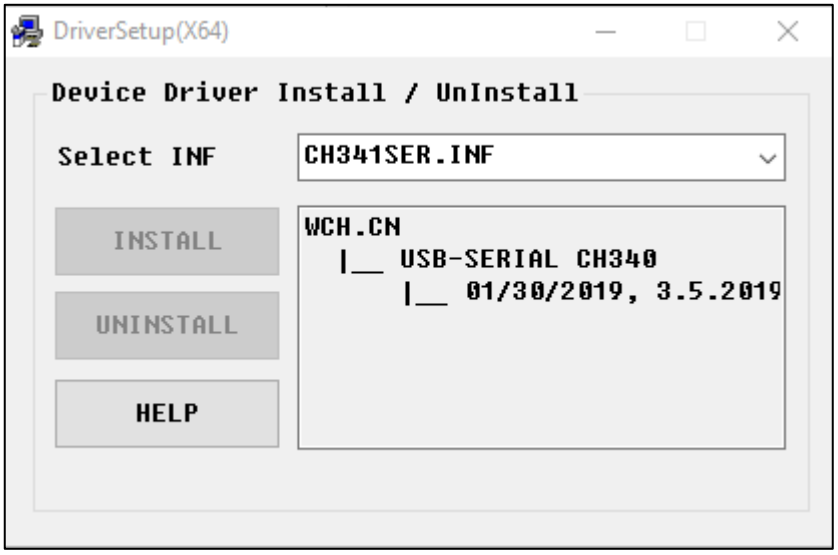
2. Open the folder "**Freenove_4WD_Car_Kit_for_ESP32/CH340/Windows/**"



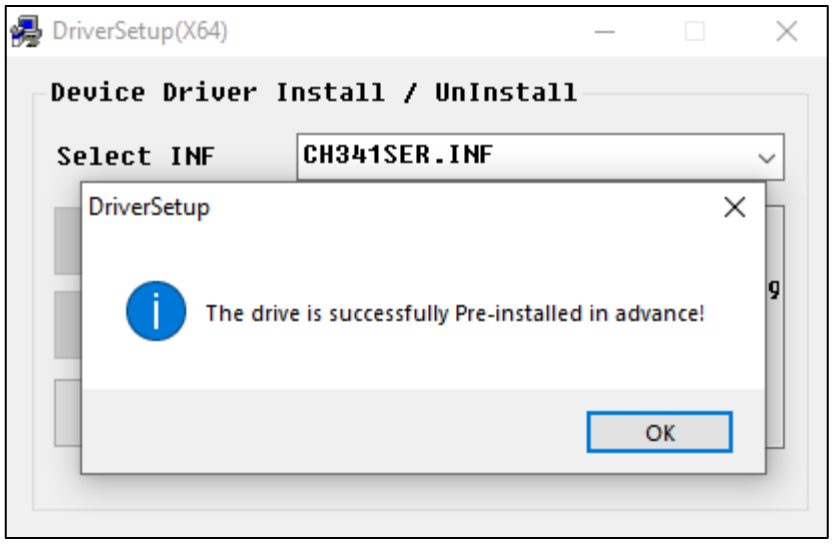
3. Double click "**CH341SER.EXE**".



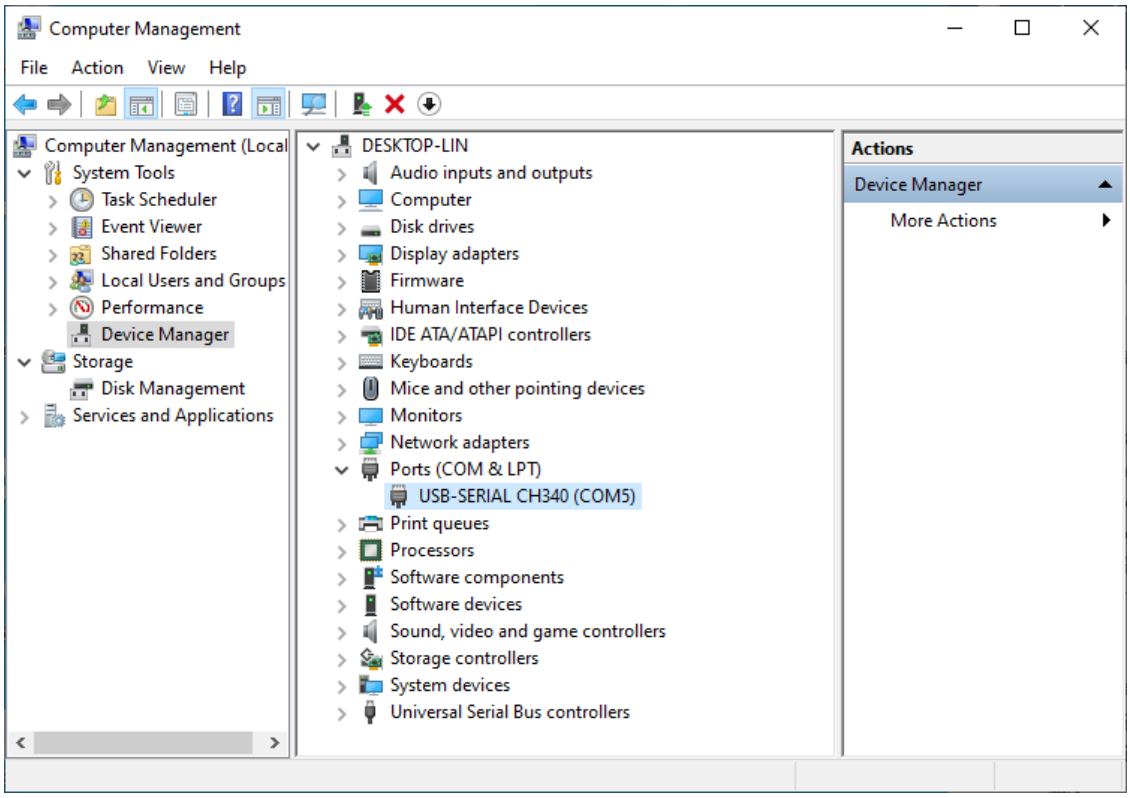
4. Click "INSTALL" and wait for the installation to complete.



5. Install successfully. Close all interfaces.



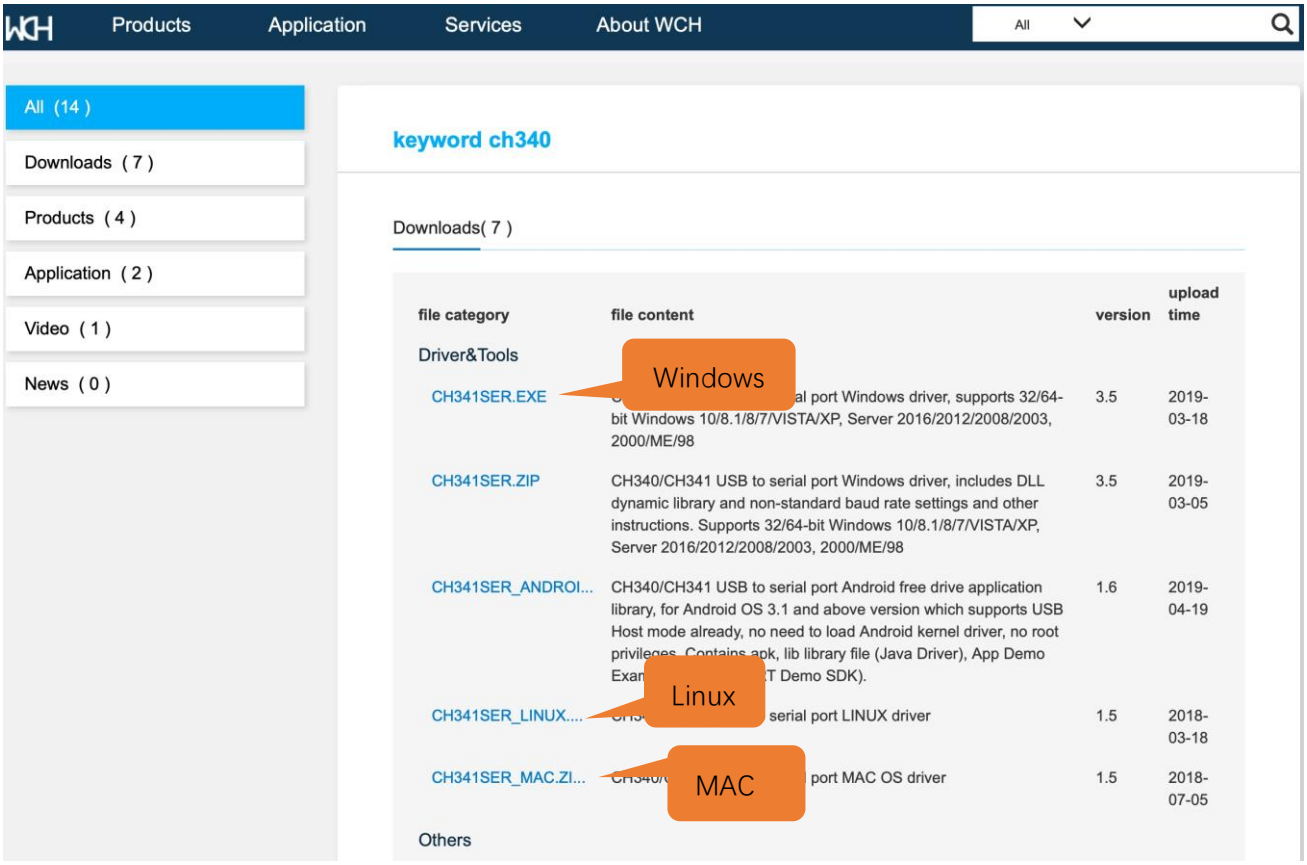
6. When ESP32 is connected to computer, select “This PC”, right-click to select “Manage” and click “Device Manager” in the newly pop-up dialog box, and you can see the following interface.



7. So far, CH340 has been installed successfully. Close all dialog boxes.

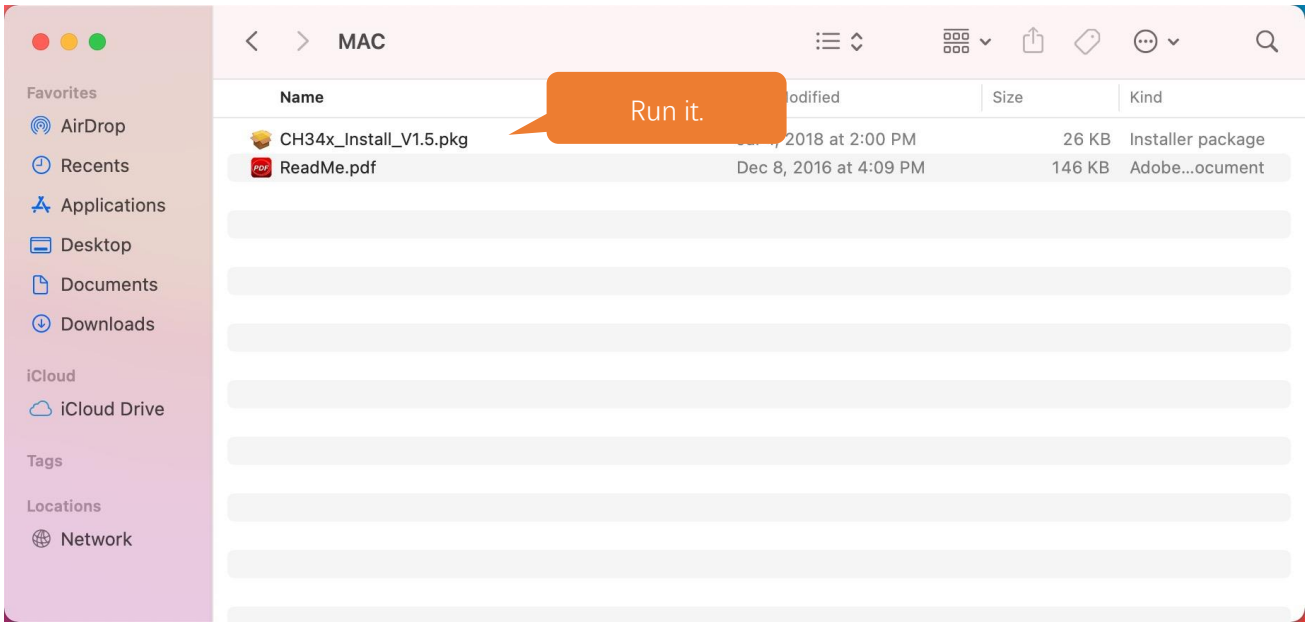
MAC

First, download CH340 driver, click http://www.wch-ic.com/search?q=CH340&t=downloads to download the appropriate one based on your operating system.

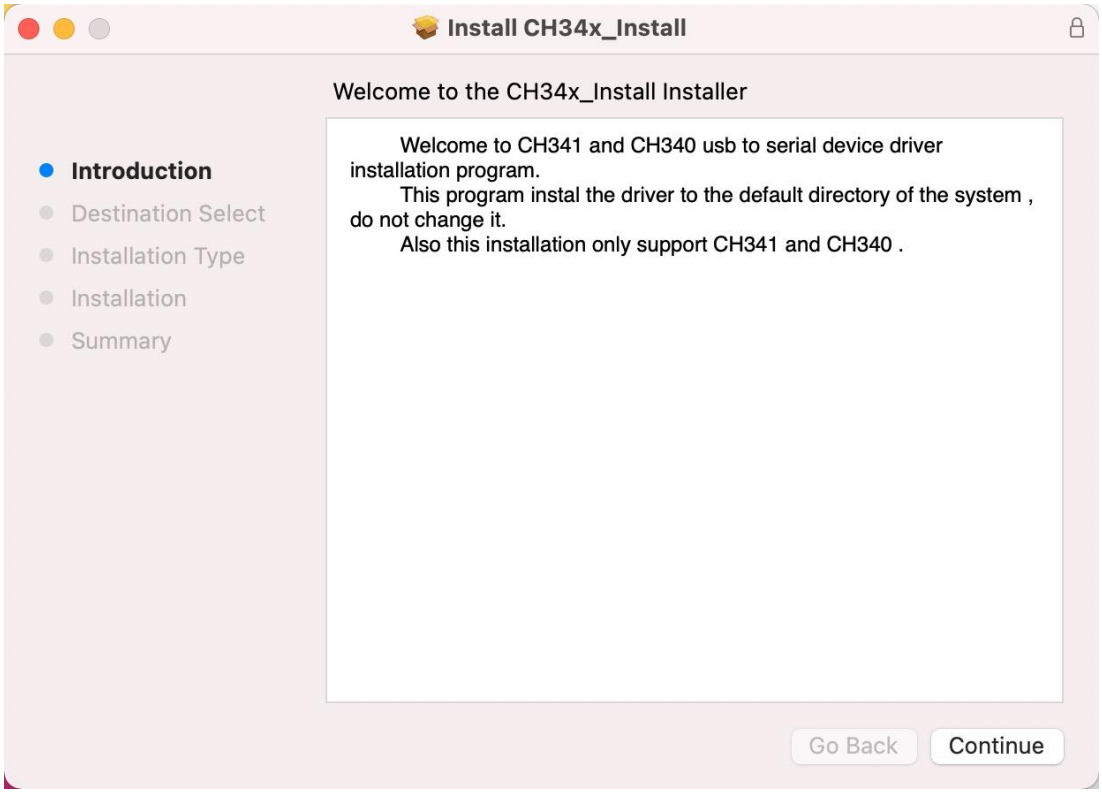


If you would not like to download the installation package, you can open **"Freenove_4WD_Car_Kit_for_ESP32/CH340"**, we have prepared the installation package.

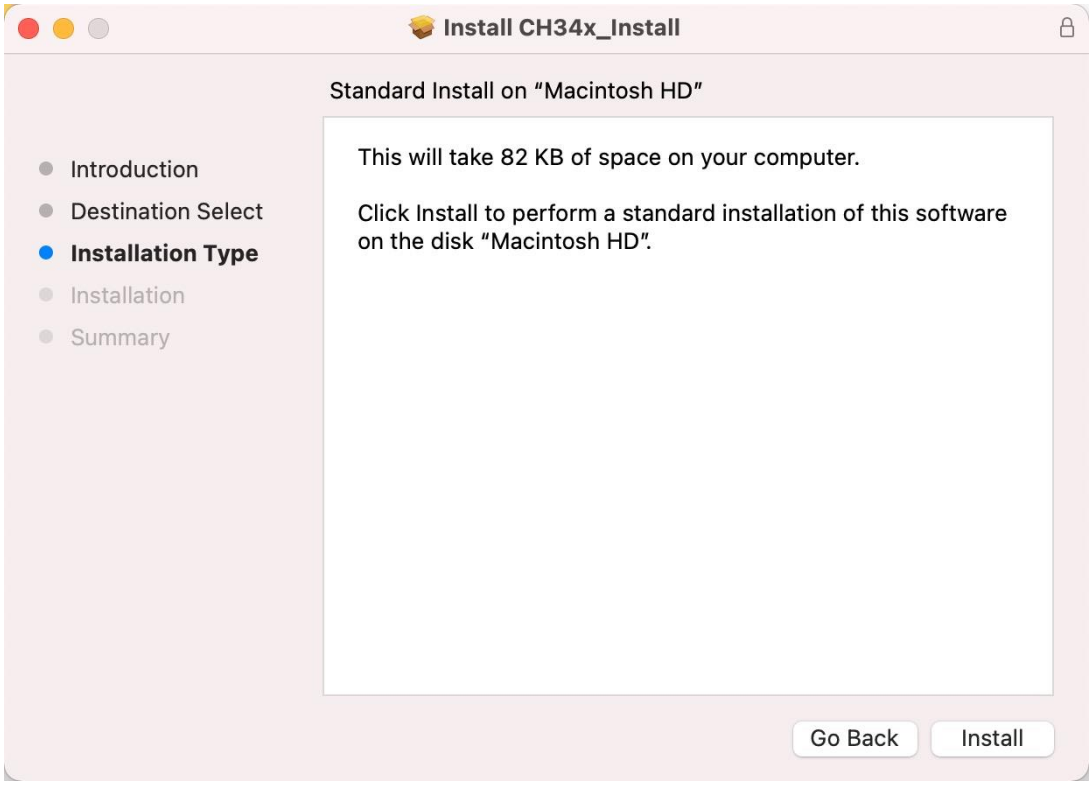
Second, open the folder "**Freenove_4WD_Car_Kit_for_ESP32/CH340/MAC/**"



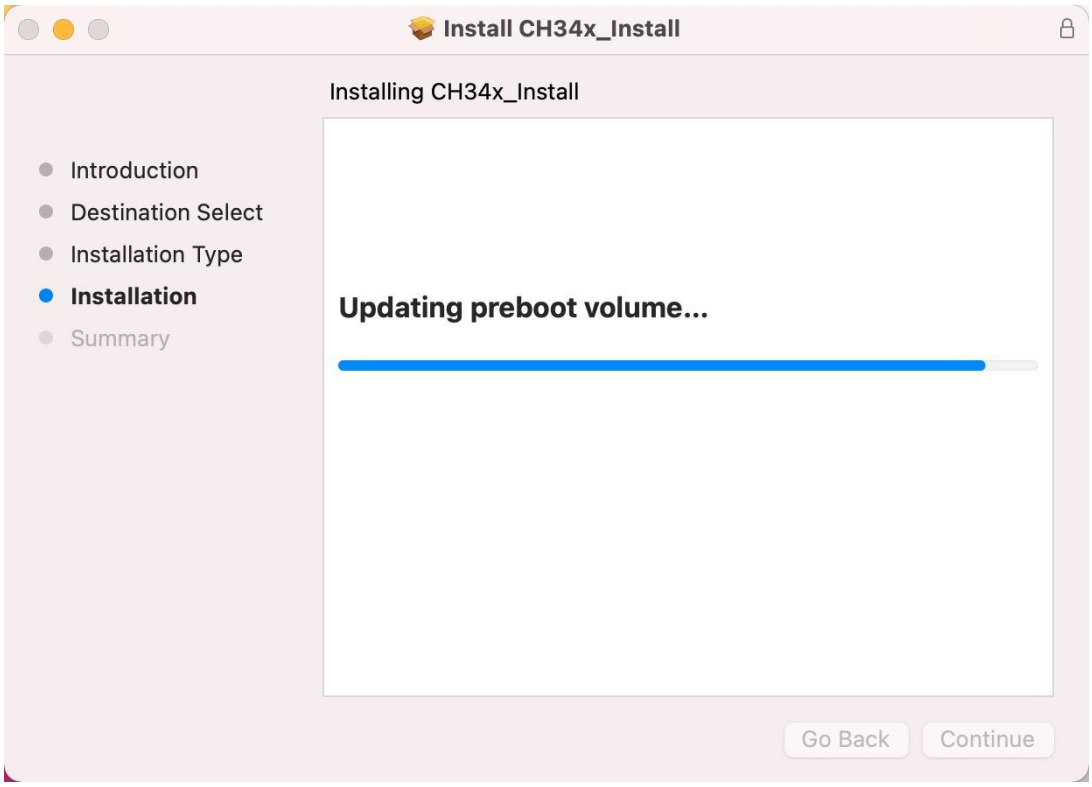
Third, click Continue.



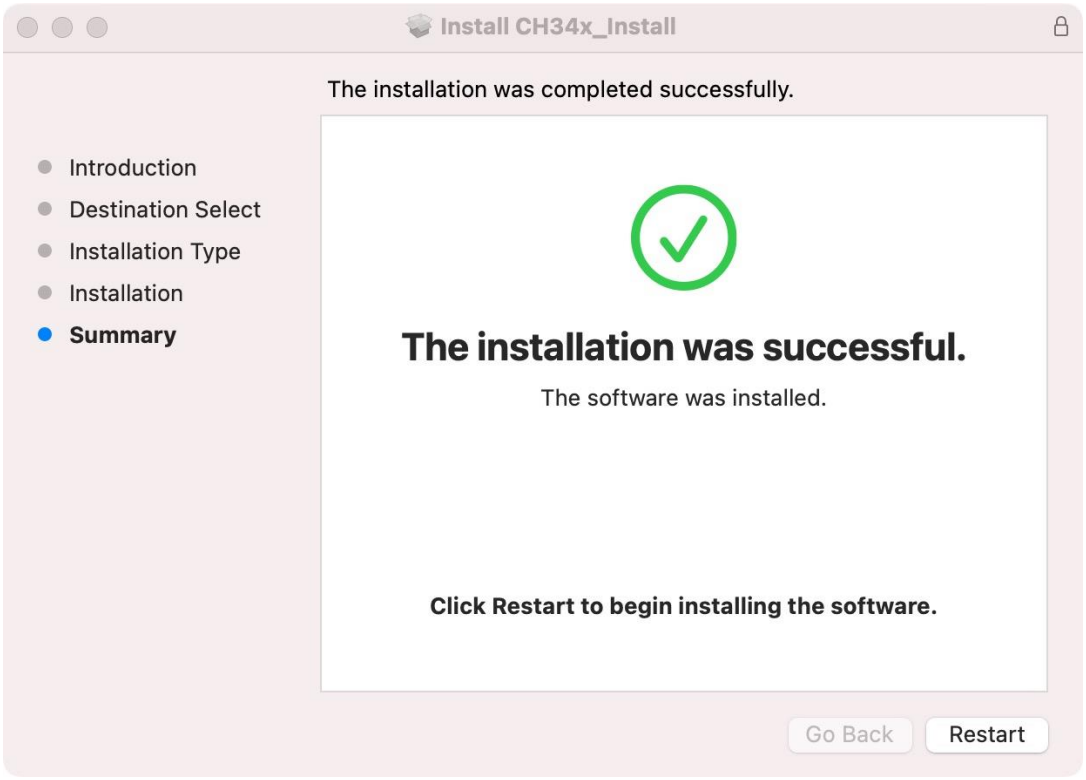
Fourth, click Install.



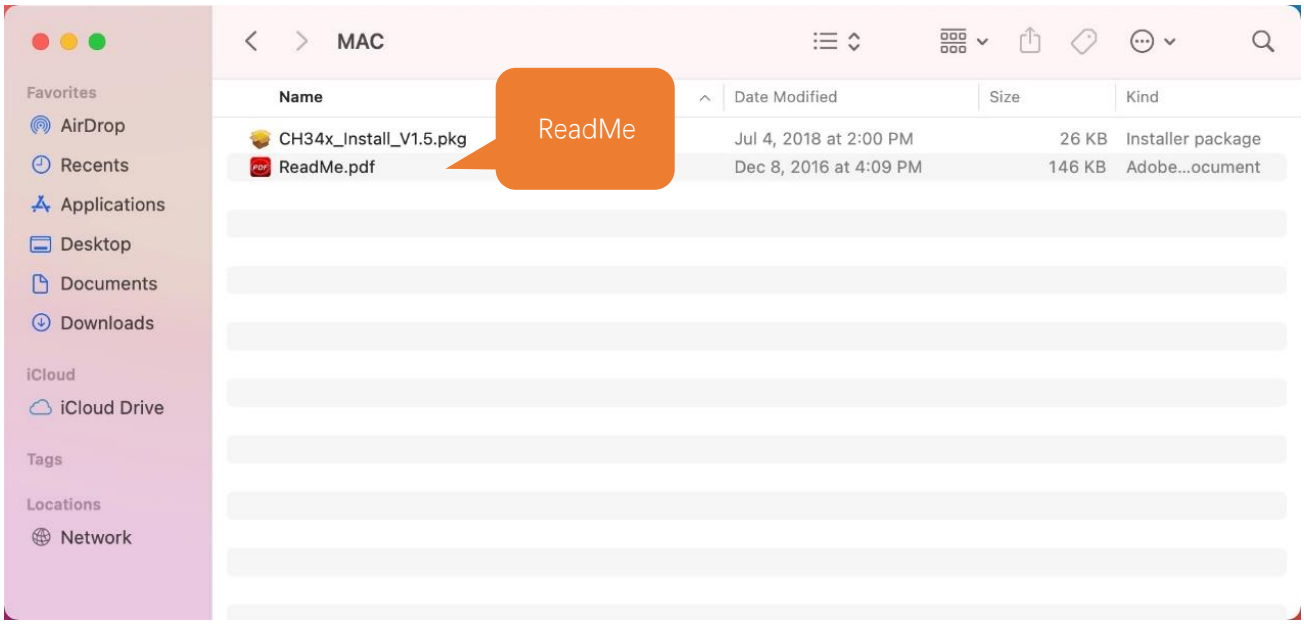
Then, waiting Finsh.



Finally, restart your PC.



If you still have not installed the CH340 by following the steps above, you can view readme.pdf to install it.



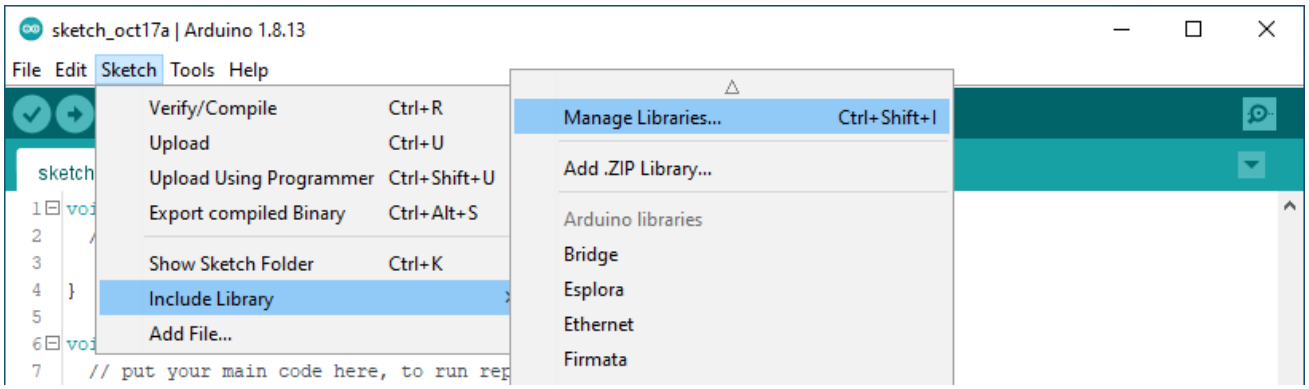
Uploading the First Code

Here we use "00.0_Servo_90" in **"Freenove_4WD_Car_Kit_for_ESP32\Sketches"** as an example.
The servo on the car is controlled by PCA9685. Therefore, it is necessary to add the related libararies to Arduino IDE.

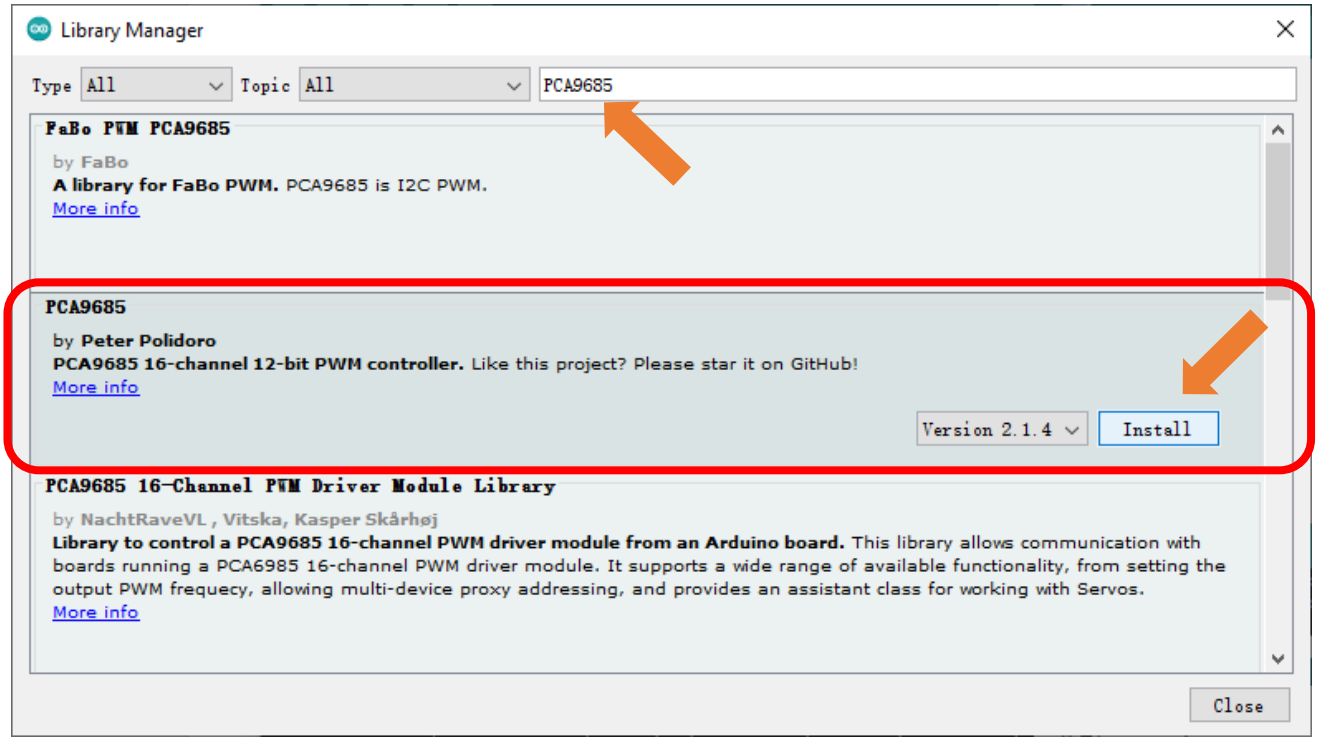
How to Add libraries

Method 1

Open Arduino IDE, click Sketch on Menu bar, move your mouse to Include Library and then click Manage Libraries.



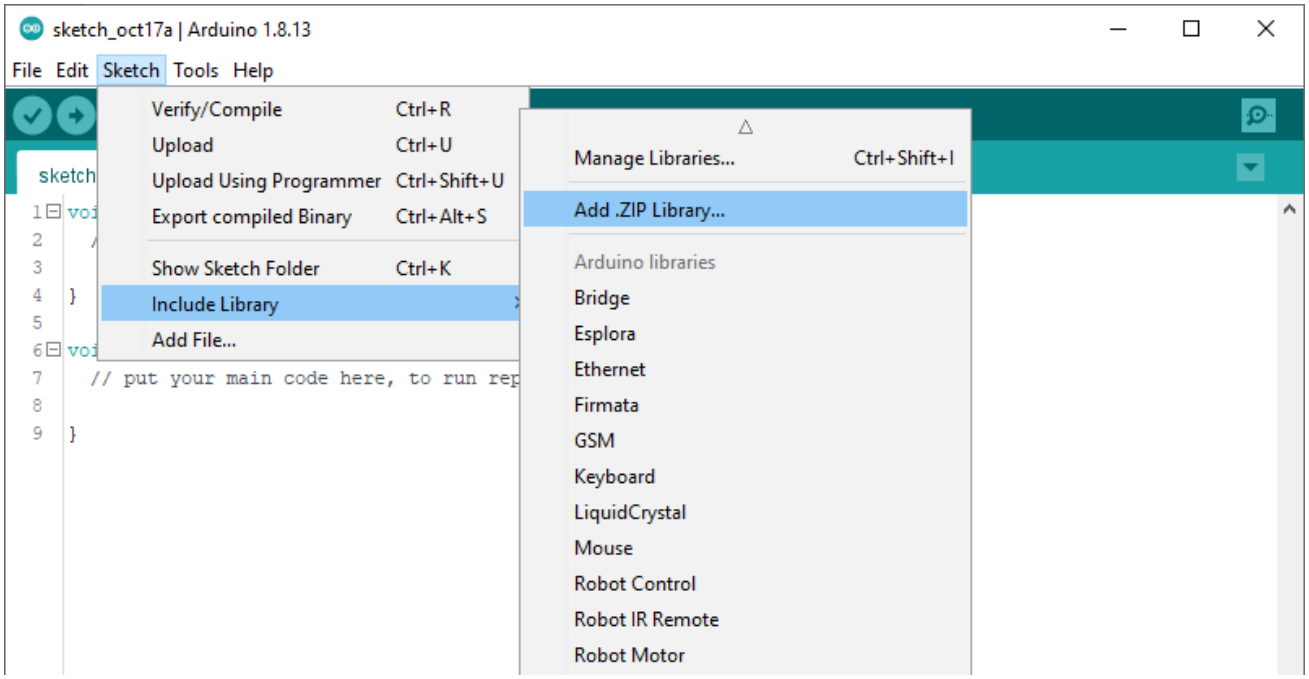
There is an input field on the right top of the pop-up window. Enter PCA9685 there and click to install the library boxed in the following picture, click to



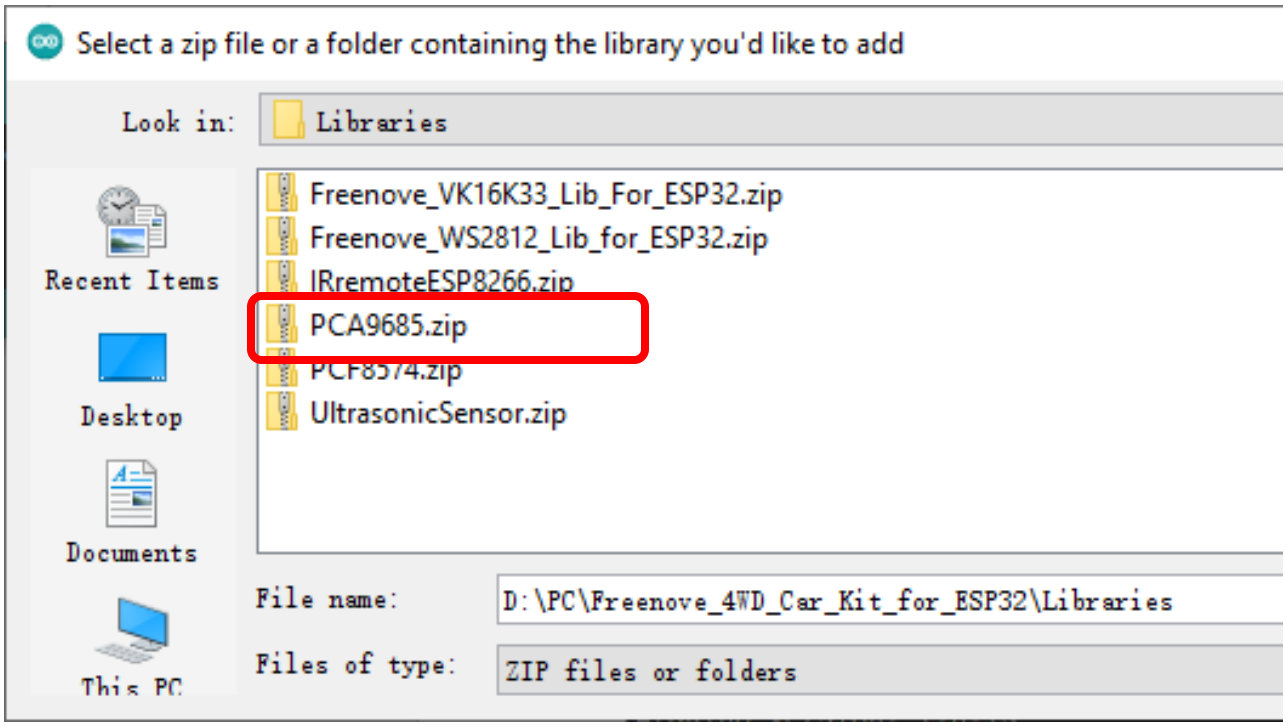
Wait for the installation to finish.

Method 2

Open Arduino IDE, click Sketch on Menu bar, move your mouse to Include Library and then click Add .ZIP library.

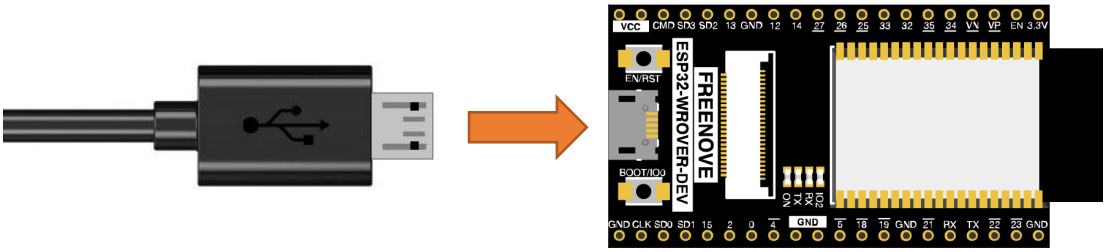


On the pop-up window, select PCA9685.zip of Libraries folder in **"Freenove_4WD_Car_Kit_for_ESP32\Sketches"**, and then click Open.

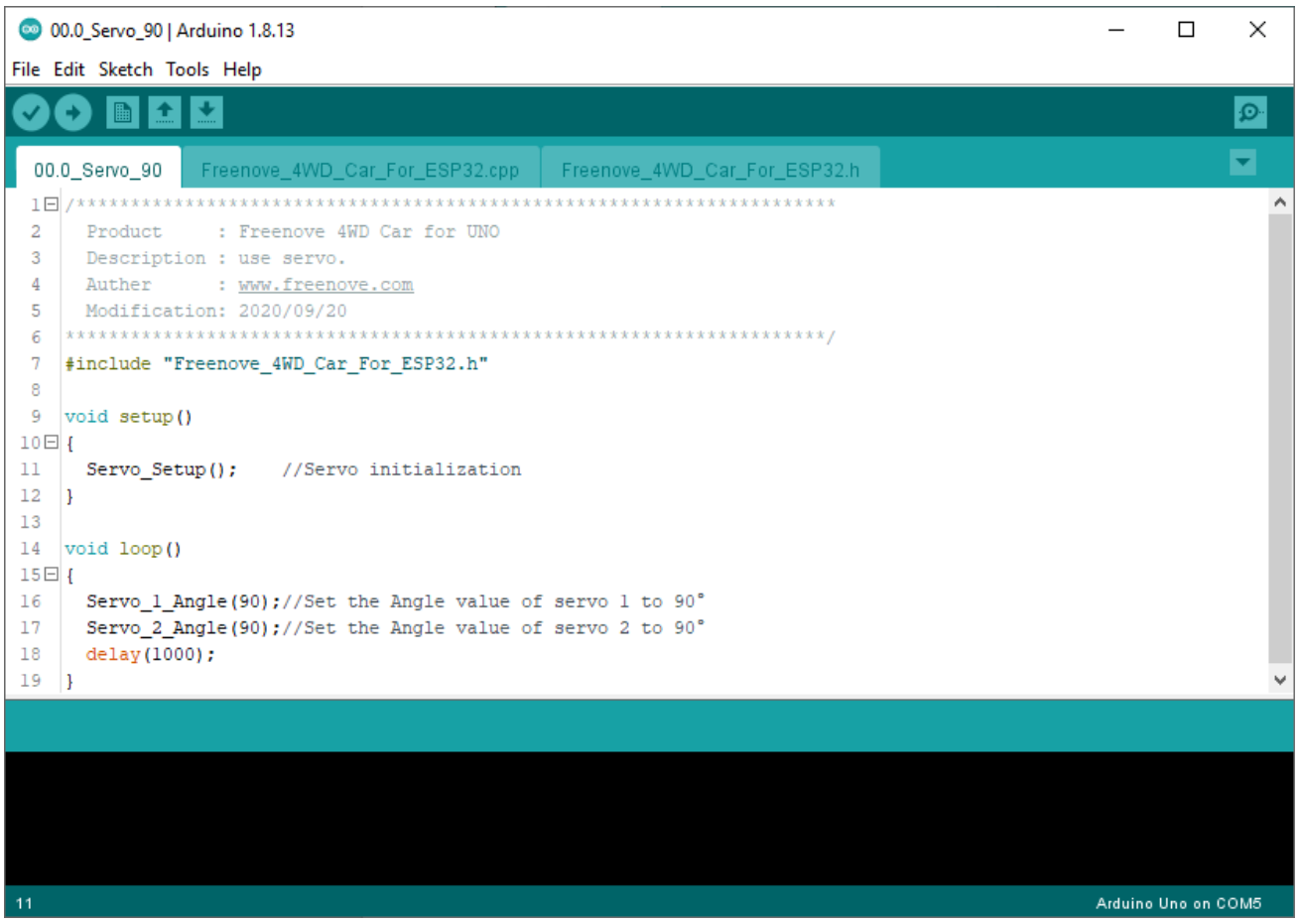


How to compile and upload code

Step 1. Connect your computer and ESP32 with a USB cable.

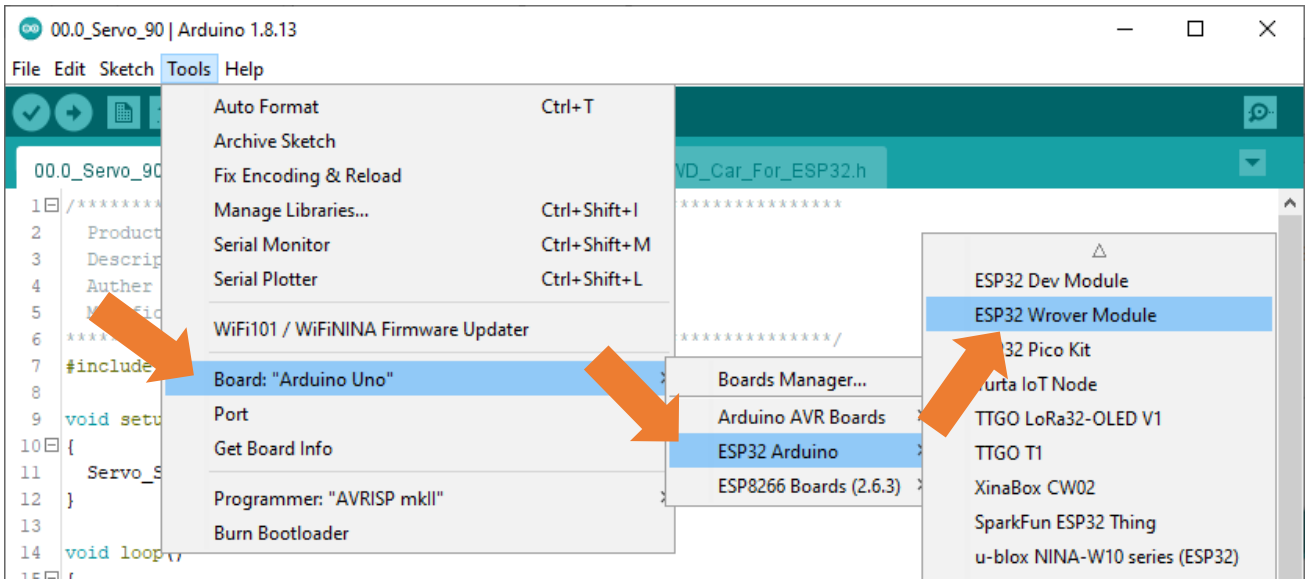


Step 2. Open "00.0_Servo_90" folder in **"Freenove_4WD_Car_Kit_for_ESP32\Sketches"**, double-click "00.0_Servo_90.ino". The code is to rotate the two servo motors to 90°.



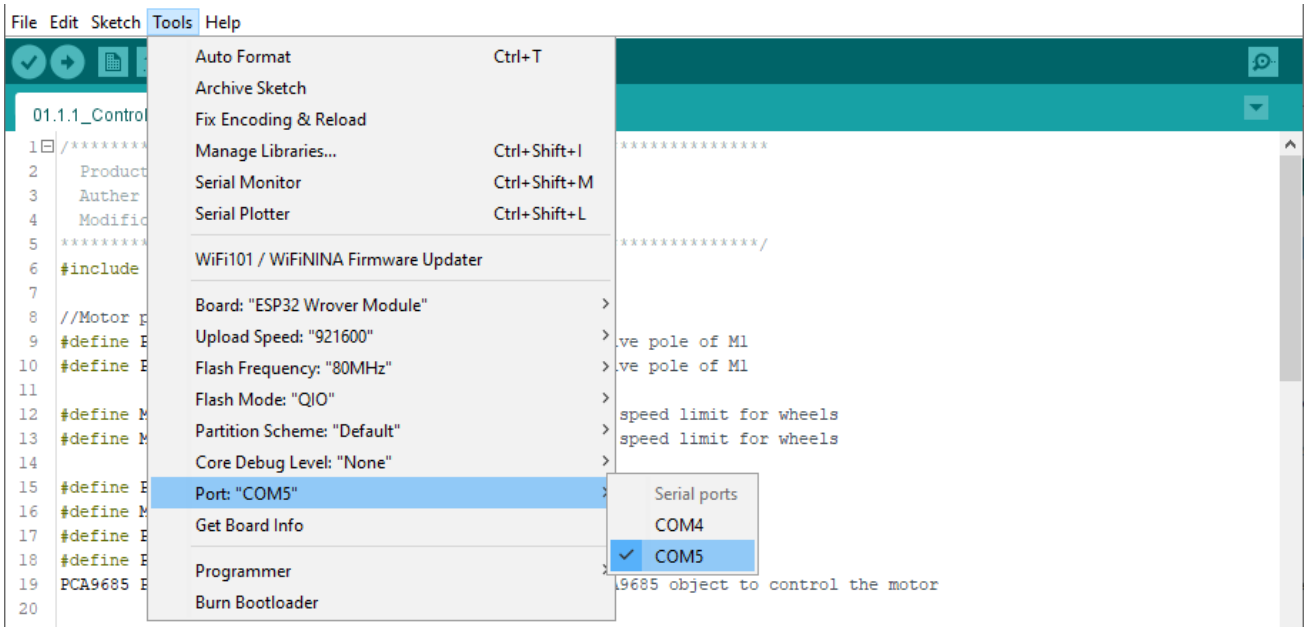
Step 3. Select development board.

Click Tolls on Menu bar, move your mouse to Board: “Arduino Uno”, select ESP32 Arduino and then select ESP32 Wrover Module.



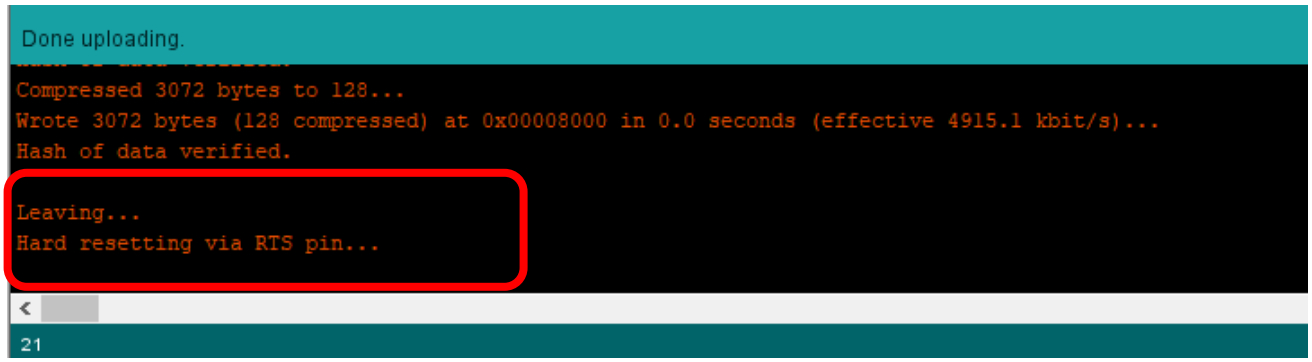
Step 4. Select serial port.

Cilick Tools on Menu bar, move your mouse to Port and select COMx on your computer. The value of COMx varies in different computers, but it won’t affect the download function of ESP32, as long as you select the correct one.

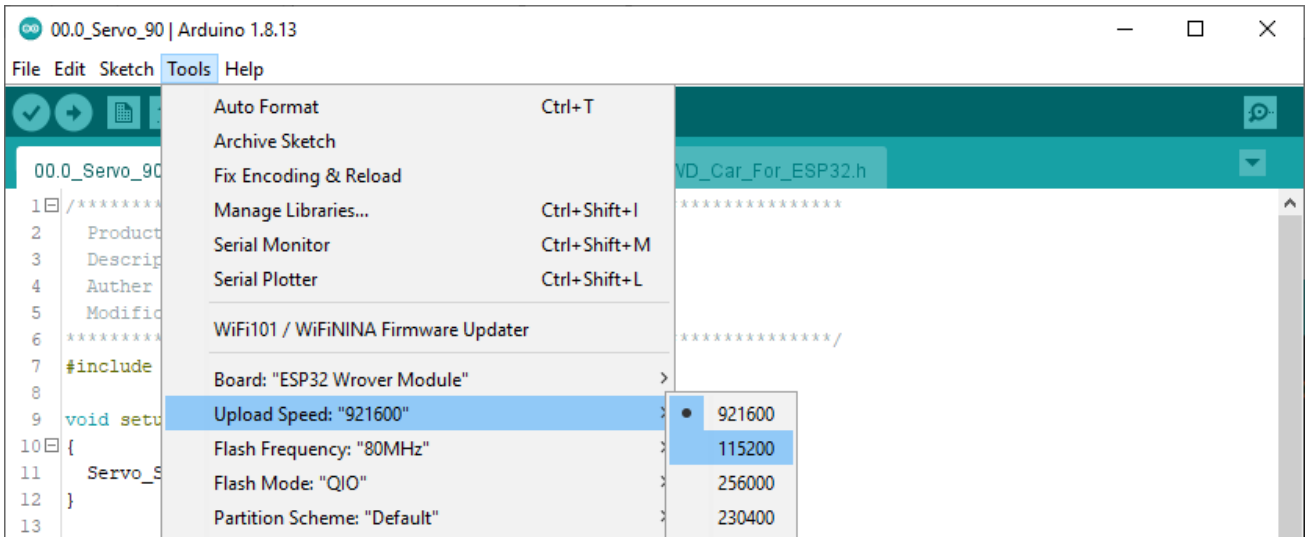


Click "Upload Using Programmer" and the program will be downloaded to ESP32.

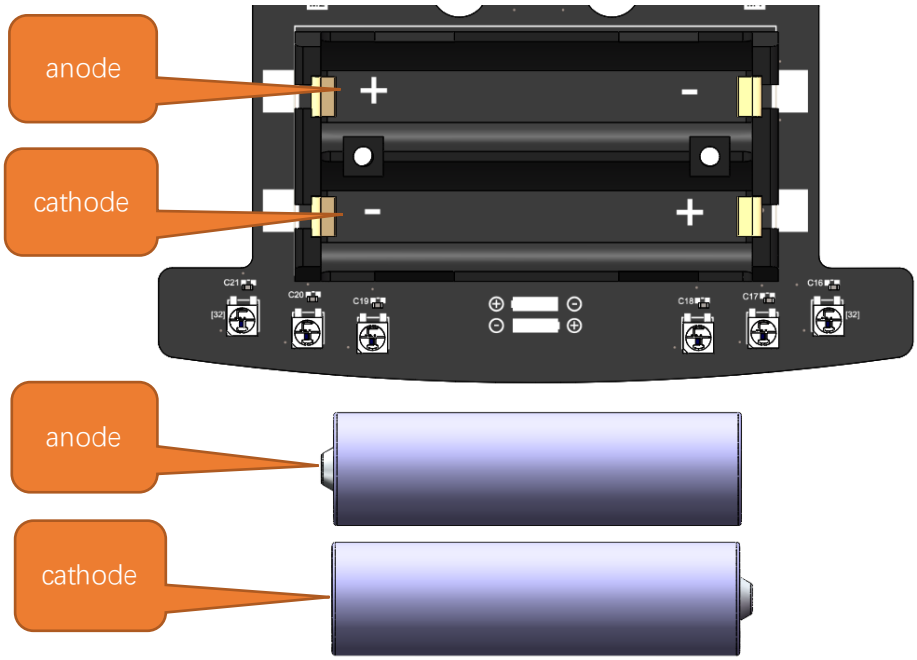
When you see the following content, it indicates that the program has been uploaded to ESP32.



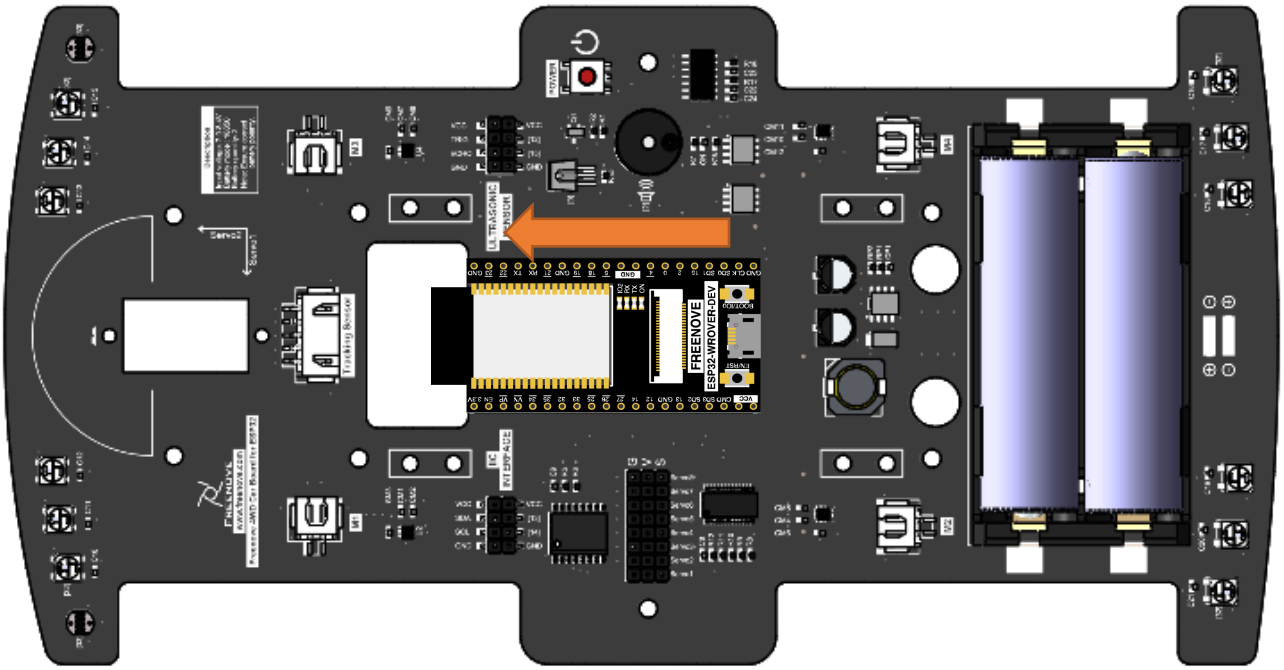
Note: For macOS users, if the uploading fails, please set the baud rate to 115200 before clicking "Upload Using Programmer".



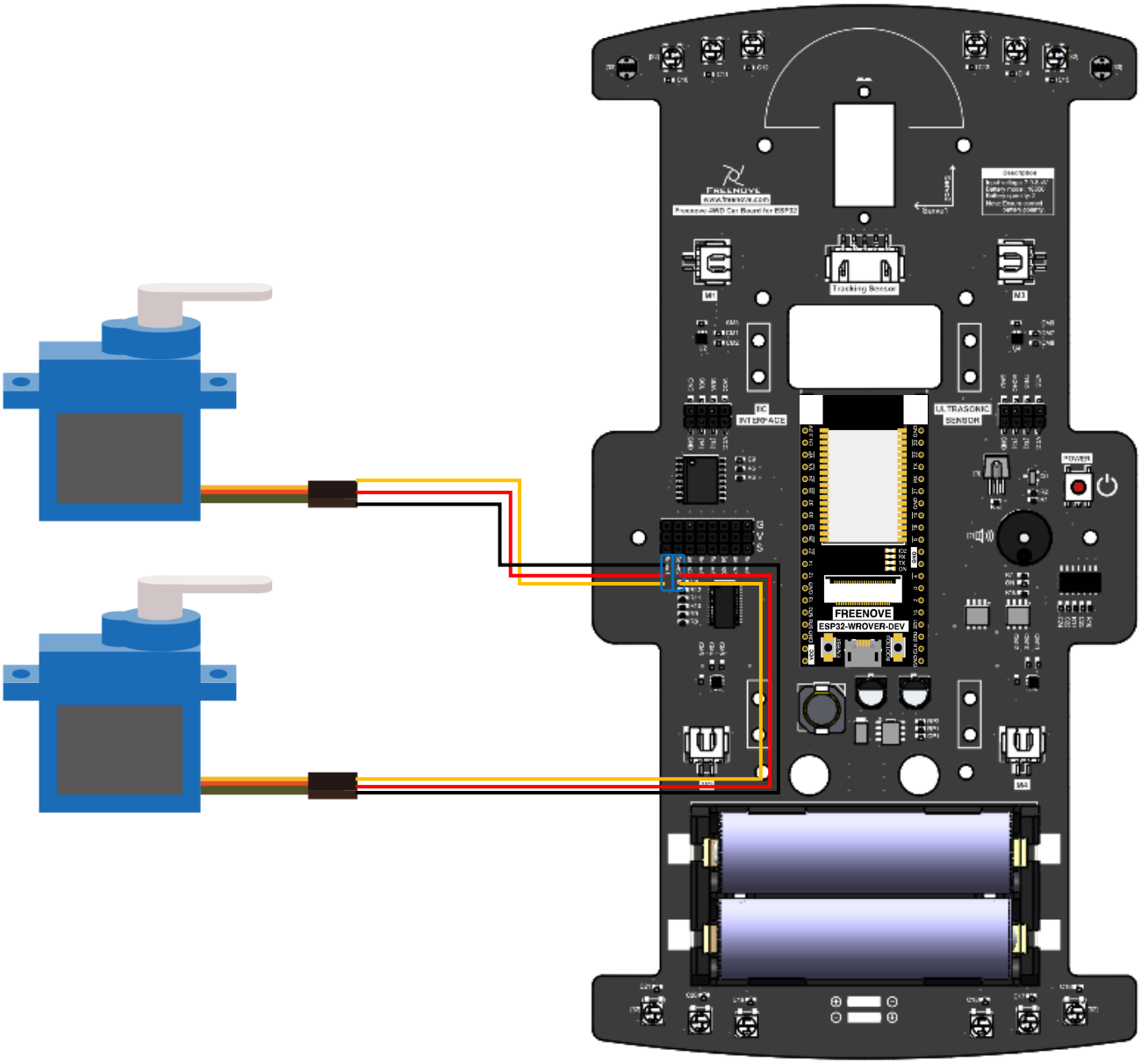
The car needs to be installed with batteries. When installing them, please following the silk print on the board.



Plug the ESP32 to the car shield. Pay attention to the orientation of ESP32.



Make sure ESP32 is plugged into the shield correctly. Take out two servo motors and plug them into the car shield. Please note the color of the wires Do NOT connect them wrongly.



Turn ON the switch and the two servos will keep at 90°.

Chapter 1 Assembling Smart Car

If you have any concerns, please feel free to contact us via support@freenove.com

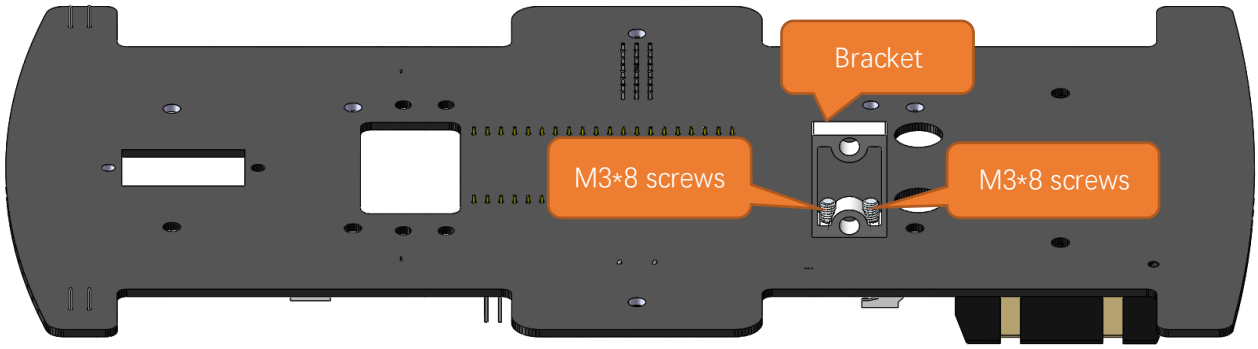
Assembling the Car

Installing Motor and Wheels

There is a special fixed bracket to fix motor, which contains an aluminum bracket, two M3*30 screws, two M3*8 screws, and two M3 nuts, as shown below:

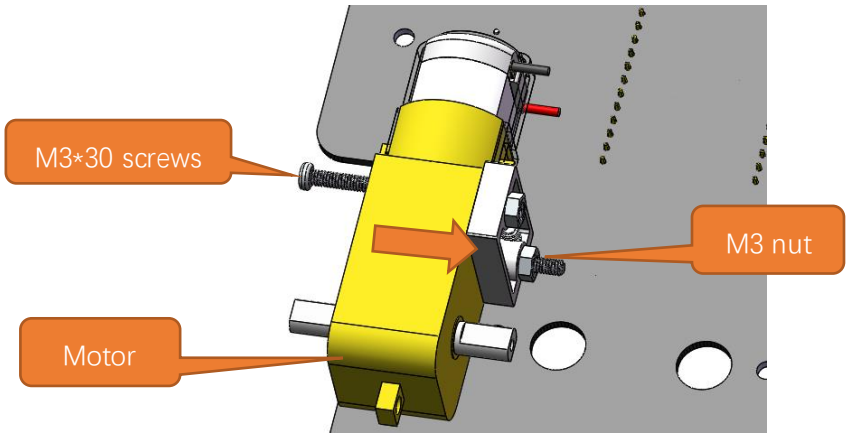
Installation steps:

Step 1 Install servos to fix the bracket on the shield.



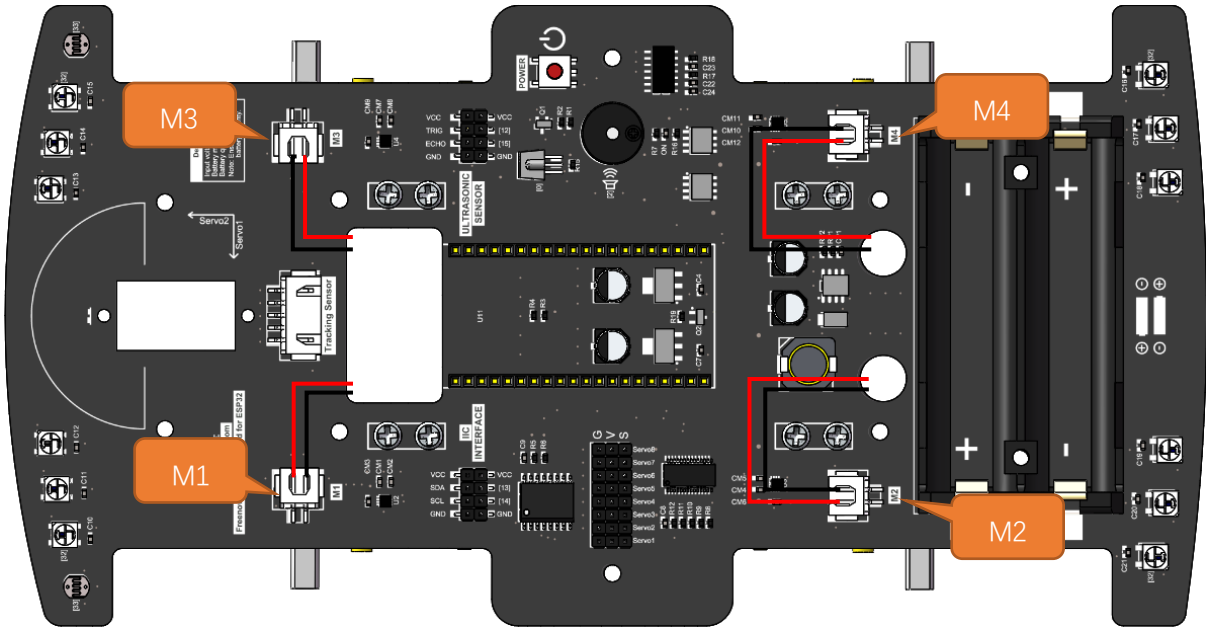
Put the bottom of the car up, use two M3*8 screws to fix the bracket on the shield.

Step 2 Install the motor to the bracket.



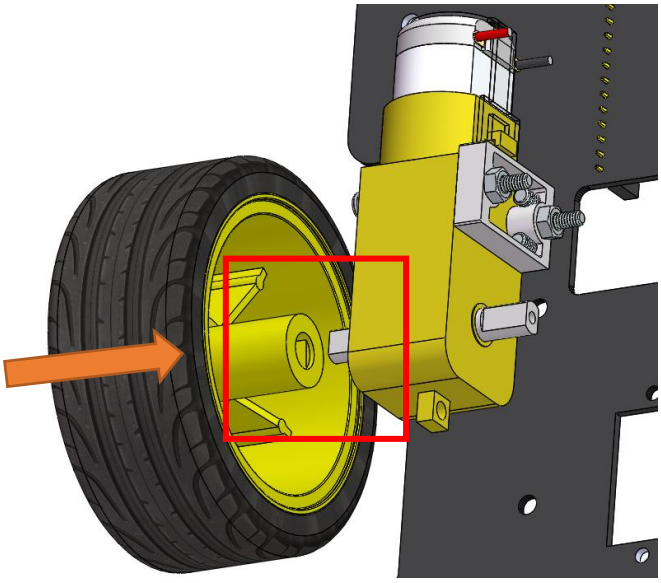
Use two M3*30 screws and two M3 nuts to fix the motor on the bracket

Step 3 Connect the motor wire to the corresponding terminal.

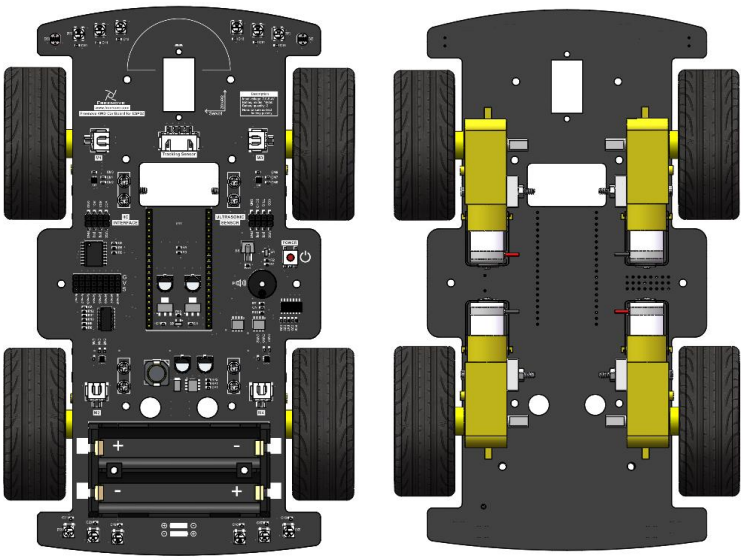


Pass the motor wire through the wire hole of the shield and connect it to the motor interface on the top.

Step 4 Install the wheel to the motor.



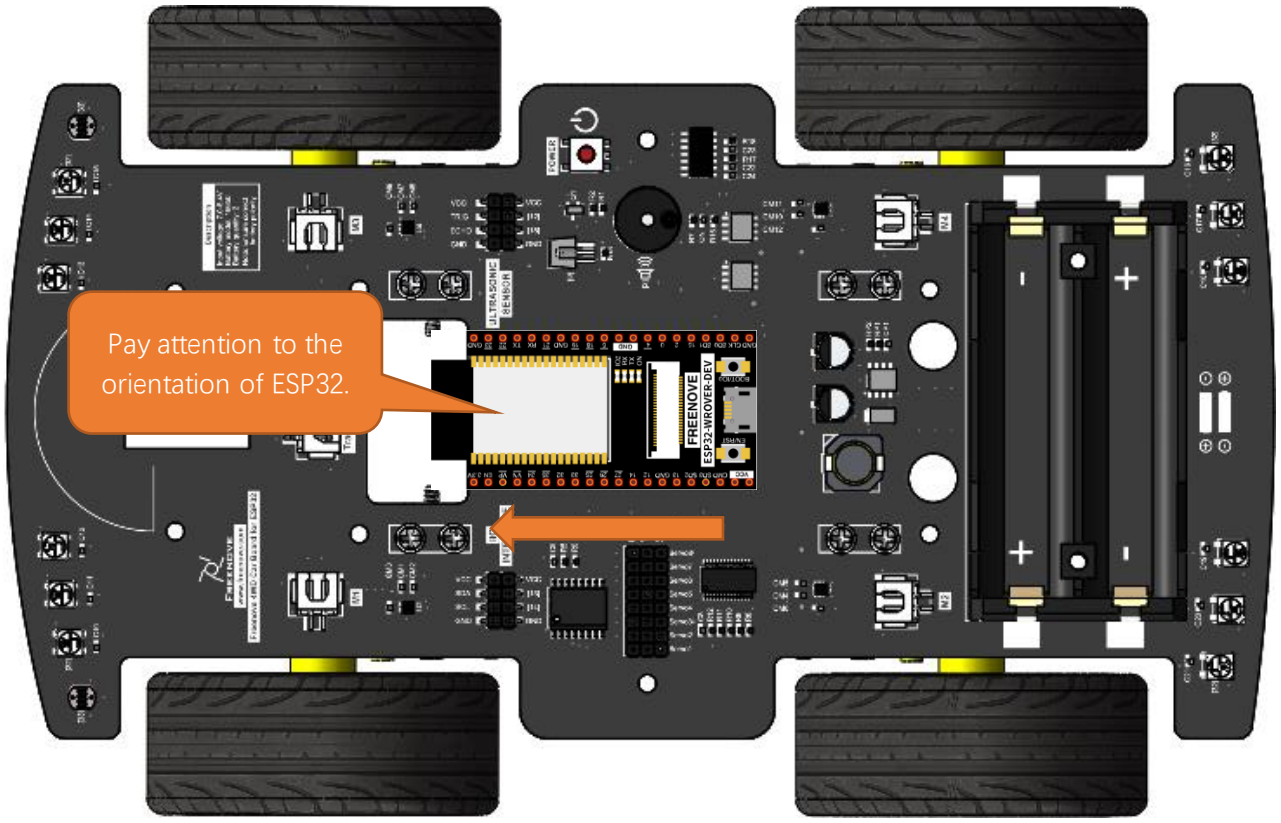
Note: The hole is not circle. Please align the hole and fix the wheel to the motor.



The installation of the four wheels is similar. Repeat the above installation steps to complete the installation

Installing ESP32

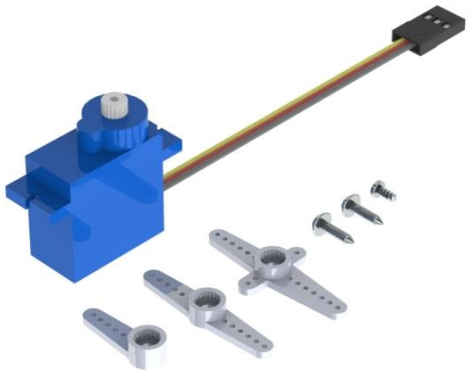
Step: Plug ESP32-Wrover-Dev in the shield.



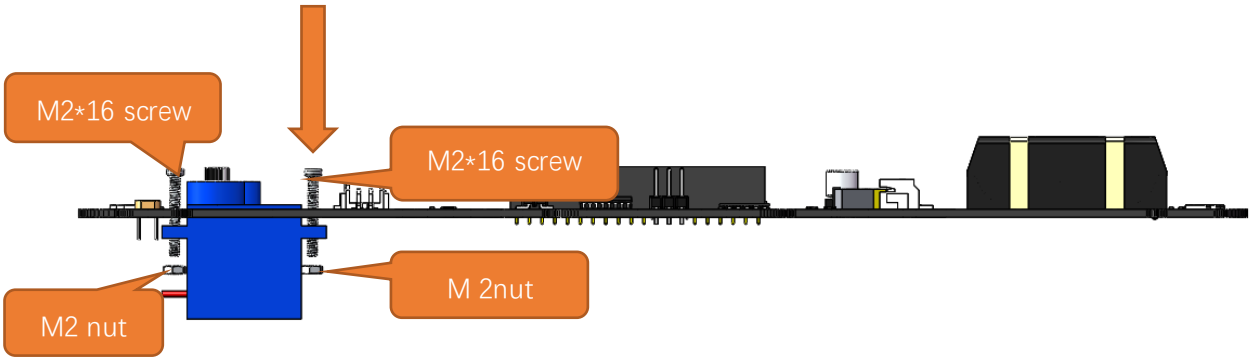
Please pay attention to the orientation of ESP32. Do NOT reverse it, otherwise it may burn the ESP32.

Installing Servo

Servo package: one servo, three rocker arms, two M2*8 screws and one M2*4 screw.

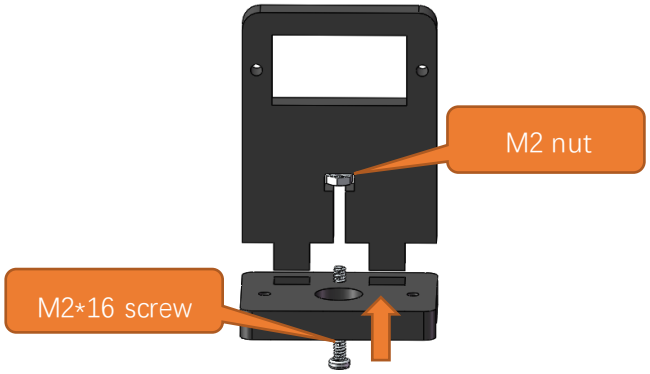


Step 1 Fix servo1 to the shield.



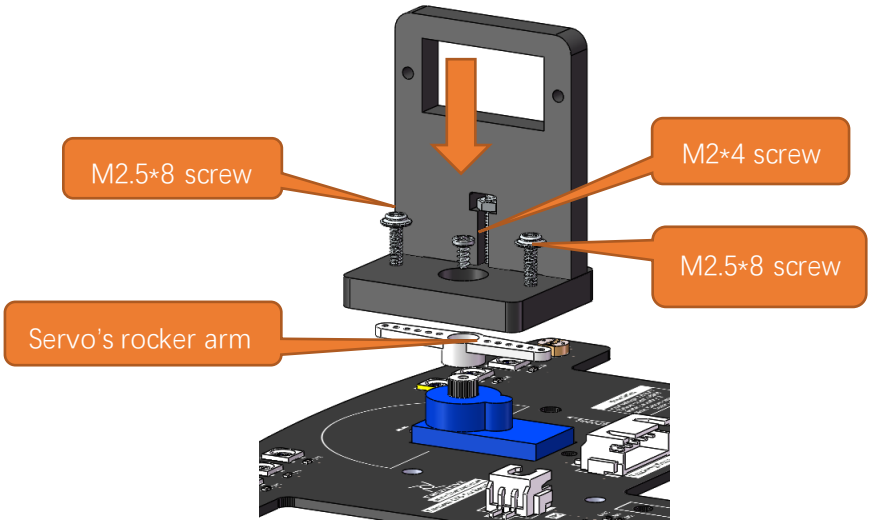
Use two M2*16 screws and two M2 nuts to fix the servo1 on the shield. Pay attention to the servo's direction.

Step 2 Fix two acrylics of the Pan-tilt together.



Use a M2*16 screw and a M2 nut to fix two acrylics of the Pan-tilt together.

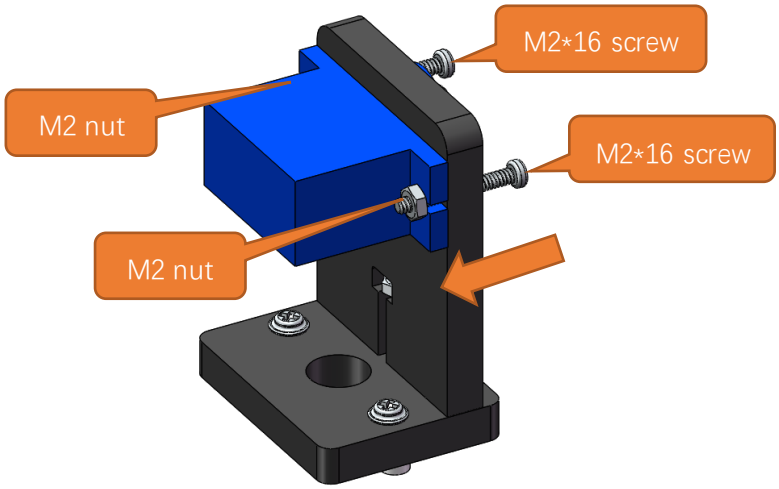
Step 3 Fix the Pan-tilt to servo1.



Use two M2.5*8 screws to fix the pan-tilt to rocker arm and a M2*4 to fix the rocker arm to servor1.

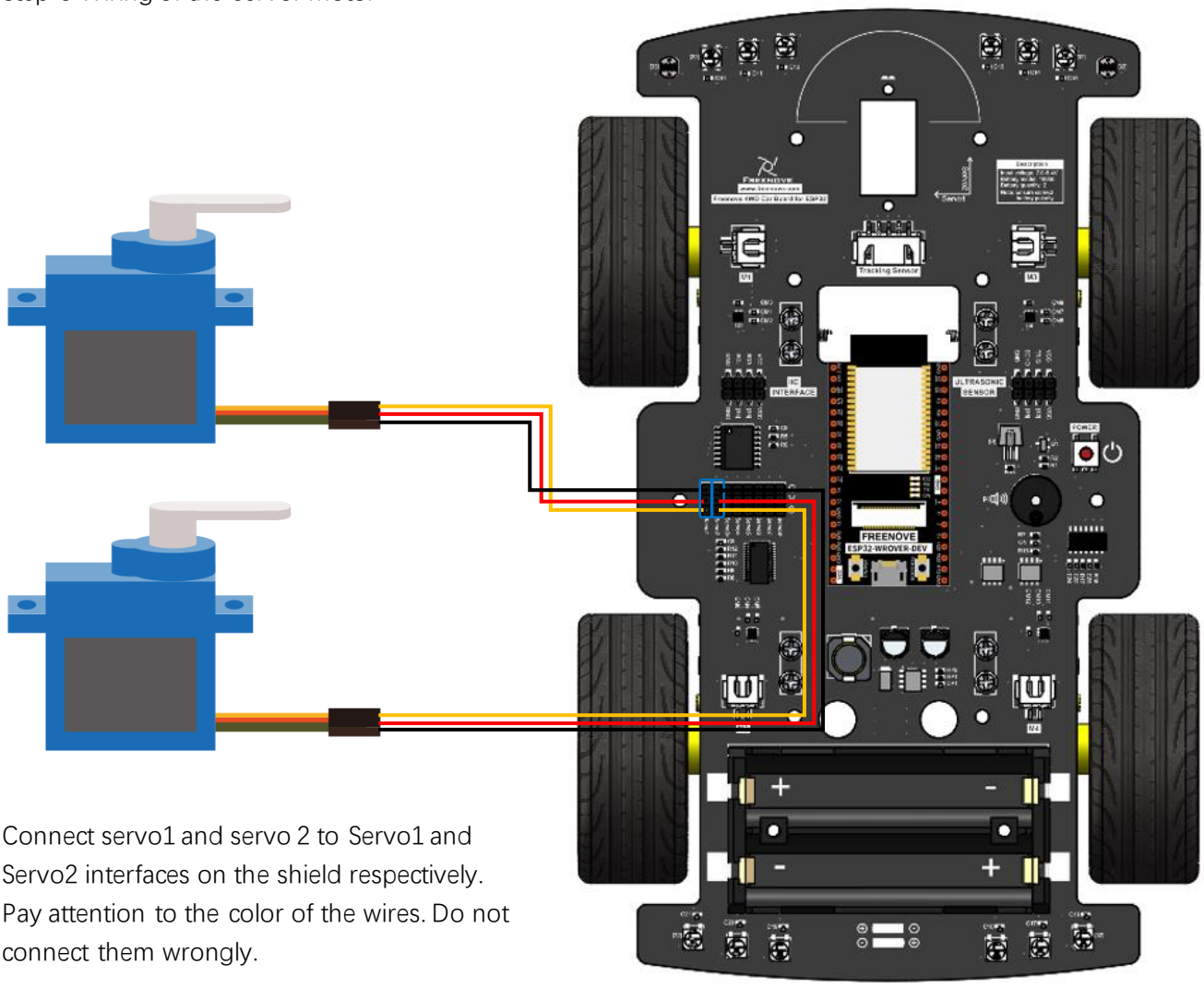
Note: Before fixing the rocker arm to servo1, please adjust the servo at 90°.You can refer to here.

Step 4 Fix servo2 to Pan-tilt.



Use two M2*16 screws and two M2 nuts to fix servo2 to Pan-tilt.

Step 5 Wiring of the servor motor



Connect servo1 and servo 2 to Servo1 and Servo2 interfaces on the shield respectively. Pay attention to the color of the wires. Do not connect them wrongly.

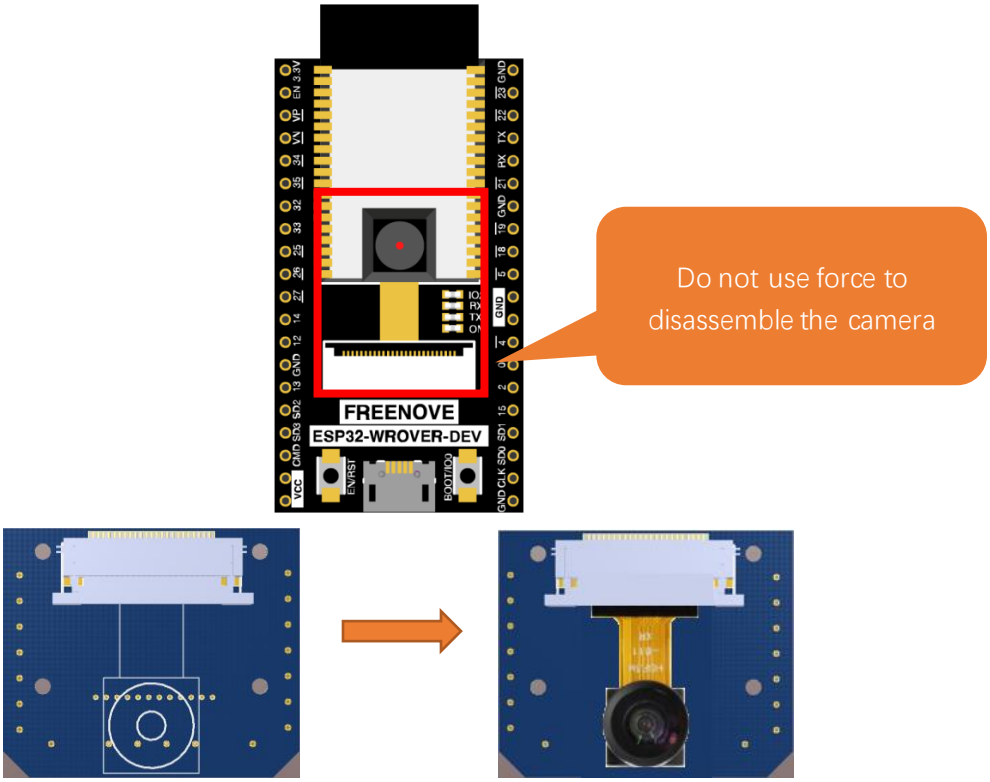
Assembling LED Matrix Pan-tilt.

Acrylic parts of the Pan-tilt are as follows:

Installation steps: (**Note: Do not disorder Servo1 and Servo2 during the installation.**)

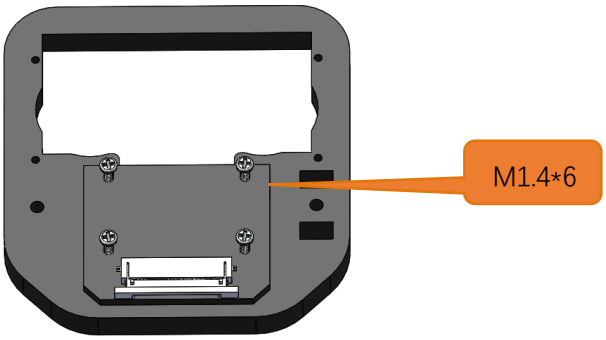
Step 1

Remove the camera from the ESP32.



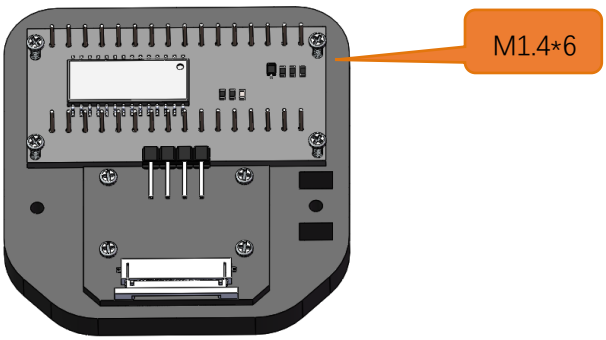
Install the camera to the board. Note: The connector and the ESP32 are flip-top type, please do not install it violently.

Step 2



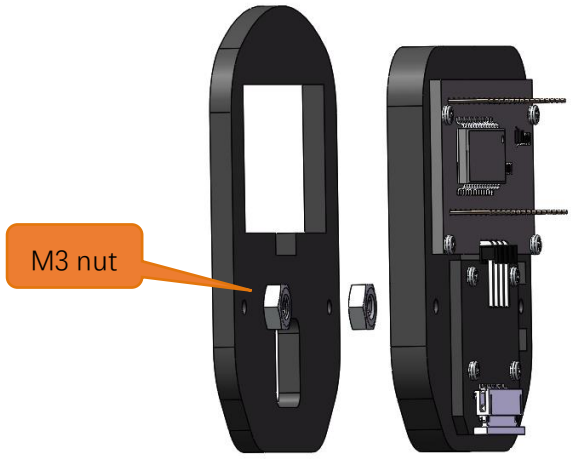
Use four M1.4*6 screws to fix the camera connector to the Pan-tilt.

Step 3



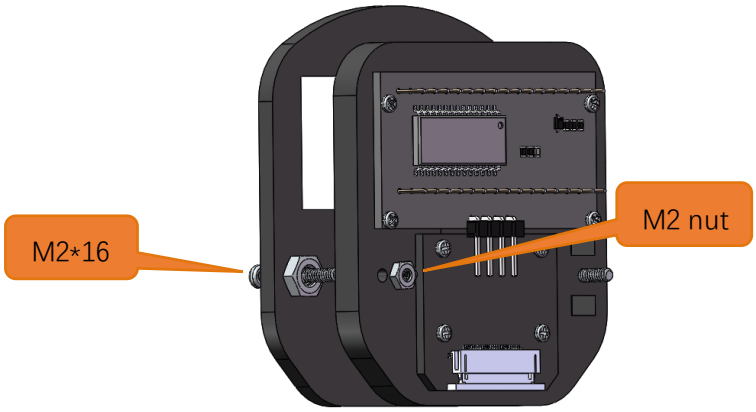
Use four M1.4*6 screws to fix the LED matrix to the Pan-tilt.

Step 4



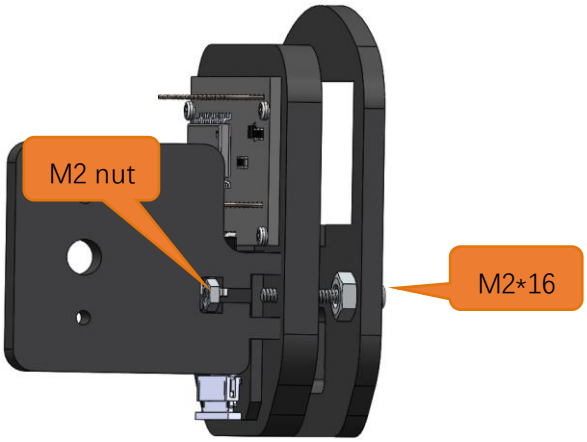
Between two acrylic plates, use two M3 nuts as washers.

Step 4



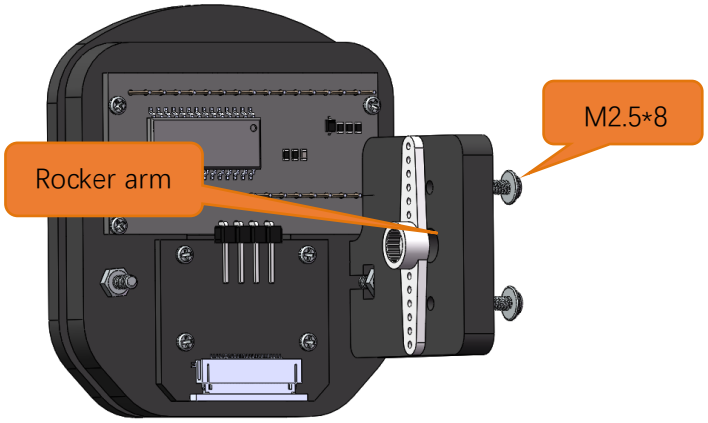
Use a M2*16 screw and a M2 nut to fix the two acrylics.

Step 5



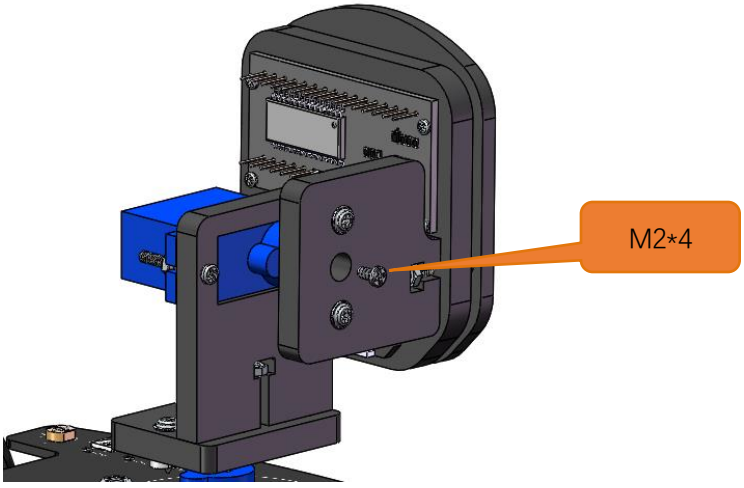
Use a M2*16 screw and a M2 nut to fix the two acrylic parts.

Step 6



Use two M2.5*8 screws to fix the rocker arm to acrylic part.

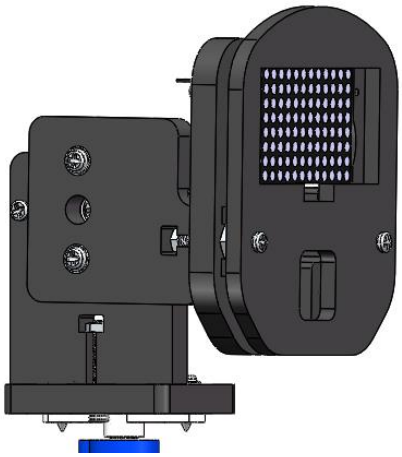
Step 7



Use two M2*4 screws to fix the ultrasonic pan-tilt to servor2.

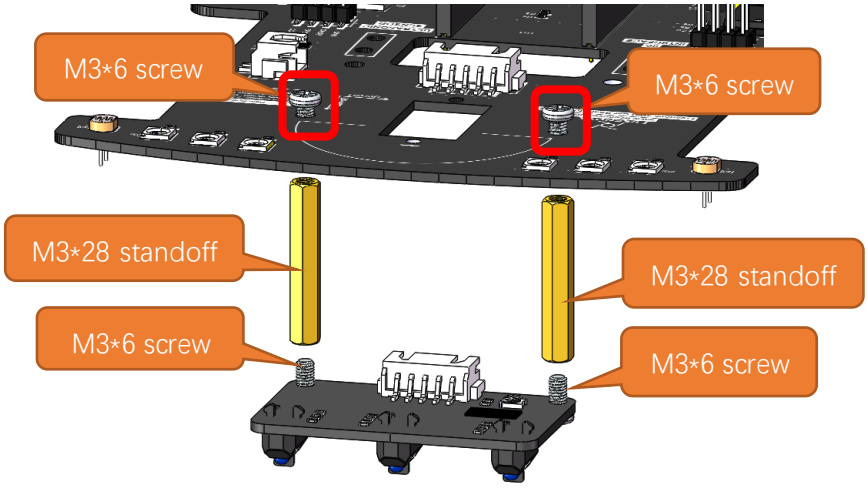
Note: Before fixing the rocker arm to servo2, please adjust the servo at 90°. You can refer to here.

After finished



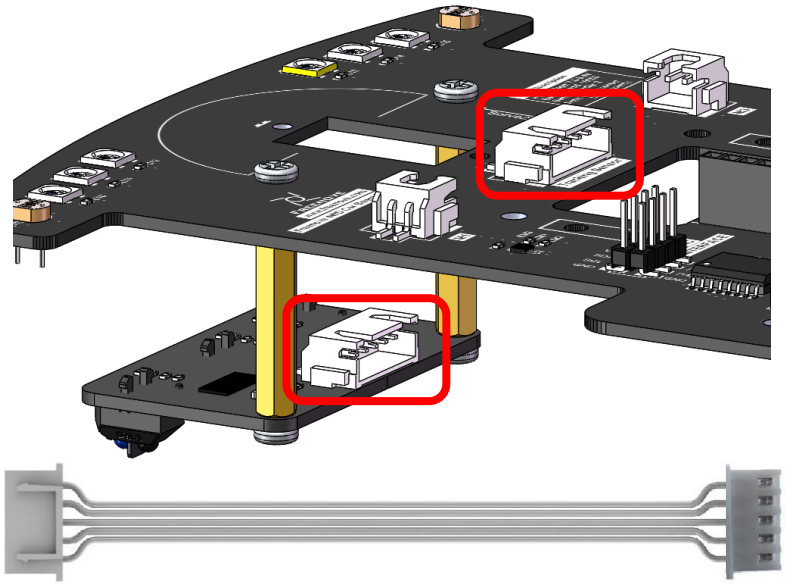
Installing Line Tracking Module

Step 1 Installing line tracking module



First, use two M3*6 screws to fix two M3*28 standoff to the bottom of the car, and then use two M3*6 screws to fix the line tracking module to standoff.

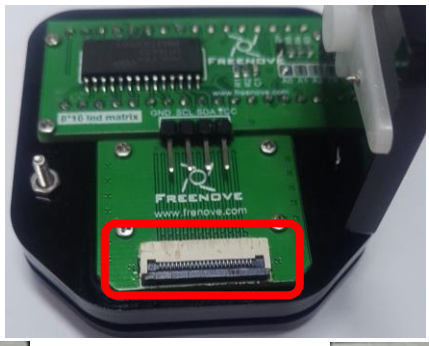
Step 2 Connect the cable to the tracking module



Use cable to connect the two connectors marked above.

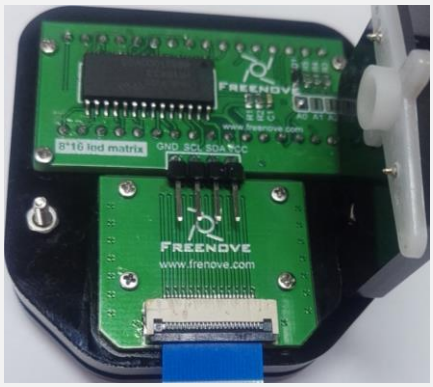
Wiring of Head

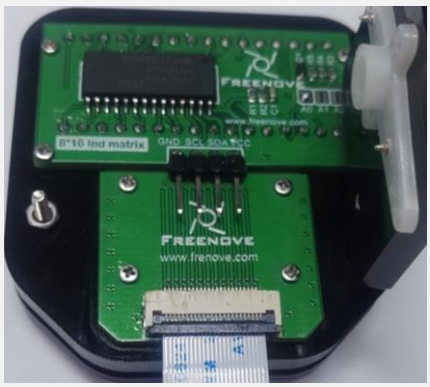
Step 1



Turn up the FPC connector.

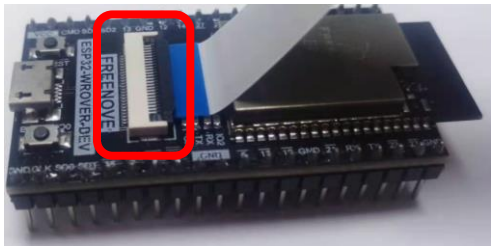
Step 2





Make sure that the blue side is upwards and the metal side is downwards. Do NOT connect it reversely.

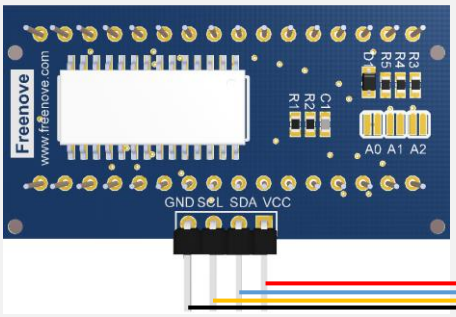
Step 3



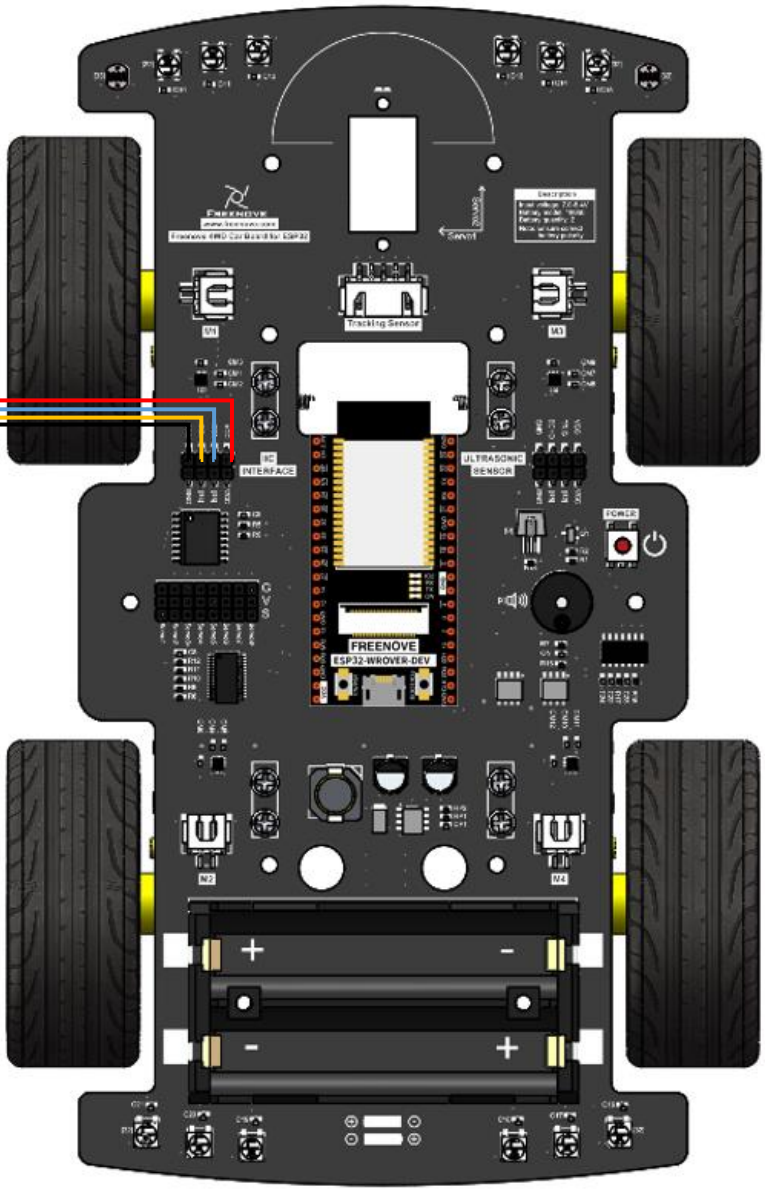
Make sure that the blue side is upwards and the metal side is downwards. Do NOT connect it reversely.

Please make sure that FPC Wire can pass through the acrylic sheet. (Refer to the “Installing the Acrylic Parts-Step 3”)

Step 4



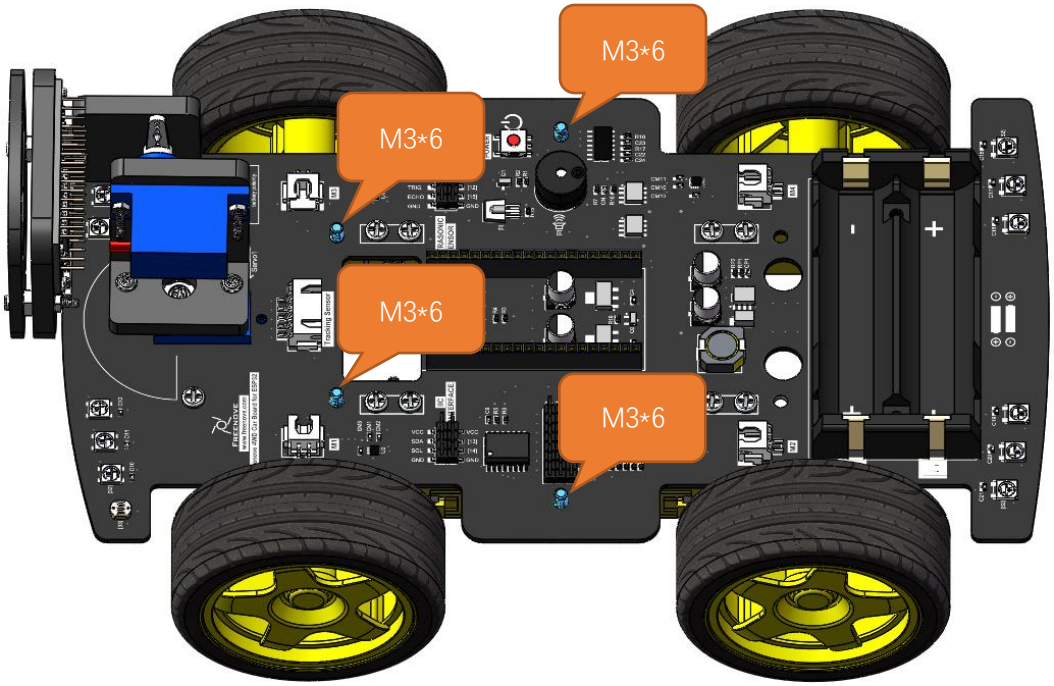
Module	GND	SCL	SDA	VCC
Car	GND	SCL	SDA	VCC
	GND	14	13	VCC



According to the prompts on the module and the car, use the 4P cable to connect the LED matrix module and the car. Please note that the order of the lines cannot be connected wrongly.

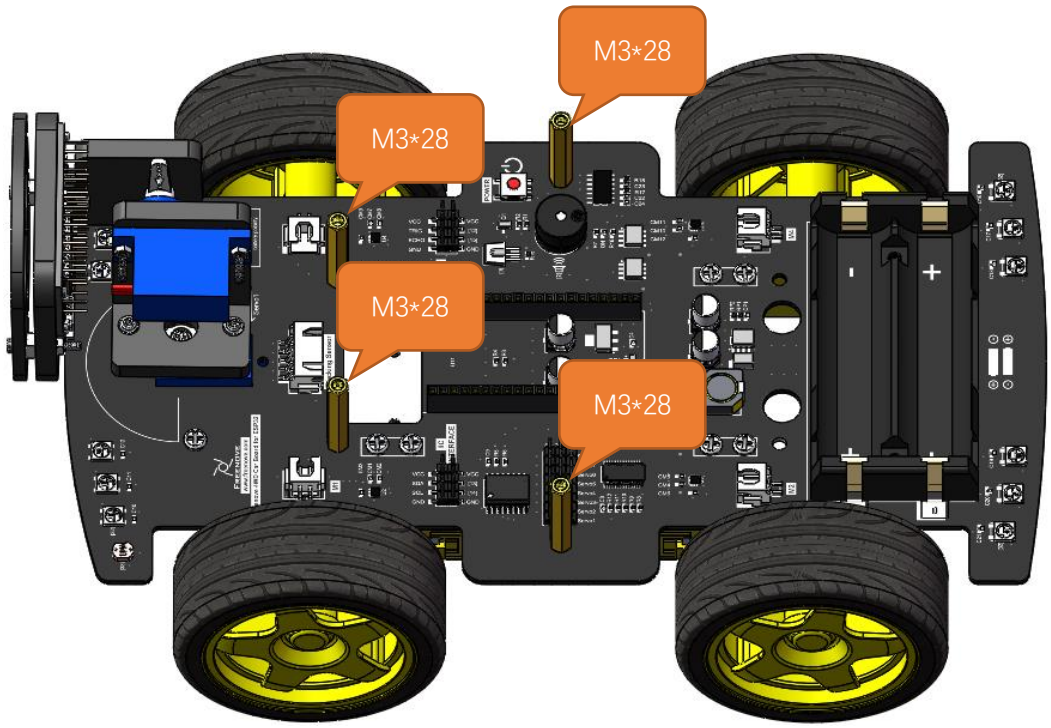
Installing the Acrylic Parts

Step 1



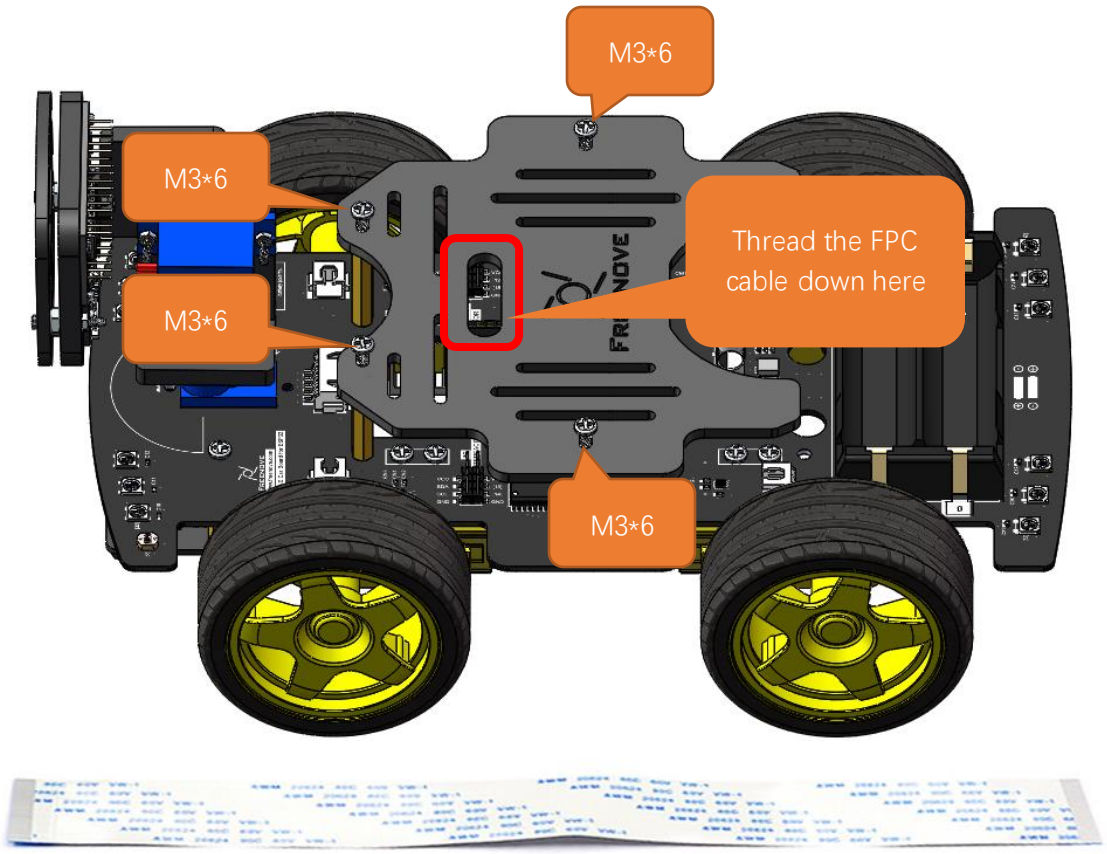
Pass four M3*6 screws upward from the bottom of the shield.

Step 2



Fix 4 M3*28 standoffs to M3*6 screws.

Step 3



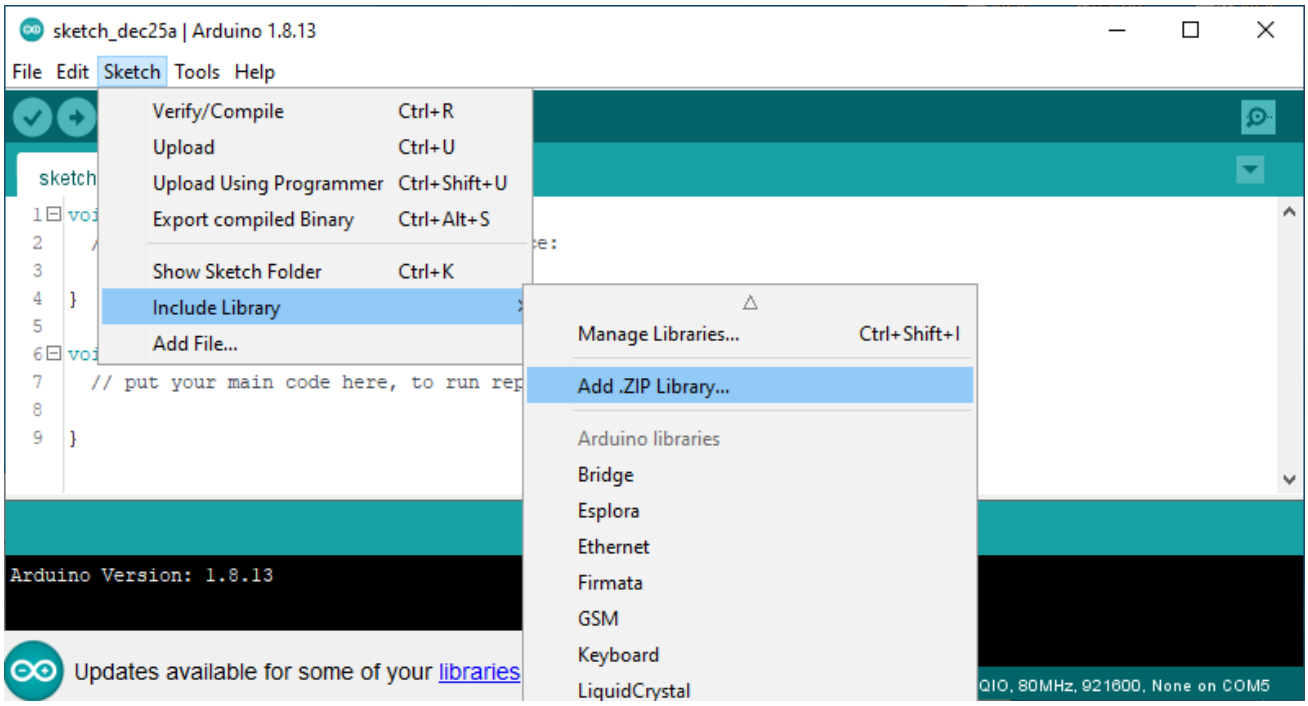
Align the acrylic part with the mounting hole of the standoffs, and use four M3*6 screws to fix the acrylic on the standoffs.

Please make sure that FPC Wire can pass through the acrylic sheet.

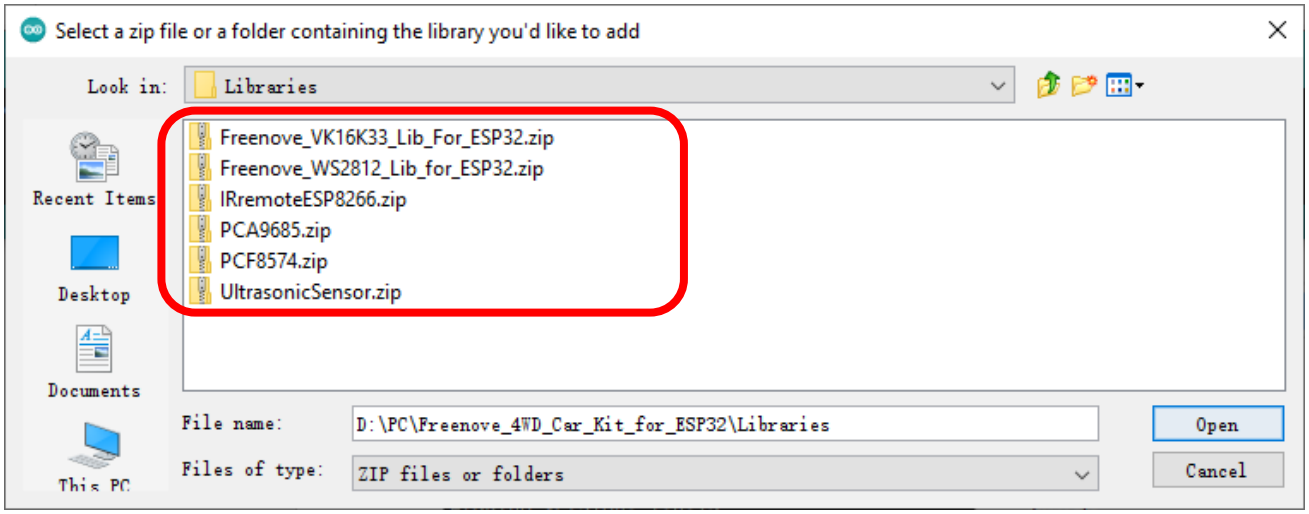
How to Play

Add libraries

Open the **Arduino IDE**, Click **Sketch** on the menu bar, select **Include Library**, click **Add .ZIP Library...**



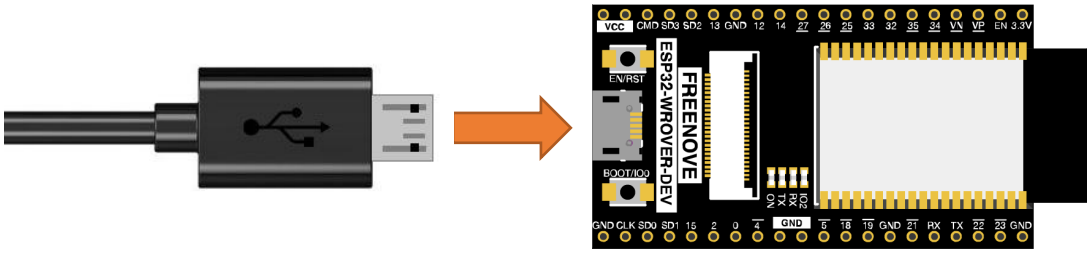
In the new pop-up window, select **Freenove_4WD_Car_Kit_for_ESP32\Libraries**, select every Library, click **Open**, and repeat this process several times until you have installed all six Libraries into the Arduino.



IR remote control

Step 1 Upload Code

Connect your computer and ESP32 with a USB cable.



Open the folder **Freenove_4WD_Car_Kit_for_ESP32\Sketches\05.3_Multi_Functional_Car**.

Double-click to open the **05.3_Multi_Functional_Car.ino**.

Click **Upload**.

Step 2 Control the Car

After the code is successfully uploaded, turn on the power of the car and use the infrared remote control to control the car and other functions. The corresponding keys and their functions are shown in the following table:

ICON	KEY Value	Function	ICON	KEY Value	Function
+	FF02FD	Move forward	TEST	FF22DD	Control the buzzer
⏮	FFE01F	Turn left	2	FF18E7	Random emoticons
⏭	FF906F	Turn light	5	FF38C7	Turn off emoticons
−	FF9867	Move back	7	FF42BD	Random display of WS2812
▶	FFA857	Stop the car	8	FF4AB5	Turn off WS2812 display
0	FF6897	Control servo 1 turn left	C	FFB04F	Control servo 2 turn left
1	FF30CF	Control servo 1 turn right	3	FF7A85	Control servo 2 turn right
4	FF10EF	Control servo 1 turn to 90°	6	FF5AA5	Control servo 2 turn to 90°

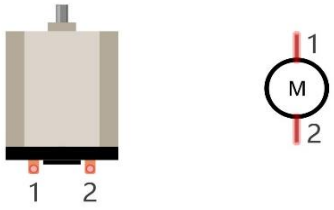
Chapter 2 Module test

If you have any concerns, please feel free to contact us via support@freenove.com

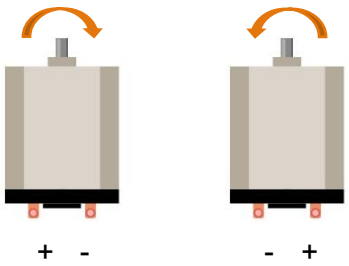
2.1 Motor

Motor

A motor is a device that converts electrical energy into mechanical energy. Motor consists of two parts: stator and rotor. When motor works, the stationary part is stator, and the rotating part is rotor. Stator is usually the outer case of motor, and it has terminals to connect to the power. Rotor is usually the shaft of motor, and can drive other mechanical devices to run. Diagram below is a small DC motor with two pins.



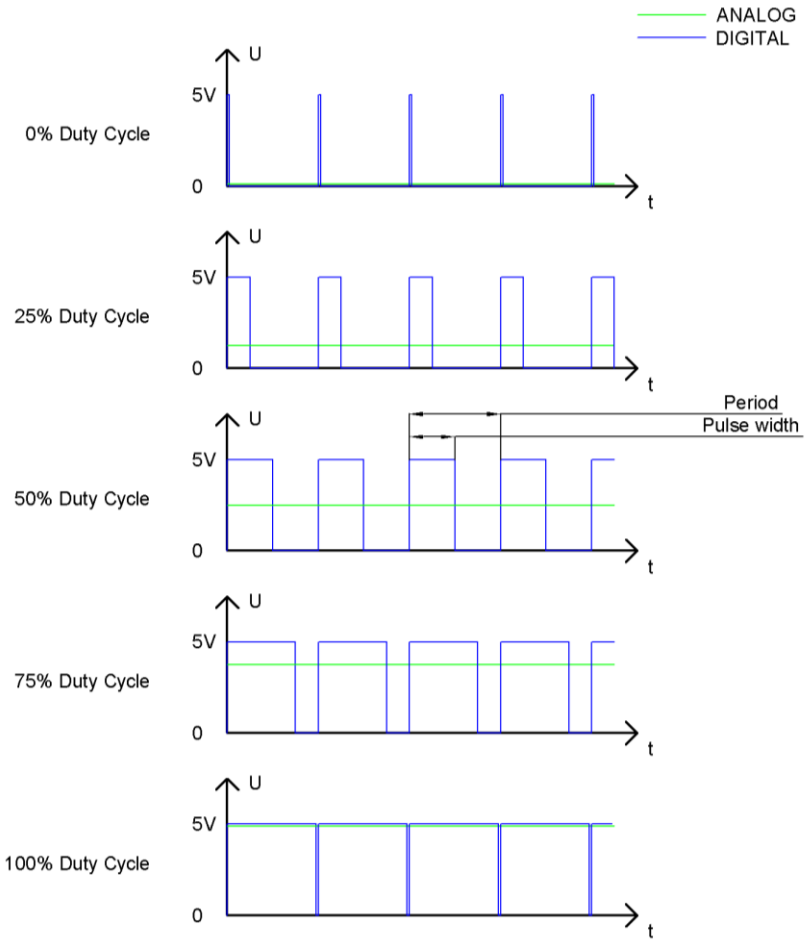
When a motor gets connected to the power supply, it will rotate in one direction. Reverse the polarity of power supply, then the motor rotates in opposite direction.



PWM

PWM, Pulse Width Modulation, uses digital pins to send certain frequencies of square waves, that is, the output of high levels and low levels, which alternately last for a while. The total time for each set of high levels and low levels is generally fixed, which is called the period (the reciprocal of the period is frequency). The time of high level outputs are generally called "pulse width", and the duty cycle is the percentage of the ratio of pulse duration, or pulse width (PW) to the total period (T) of the waveform.

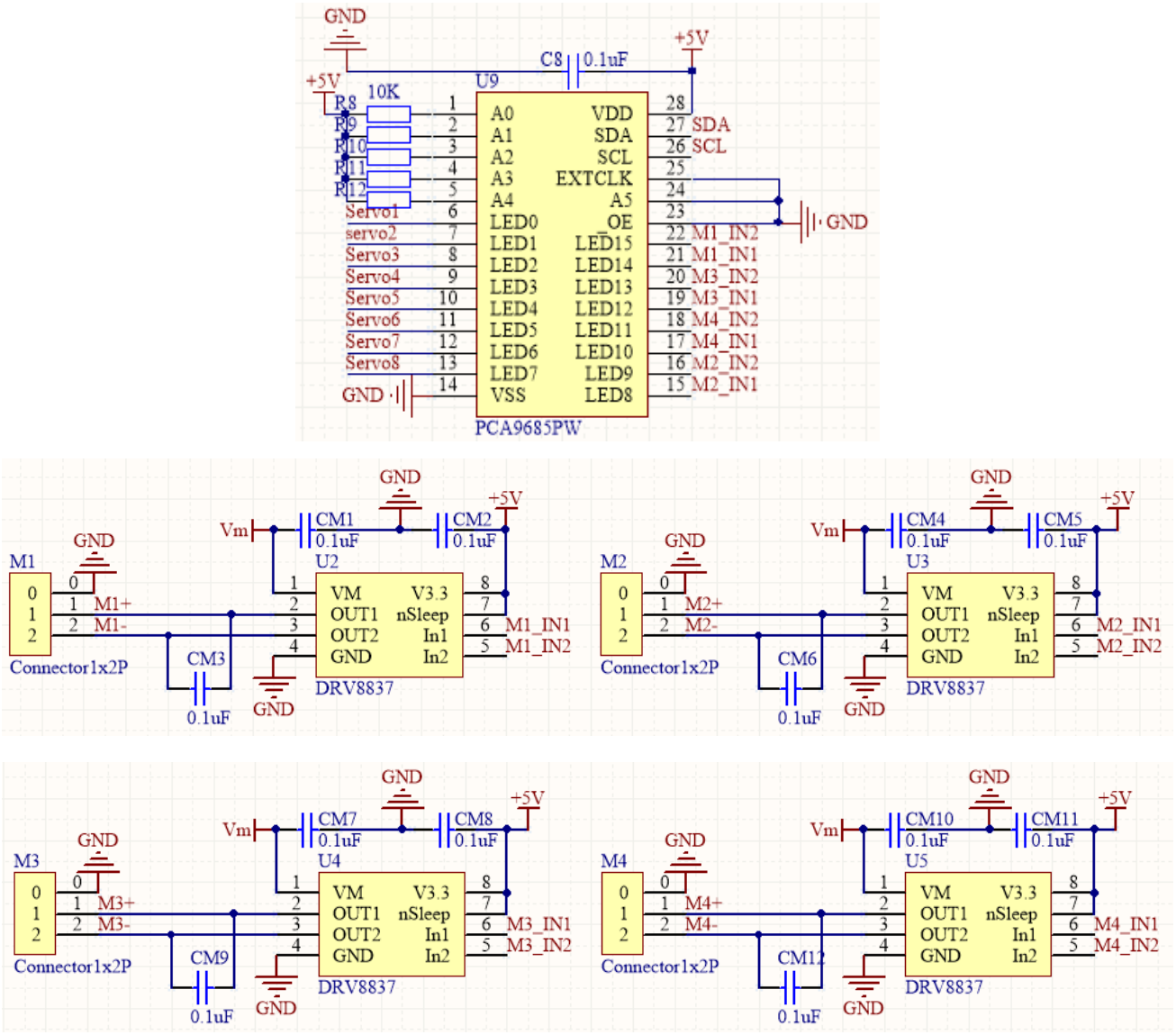
The longer the output of high levels last, the larger the duty cycle and the higher the corresponding voltage in analog signal will be. The following figures show how the analogs signal voltage vary between 0V-5V (high level is 5V) corresponding to the pulse width 0%-100%:



The longer the PWM duty cycle is, the higher the output power will be. Now that we understand this relationship, we can use PWM to control the brightness of an LED or the speed of DC motor and so on.

Schematic

For this tutorial, the driver chip of the car is controlled by the 8-15th output channels of the IIC chip PCA9685. When driving the motor, it is necessary to initialize the PCA9685 chip first, and then set the PWM value of the corresponding output channel to make its output voltage drive the motor.



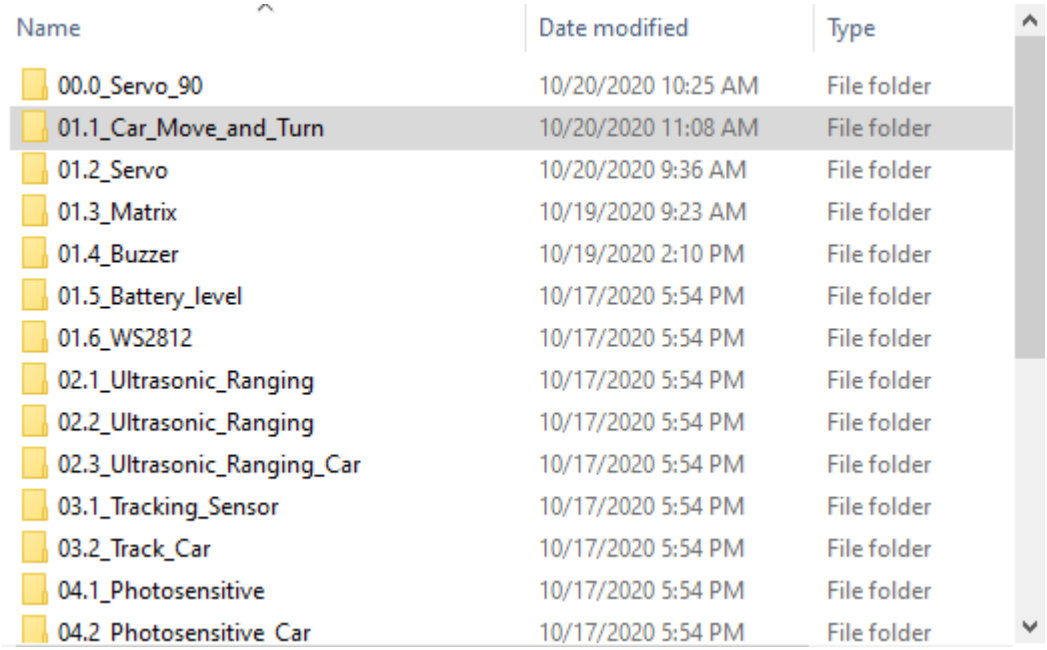
As can be seen from the above figure, if we want to control the motor M1, we need to set the 14th and 15th channels of the PCA9685.

Similarly, to control the motor M2, we need to set the 8th and 9th channels of the PCA9685, to control the motor M3, we need to set the 12th and 13th channels of the PCA9685, and to control the motor M4, we need to set the 10th and 11th channels of the PCA9685.

Mx_IN1	Mx_IN2	Rotating direction of the wheels
1	0	Forward
0	1	backward

Sketch

Next we will download the code to ESP32 to test the motor. Open "01.1_Car_Move_and_Turn" folder in **"Freenove_4WD_Car_Kit_for_ESP32\Sketches"** and double-click "01.1_Car_Move_and_Turn.ino".



Code

```
#include "Freenove_4WD_Car_For_ESP32.h"

void setup()
{
  PCA9685_Setup();         //Initialize PCA9685 to control Motor
}

void loop()
{
  Motor_Move(2000, 2000, 2000, 2000);     //go forward
  delay(1000);
  Motor_Move(0, 0, 0, 0);                 //stop
  delay(1000);
  Motor_Move(-2000, -2000, -2000, -2000); //go back
  delay(1000);
  Motor_Move(0, 0, 0, 0);                 //stop
  delay(1000);

  Motor_Move(-2000, -2000, 2000, 2000);   //turn left
  delay(1000);
  Motor_Move(0, 0, 0, 0);                 //stop
  delay(1000);
```

```
23    Motor_Move(2000, 2000, -2000, -2000);   //turn right
24    delay(1000);
25    Motor_Move(0, 0, 0, 0);                 //stop
26    delay(1000);
27  }
```

After downloading the code, please put the car to a relatively open area. Turn on the power switch and you can see the car go forward, backward, turn left and turn right repeatedly.

Code Explanation

If you are not familiar with Arduino IDE, you can visit https://www.arduino.cc/reference/en/ to learn more.

Add the header file of the car. Each time before controlling the car, please add header file first.

```
1   #include "Freenove_4WD_Car_For_ESP32.h"
```

Motor_Setup function, initialize the driver chip of the motor.

```
5     PCA9685_Setup();          //Initialize PCA9685 to control Motor
```

Loop functopn can be used repeatedly in the program. Here we call Motor_Move function to control the car to move forward, backward, turn left and turn right repeatedly.

```
8   void loop()
9   {
10    Motor_Move(2000, 2000, 2000, 2000);     //go forward
11    delay(1000);
12    Motor_Move(0, 0, 0, 0);                 //stop
13    delay(1000);
14    Motor_Move(-2000, -2000, -2000, -2000); //go back
15    delay(1000);
16    Motor_Move(0, 0, 0, 0);                 //stop
17    delay(1000);
18
19    Motor_Move(-2000, -2000, 2000, 2000);   //turn left
20    delay(1000);
21    Motor_Move(0, 0, 0, 0);                 //stop
22    delay(1000);
23    Motor_Move(2000, 2000, -2000, -2000);   //turn right
24    delay(1000);
25    Motor_Move(0, 0, 0, 0);                 //stop
26    delay(1000);
27  }
```

Motor_Move function is to control the car. The four parameters m1_speed, m2_speed, m3_speed, m4_speed range from -4095 to 4095. When the value is positive number, the motors move forward. Otherwise, they move backwards. m1_speed, m2_speed represent the two motors on the left and m3_speed, m4_speed embody those on the right.

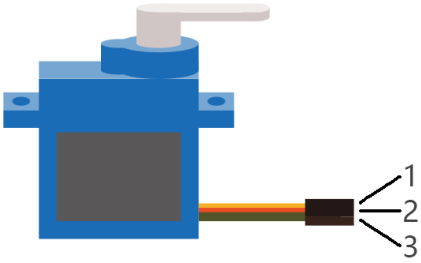
```
    void Motor_Move(int m1_speed, int m2_speed, int m3_speed, int m4_speed);
```

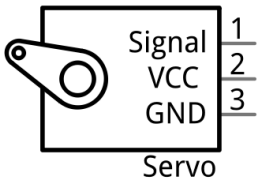
If you are not familiar with the position of the motors, you can click here to check the previous content.

2.2 Servo

Servo

Servo is a compact package which consists of a DC motor, a set of reduction gears to provide torque, a sensor and control circuit board. Most servos only have a 180-degree range of motion via their "horn". Servos can output higher torque than a simple DC motor alone and they are widely used to control motion in model cars, model airplanes, robots, etc. Servos have three wire leads which usually terminate to a male or female 3-pin plug. Two leads are for electric power: positive (2-VCC, Red wire), negative (3-GND, Brown wire), and the signal line (1-Signal, Orange wire), as represented in the Servo provided in your Kit.





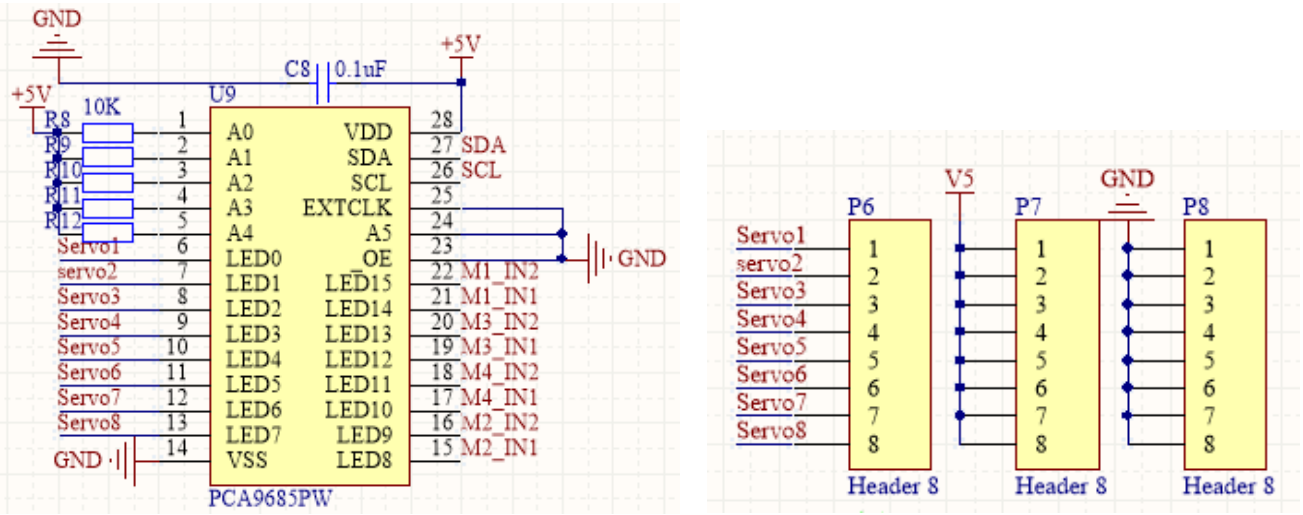
We will use a 50Hz PWM signal with a duty cycle in a certain range to drive the Servo. The lasting time of 0.5ms-2.5ms of PWM single cycle high level corresponds to the servo angle 0 degrees - 180 degree linearly. Part of the corresponding values are as follows:

High level time	Servo angle
0.5ms	0 degree
1ms	45 degree
1.5ms	0 degree
2ms	45 degree
2.5ms	180 degree

When you change the servo signal value, the servo will rotate to the designated angle.

Schematic

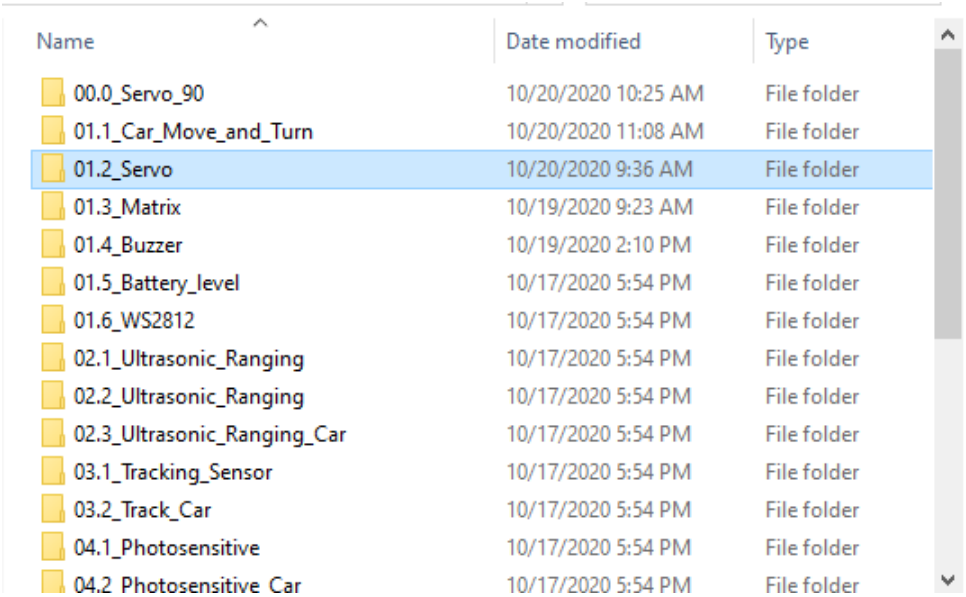
For this tutorial, the driver chip of the servo is controlled by the 0-7th output channels of the IIC chip PCA9685. When driving the servos, it is necessary to initialize the PCA9685 chip first, and then set the PWM value of the corresponding output channel to make its output voltage drive the servo.



As can be seen from the above figure, channels 0-7 of PCA9685 are used to control servos.

Sketch

Download the code to ESP32 to make servo sweep back and forth.
Open "01.2_Servo" in **"Freenove_4WD_Car_Kit_for_ESP32\Sketches"** and double-click "01.2_Servo.ino".



Code

```
#include "Freenove_4WD_Car_For_ESP32.h"

void setup()
{
  PCA9685_Setup();          //Initialize PCA9685 to control Servo
  Servo_1_Angle(90); //Set servo 1 Angle
  Servo_2_Angle(90); //Set servo 2 Angle
  delay(1000);
}

void loop()
{
  // Servo 1 motion path; 90° -0° - 180° - 90°
  Servo_Sweep(1, 90, 0);
  Servo_Sweep(1, 0, 180);
  Servo_Sweep(1, 180, 90);

  // Servo 2 motion path; 90° - 150° - 90°
  Servo_Sweep(2, 90, 150);
  Servo_Sweep(2, 150, 90);
}
```

Code Explanation

Add header file of servos. Each time before controlling servos, please add header file first.

```
#include "Freenove_4WD_Car_For_ESP32.h"
```

Initialize the servo driven chip.

```
  PCA9685_Setup();          //Initialize PCA9685 to control Servo
```

Set the angle of the two servos to 90°.

```
  Servo_1_Angle(90);
  Servo_2_Angle(90);
```

Sweep function of the servos. Servo_id stands for the servo number, ranging from 1 to 2. Angle_start represents the starting position of the servo when sweeping while angle_end is the ending position. Note: the sweeping range of servo1 is 0-180°, and that of servo2 is 90-150°.

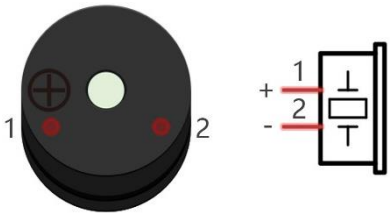
```
void Servo_Sweep(int servo_id, int angle_start, int angle_end);
```

2.3 Buzzer

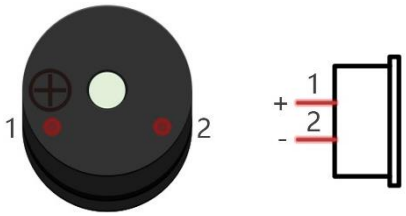
Buzzer

Buzzer is a sounding component, which is widely used in electronic devices such as calculator, electronic warning clock and alarm. Buzzer has two types: active and passive. Active buzzer has oscillator inside, which will sound as long as it is supplied with power. Passive buzzer requires external oscillator signal (generally use PWM with different frequency) to make a sound.

Active buzzer



Passive buzzer



Active buzzer is easy to use. Generally, it can only make a specific frequency of sound. Passive buzzer requires an external circuit to make a sound, but it can be controlled to make a sound with different frequency. The resonant frequency of the passive buzzer is 2kHz, which means the passive buzzer is loudest when its resonant frequency is 2kHz.

How to identify active and passive buzzer

1. Usually, there is a label on the surface of active buzzer covering the vocal hole, but this is not an absolute judgment method.
2. Active buzzers are more complex than passive buzzers in their manufacture. There are many circuits and crystal oscillator elements inside active buzzers; all of this is usually protected with a waterproof coating (and a housing) exposing only its pins from the underside. On the other hand, passive buzzers do not have protective coatings on their underside. From the pin holes viewing of a passive buzzer, you can see the circuit board, coils, and a permanent magnet (all or any combination of these components depending on the model.

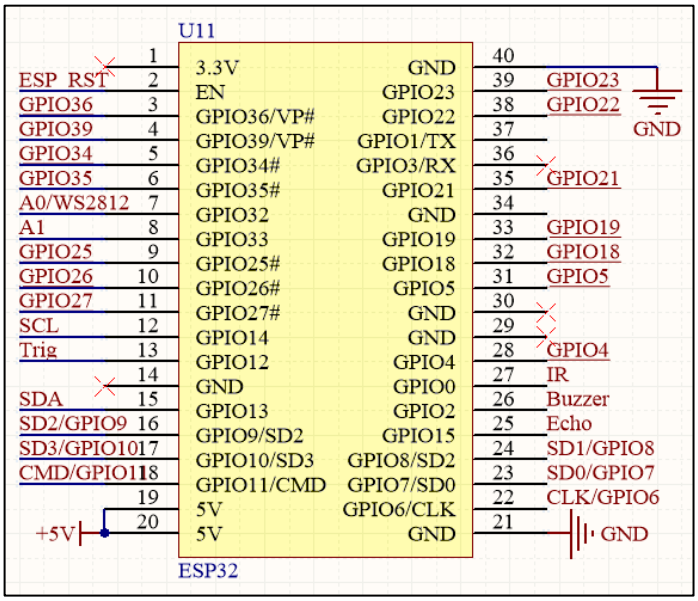
Active buzzer

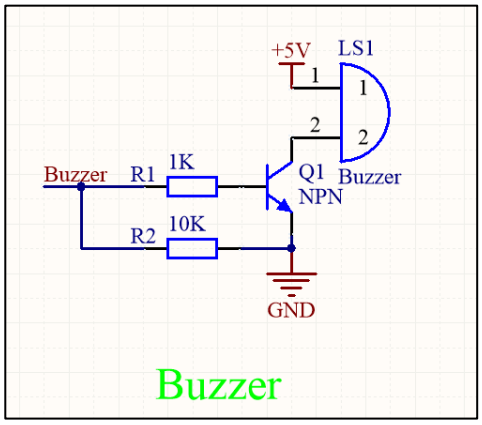
Passive buzzer

The buzzer used in this car is a passive buzzer that can make sounds with different frequence.

Schematic

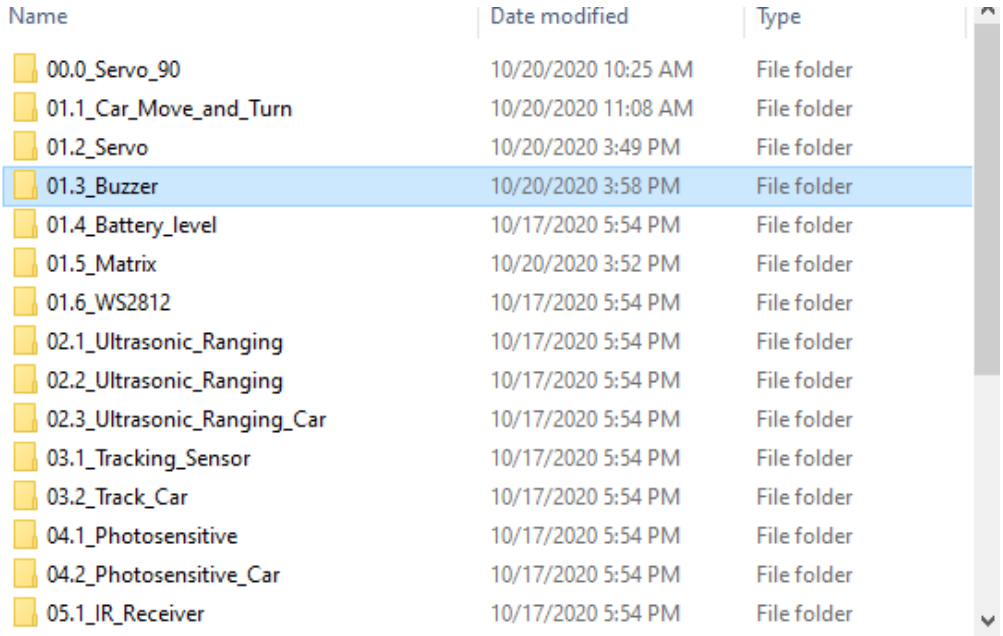
As we can see, the buzzer is controlled by GPIO2 of ESP32. When the buzzer receives PWM signal, NPN will be activated to make the buzzer sound. When the buzzer receives no signal, it will be controlled at low level by R2 and NPN won't be activated, then the buzzer won't make any sounds.





Sketch

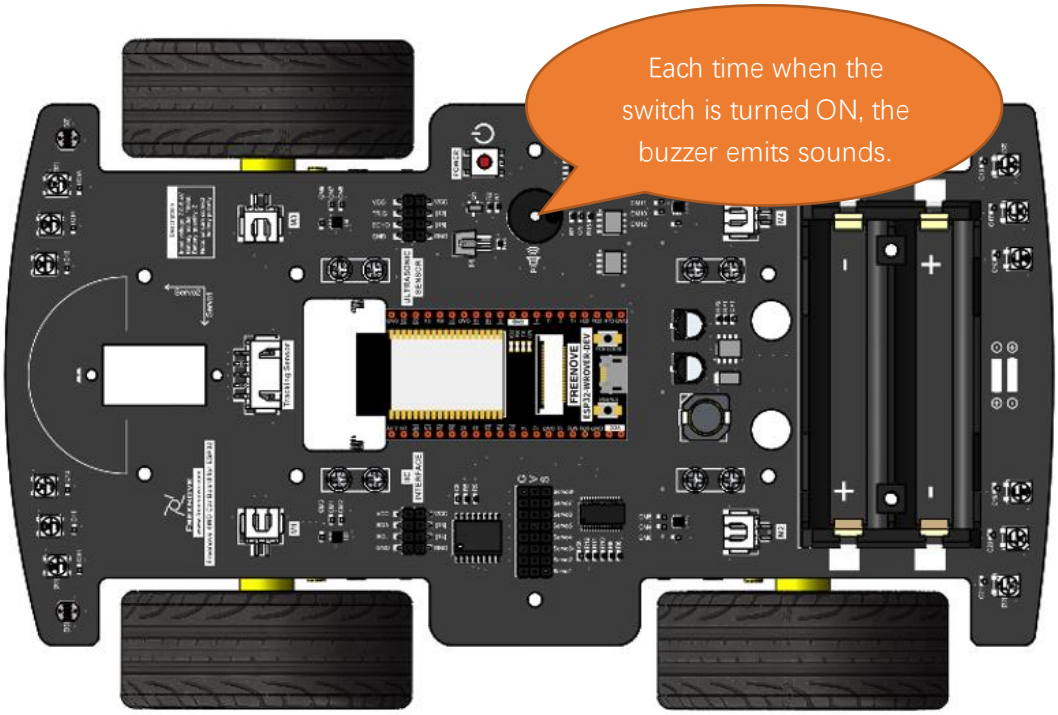
In this section, we will test the buzzer to make it sound like an alarm.
Open "01.3_Buzzer" folder in the **"Freenove_4WD_Car_Kit_for_ESP32\Sketches"**, and then double-click "01.3_Buzzer.ino".



Code

```
#include "Freenove_4WD_Car_For_ESP32.h"

void setup() {
  Buzzer_Setup();     //Buzzer initialization function
  Buzzer_Alert(4, 3);//Control the buzzer to sound
}

void loop() {
  delay(1000);
}
```

After the program is downloaded to ESP32, the buzzer emits 4 short beeps, repeating for 3 times.



Code Explanation

Configure the PWM of ESP32 to associate it with GPIO2 pin to control the buzzer to make sounds.

```
void Buzzer_Setup(void);                    //Buzzer initialization
```

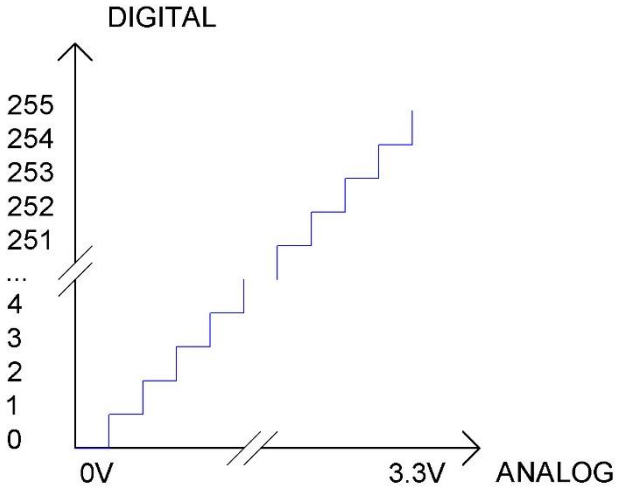
Control the buzzer to sound regularly. beat represents the number of times the buzzer sounds in each sounding cycle. Rebeat represents how many cycles the buzzer sounds.

```
void Buzzer_Alert(int beat, int rebeat);//Buzzer alarm function
```

2.4 ADC Module

ADC

ADC is an electronic integrated circuit used to convert analog signals such as voltages to digital or binary form consisting of 1s and 0s. The range of our ADC on ESP32 is 12 bits, that means the resolution is 2^12=4096, and it represents a range (at 3.3V) will be divided equally to 4096 parts. The rage of analog values corresponds to ADC values. So the more bits the ADC has, the denser the partition of analog will be and the greater the precision of the resulting conversion.



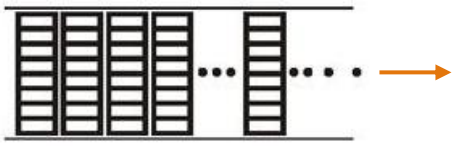
Subsection 1: the analog in rang of 0V---3.3/4095 V corresponds to digital 0;
Subsection 2: the analog in rang of 3.3/4095 V---2*3.3 /4095V corresponds to digital 1;
The following analog will be divided accordingly.
The conversion formula is as follows:

$$ADC\ Value = \frac{\text{Analog Voltage}}{3.3} * 4095$$

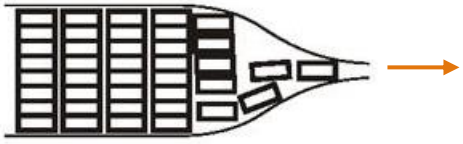
Serial Communication

Serial communication uses one data cable to transfer data one bit by another in turn. Parallel communication means that the data is transmitted simultaneously on multiple cables. Serial communication takes only a few cables to exchange information between systems, which is especially suitable for computers to computers, long distance communication between computers and peripherals. Parallel communication is faster, but it requires more cables and higher cost, so it is not appropriate for long distance communication.

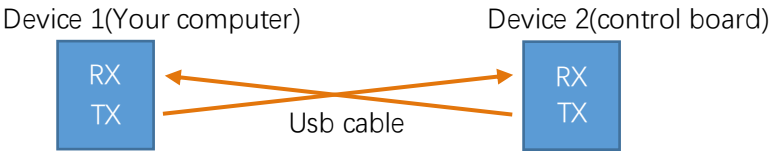
Parallel communication



Serial communication



Serial communication generally refers to the Universal Asynchronous Receiver/Transmitter (UART), which is commonly used in electronic circuit communication. It has two communication lines, one is responsible for sending data (TX line) and the other for receiving data (RX line). The serial communication connections of two devices use is as follows:

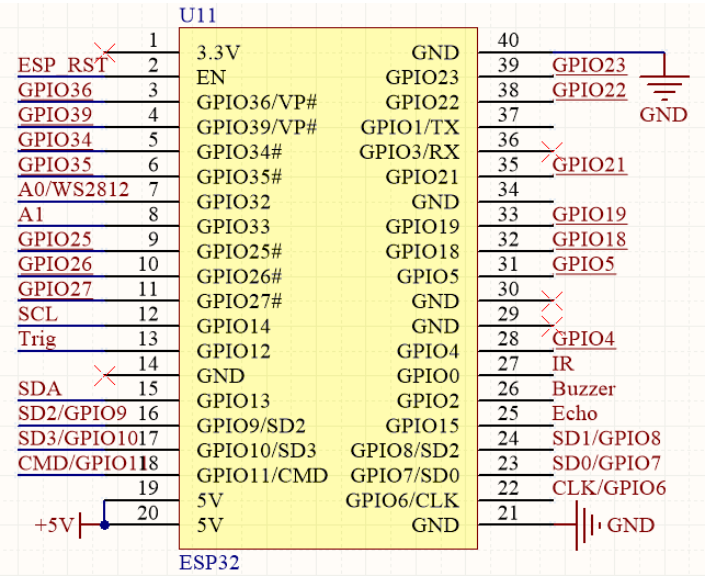


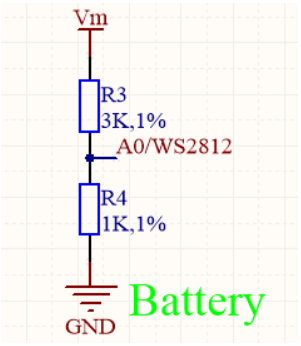
For serial communication, the **baud rate in both sides must be the same**. The baud rates commonly used are 9600 and 115200.

Computer identifies serial devices connected to your computer as COMx. We can use the Serial Monitor window of Arduino Software to communicate with Freenove control board.

Schematic

As we can see, the car reads the voltage of the batteries through GPIO32 of ESP32. Because the battery voltage is not read frequently, this GPIO is also used to control the WS2812 LED.

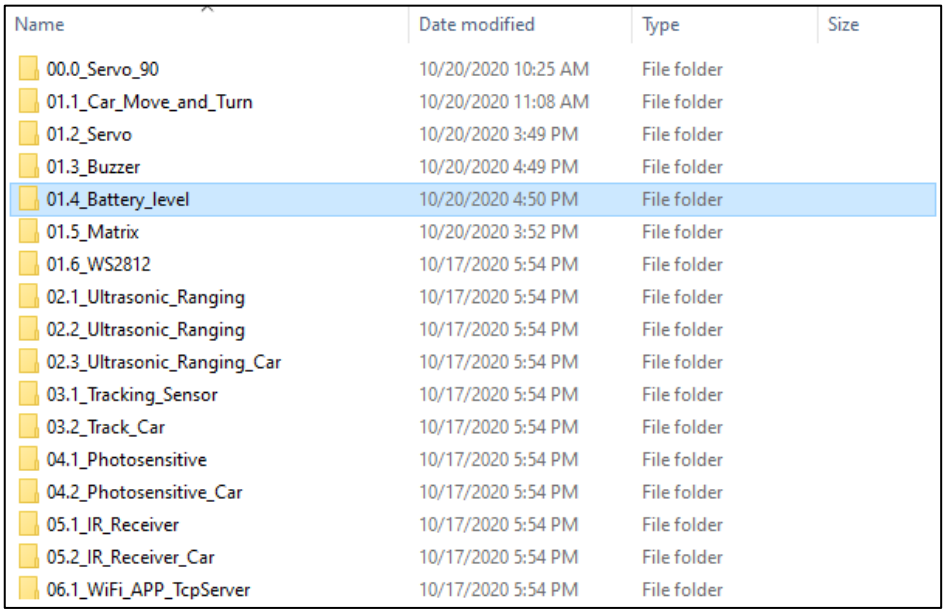




The voltage acquisition range of GPIO32 on ESP32 is 0-3.3V, while the car is powered by two 18650 lithium batteries, whose voltage is 8.4V when they are fully charged, which exceeds the acquisition range of ESP32. Therefore, after passing through the voltage divider circuit composed of R3 and R4, the voltage at A0/WS2812 will be about 1/4 of the battery voltage, 8.4/4=2.1V, which is within the voltage collection range of GPIO32.

Sketch

In this section, we will use GPIO32 of ESP32 to read the voltage value of the batteries and print it on serial monitor. Open "01.4_Battery_level" folder in **"Freenove_4WD_Car_Kit_for_ESP32\Sketches"** and then double-click "01.4_Battery_level.ino".



Code

```
1  #include "Freenove_4WD_Car_For_ESP32.h"
2
3  void setup() {
4    Serial.begin(115200);                      //Set the Serial Baud rate
5  }
6  void loop() {
7    Serial.print("Battery ADC : ");
8    Serial.println(Get_Battery_Voltage_ADC());//Gets the battery ADC value
9    Serial.print("Battery Voltage : ");
10   Serial.print(Get_Battery_Voltage());      //Get the battery voltage value
11   Serial.println("V");
12   delay(300);
13 }
```

Code Explanation

Activate the serial port and set the baud rate to 115200.

```
4    Serial.begin(115200);                    //Set the Serial Baud rate
```

Get ADC sampling value of GPIO32 and return it. The ADC has a range of 0-4095. The voltage range collected is 0-3.3V.

```
     int Get_Battery_Voltage_ADC(void);   //Gets the battery ADC value
```

Calculate the voltage of batteris and return it.

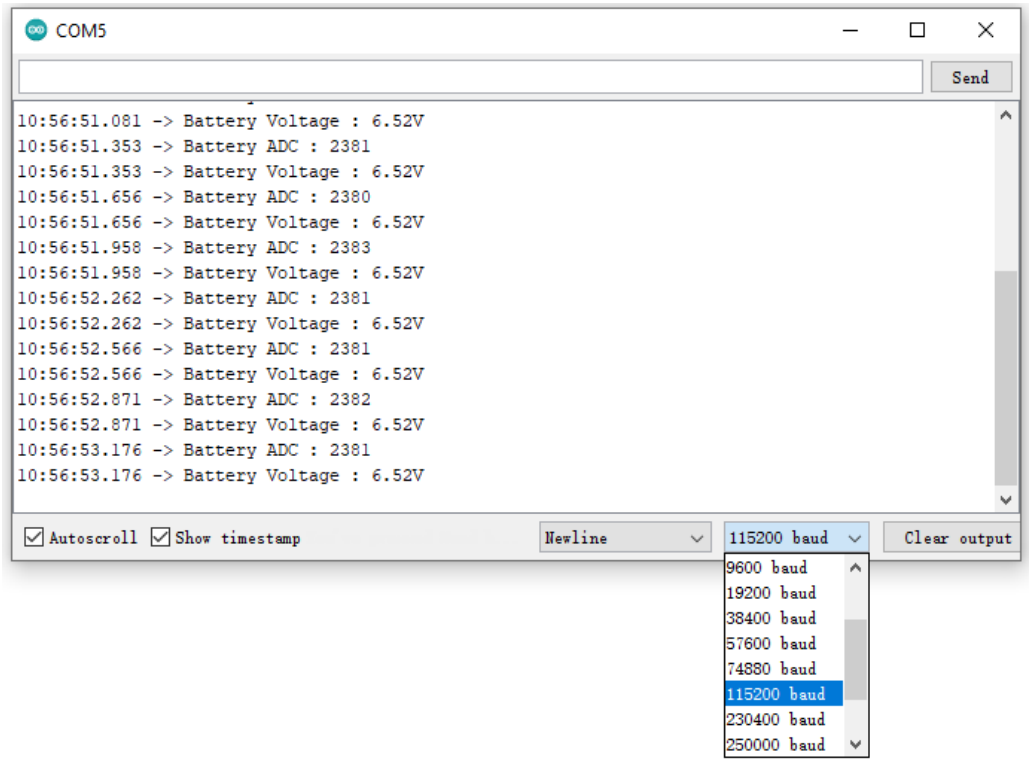
```
     float Get_Battery_Voltage(void);     //Get the battery voltage value
```

The default battery voltage coefficient is 3. Users can modify it by calling this function.

```
void Set_Battery_Coefficient(float coefficient);//Set the partial pressure coefficient
```

Click "Upload" to upload the code to ESP32. After uploading successfully, click Serial Monitor

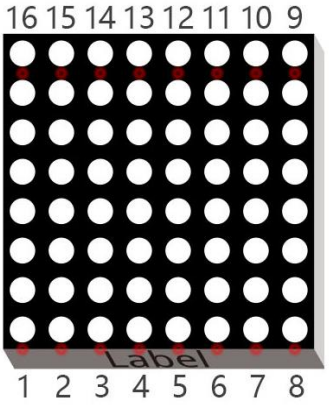
Set baud rate to 115200.



2.5 LED Matrix

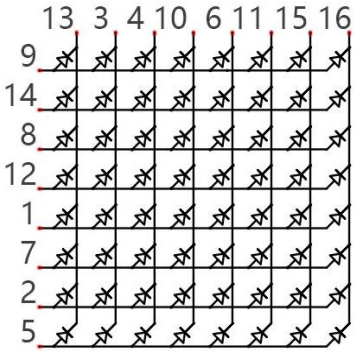
LED Matrix

A LED matrix is a rectangular display module that consists of a uniform grid of LEDs. The following is an 8X8 monochrome LED matrix containing 64 LEDs (8 rows by 8 columns).

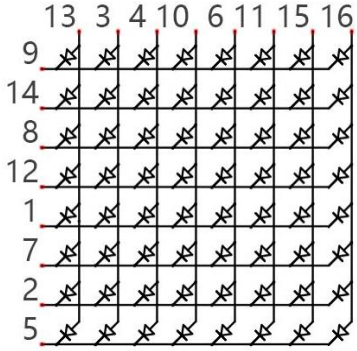


In order to facilitate the operation and reduce the number of ports required to drive this component, the positive poles of the LEDs in each row and negative poles of the LEDs in each column are respectively connected together inside the LED matrix module, which is called a common anode. There is another arrangement type. Negative poles of the LEDs in each row and the positive poles of the LEDs in each column are respectively connected together, which is called a common cathode.

Connection mode of common anode

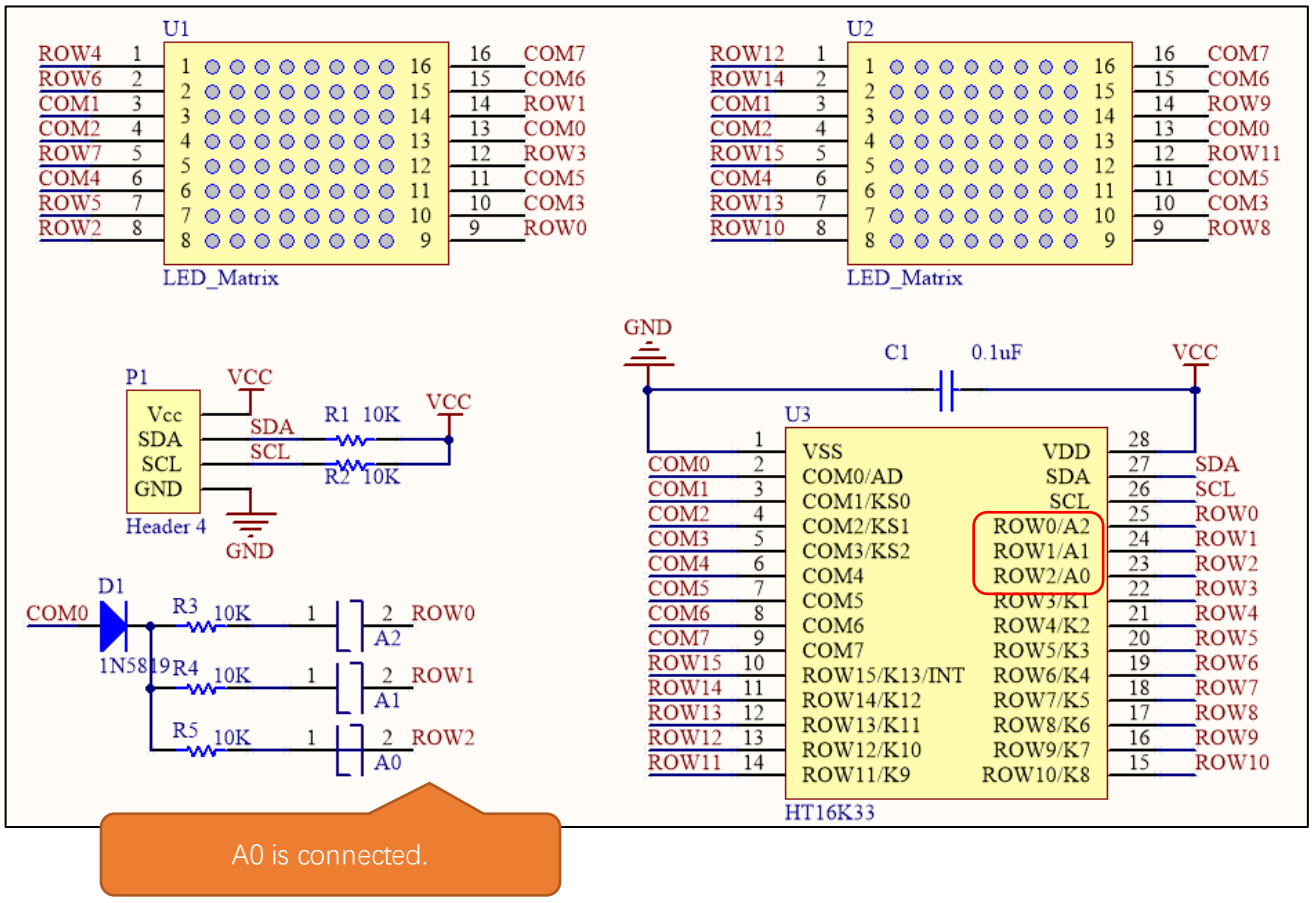


Connection mode of common cathode

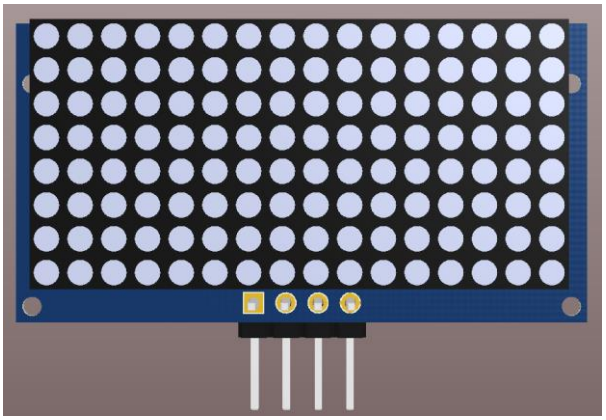
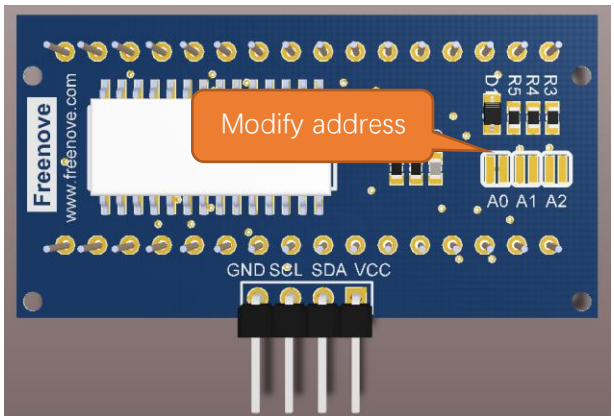


Schematic

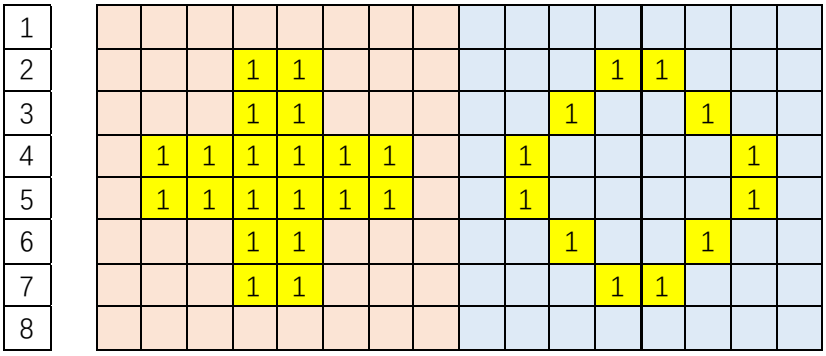
For this tutorial, the LED matrix module is individual and it is driven by IIC chip.



The LED matrix is common anode. As we can see from the schematic above, the anode of LED matrix is connected to ROWx of HT16K33 chip, and the cathode is connected to COMx. The address of HT16K33 chip is (0x70+[A2:A0]), and the default address of LED matrix is 0x71. If you want to change the address, you can use a knife to cut the connecting line in the middle of A0, or connect A1/A2.

Front of LED Matrix	Back of LED Matrix
	

We divide the LED matrix into two sides and display "+" on the left and "o" on the right. As shown below, yellow stands for lit LED while other colors reprensent the OFF LED.



Below, the table on the left corresponds to the "+" above, and the table on the right corresponds to the "o" above.

Row	Binary	Hexadecimal
1	0000 0000	0x00
2	0001 1000	0x18
3	0001 1000	0x18
4	0111 1110	0x7e
5	0111 1110	0x7e
6	0001 1000	0x18
7	0001 1000	0x18
8	0000 0000	0x00

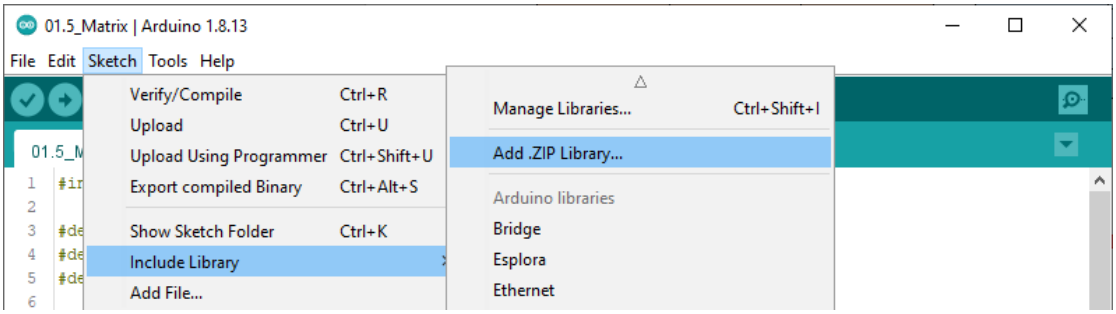
Row	Binary	Hexadecimal
1	0000 0000	0x00
2	0001 1000	0x18
3	0010 0100	0x24
4	0100 0010	0x42
5	0100 0010	0x42
6	0010 0100	0x24
7	0001 1000	0x18
8	0000 0000	0x00

Sketch

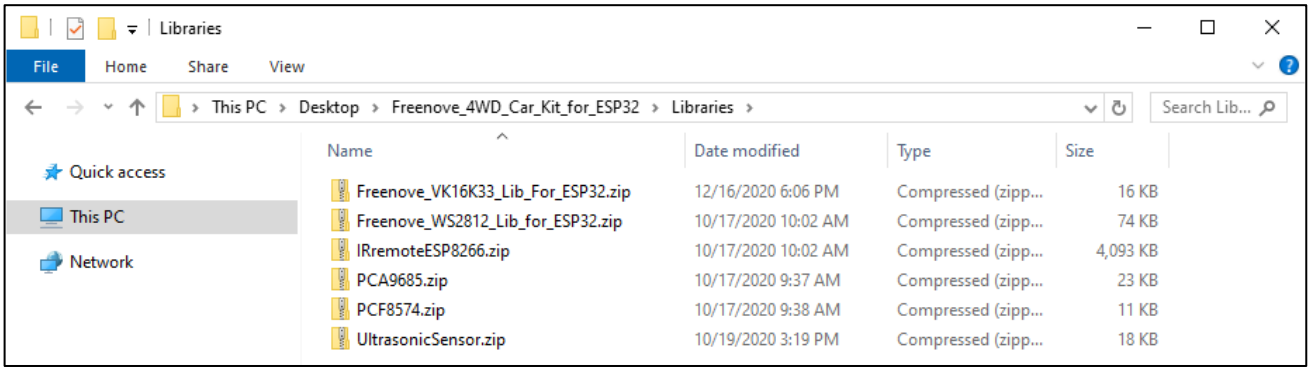
The LED matrix is controlled by HT16K33 chip. Therefore, before opening the program, we need to install Freenove_VK16K33_Lib_For_ESP32 library in advance.

Install Freenove_VK16K33_Lib_For_ESP32 Library

Click Sketch and select Add .ZIP Library in Include.



Select "Freenove_VK16K33_Lib_For_ESP32.zip" in the folder Libraries of the folder "Freenove_4WD_Car_Kit_for_ESP32".



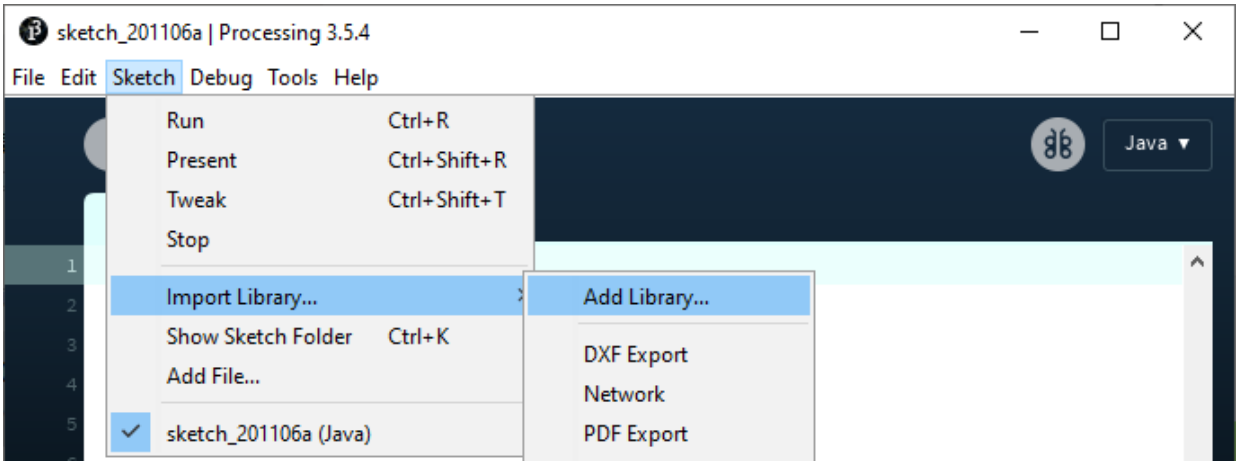
Install Processing

In this tutorial, we use Processing to build a simple Led Matrix platform.

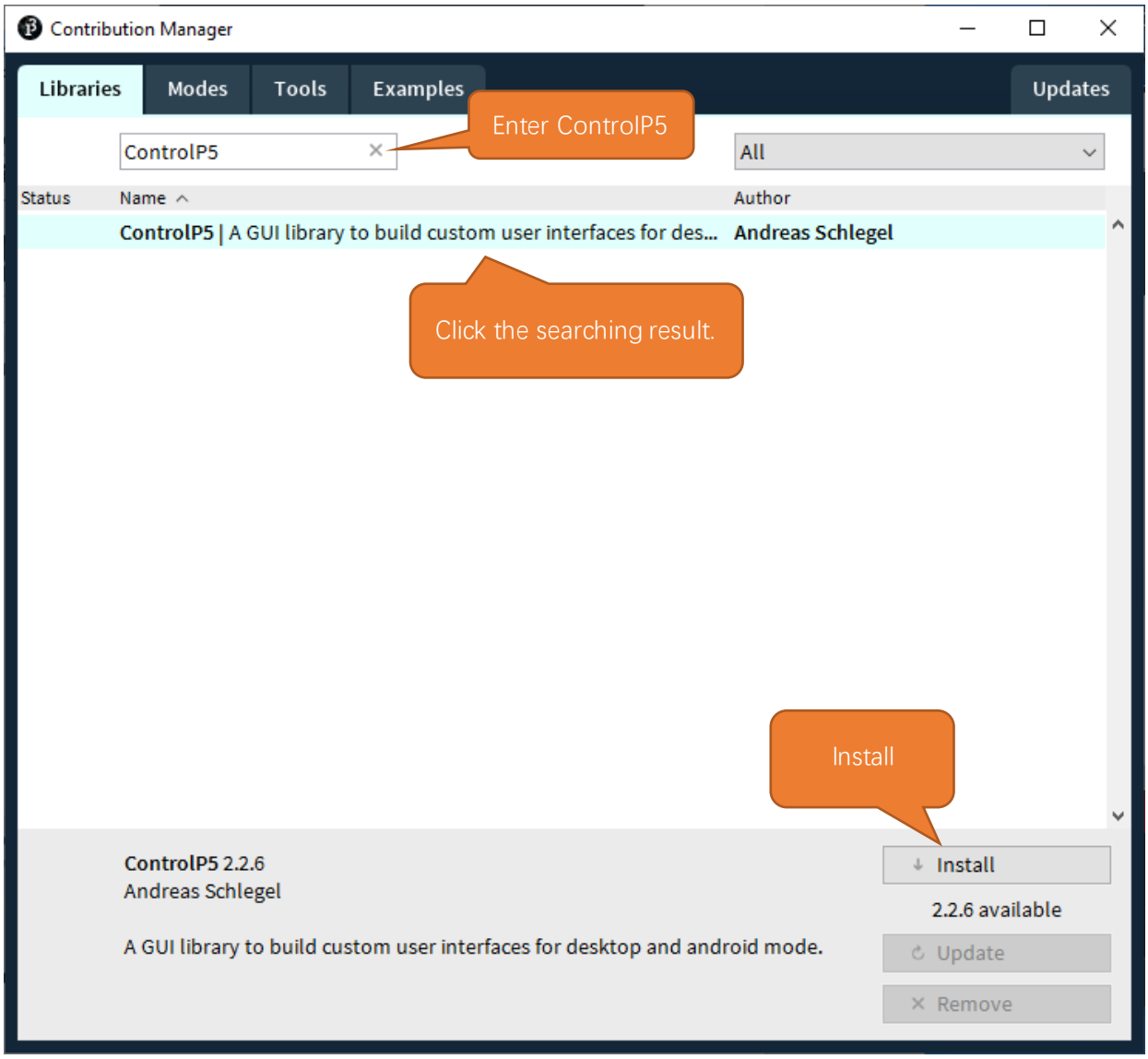
If you have not installed Processing, you can download it by clicking https://processing.org/download/. You can choose an appropriate version to download according to your PC system.

Unzip the downloaded file to your computer. Click "processing.exe" as the figure below to run this software.

In the interface of Processing, click Sketch on Menu bar, select "Import Library..." and then click "Add Library...".

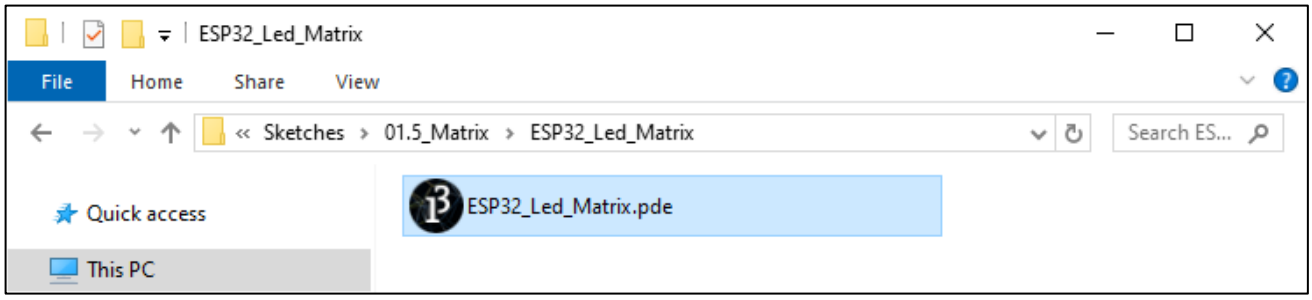


Enter "ControlP5" in the input field of the pop-up window. Click the searching result and then click "install"



When the installation finishes, restart Processing.

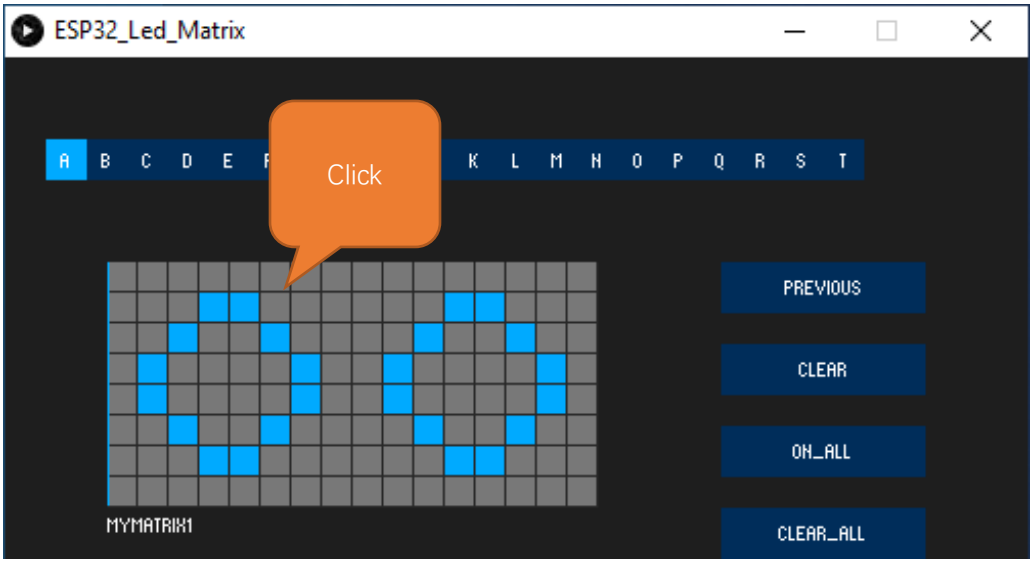
Open the folder ESP32_Led_Matrix in 01.5_Matrix of the **"Freenove_4WD_Car_Kit_for_ESP32\Sketches"**. Here we take Windows as an example. Click to open ESP32_Led_Matrix.pde.



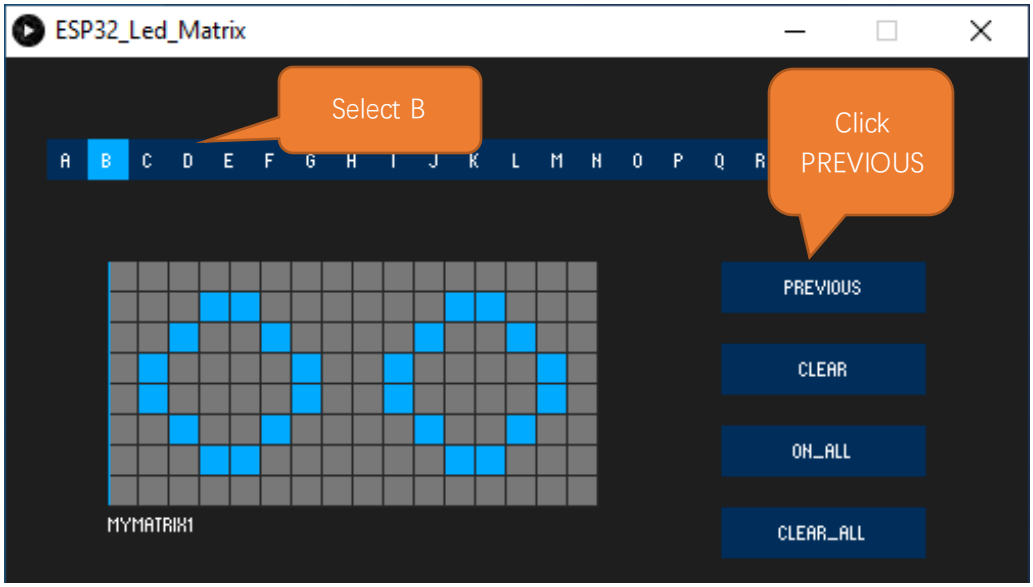
Click "Run"

There are 20 pages from A to T. Select Page A.

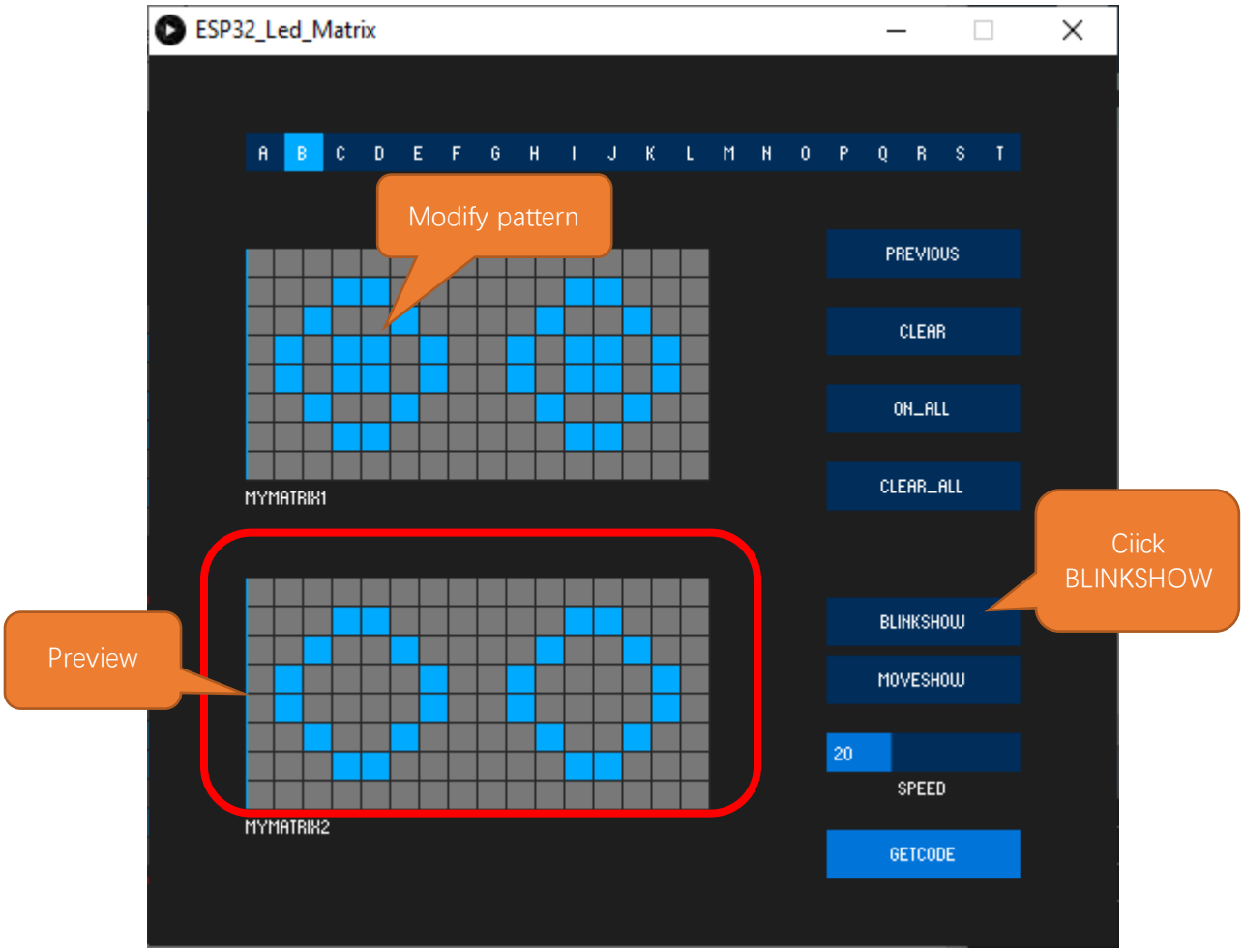
You can design your own pattern by clicking on the squares.



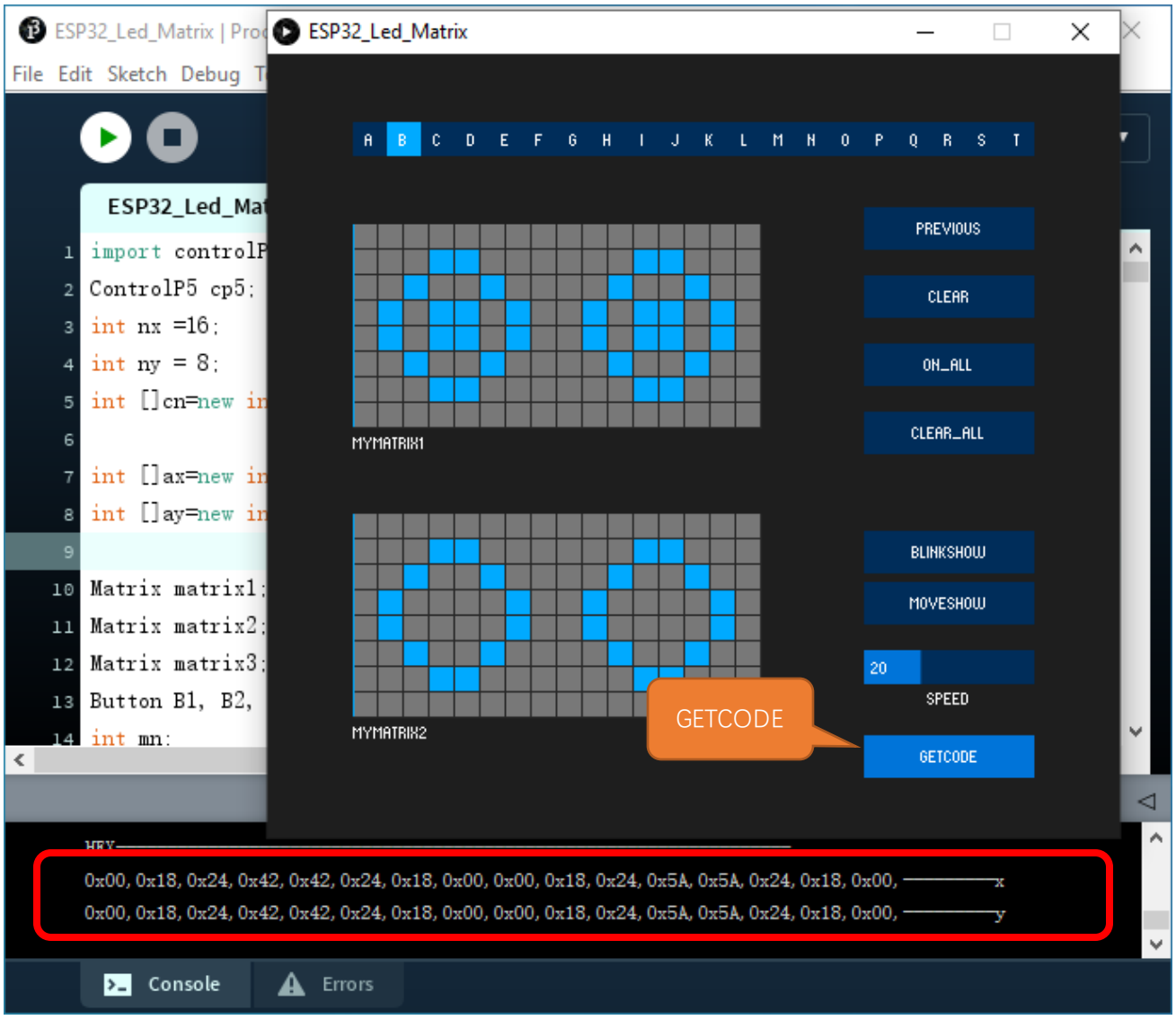
Next select Page B and click PREVIOUS. PREVIOUS will copy the previous pattern to B.



Click the squares to modify the pattern, and then click BLINKSHOW, you can browse the overlay effect of different pages.



Cick GETCODE to generate array.



The data on the left of the LED matrix are stored together and end with "----x", and the data on the right are stored together and end with "----y". Copy these two sets of dot matrix data and replace the array content in "01.5_Matrix.ino".

Open the folder "01.5_Matrix" in the **"Freenove_4WD_Car_Kit_for_ESP32\Sketches"** and double click "01.5_Matrix.ino"

Code

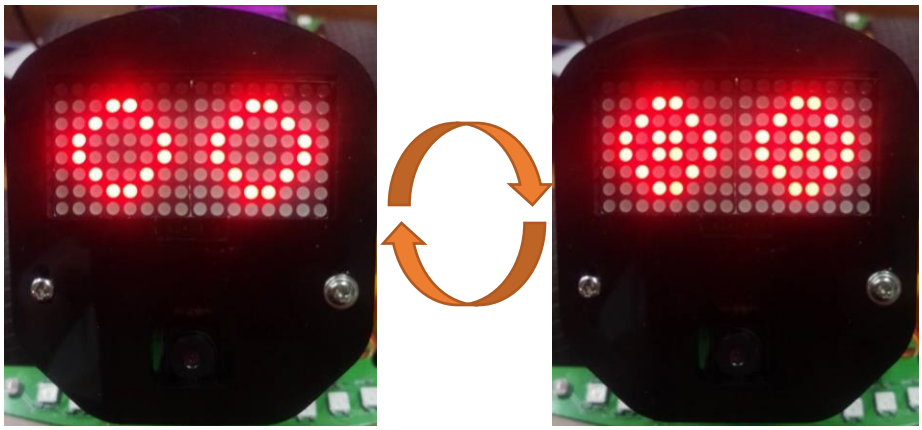
```
#include "Freenove_VK16K33_Lib_For_ESP32.h"

#define EMOTION_ADDRESS 0x70
#define EMOTION_SDA     13
#define EMOTION_SCL     14

Freenove_ESP32_VK16K33 matrix = Freenove_ESP32_VK16K33();

byte x_array[][8] = {//Put the data into the left LED matrix
  ////////////////////////////////////////////
  0x00, 0x18, 0x24, 0x42, 0x42, 0x24, 0x18, 0x00,
  0x00, 0x18, 0x24, 0x5A, 0x5A, 0x24, 0x18, 0x00,
////////////////////////////////////////////
};

byte y_array[][8] = {//Put the data into the right LED matrix
  ////////////////////////////////////////////
  0x00, 0x18, 0x24, 0x42, 0x42, 0x24, 0x18, 0x00,
  0x00, 0x18, 0x24, 0x5A, 0x5A, 0x24, 0x18, 0x00,
////////////////////////////////////////////
};

void setup()
{
  matrix.init(EMOTION_ADDRESS, EMOTION_SDA, EMOTION_SCL);
  matrix.setBlink(VK16K33_BLINK_OFF);
}

void loop()
{
  int count = sizeof(x_array) / sizeof(x_array[0]);
  for (int i = 0; i < count; i++)
  {
    matrix.showStaticArray(x_array[i], y_array[i]);
    delay(500);
  }
}
```

Replace the x array on the left

Replace the y array on the right

Copy the array generated by the auxiliary applet to the program, and then click upload.

You can see the LED matrix keep blinking like eyes.



Code Explanation

Add the header file of LED matrix. Each time before controlling LED matrix, please add its header file first.

```
1   #include "Freenove_VK16K33_Lib_For_ESP32.h"
```

Apply for an Freenove_ESP32_VK16K33 object and name it matrix.

```
7   Freenove_ESP32_VK16K33 matrix = Freenove_ESP32_VK16K33();
```

Define IIC address and IIC pins of HT16K33 chip. Call init() function to initialize it and call setBlink() to set the LED matrix not blink.

```
3   #define EMOTION_ADDRESS 0x70
4   #define EMOTION_SDA     13
5   #define EMOTION_SCL     14
…   ...
25    matrix.init(EMOTION_ADDRESS, EMOTION_SDA, EMOTION_SCL);
26    matrix.setBlink(VK16K33_BLINK_OFF);
```

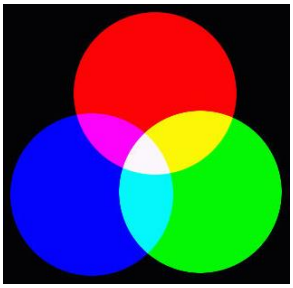
Define count to calculate the number of one-dimensional arrays contained in the two-dimensional x_array, and use the for loop to call the showStaticArray() function to continuously display the content of LED matrix.

```
31    int count = sizeof(x_array) / sizeof(x_array[0]);
32    for (int i = 0; i < count; i++)
33    {
34      matrix.showStaticArray(x_array[i], y_array[i]);
35      delay(500);
36    }
```

2.6 WS2812

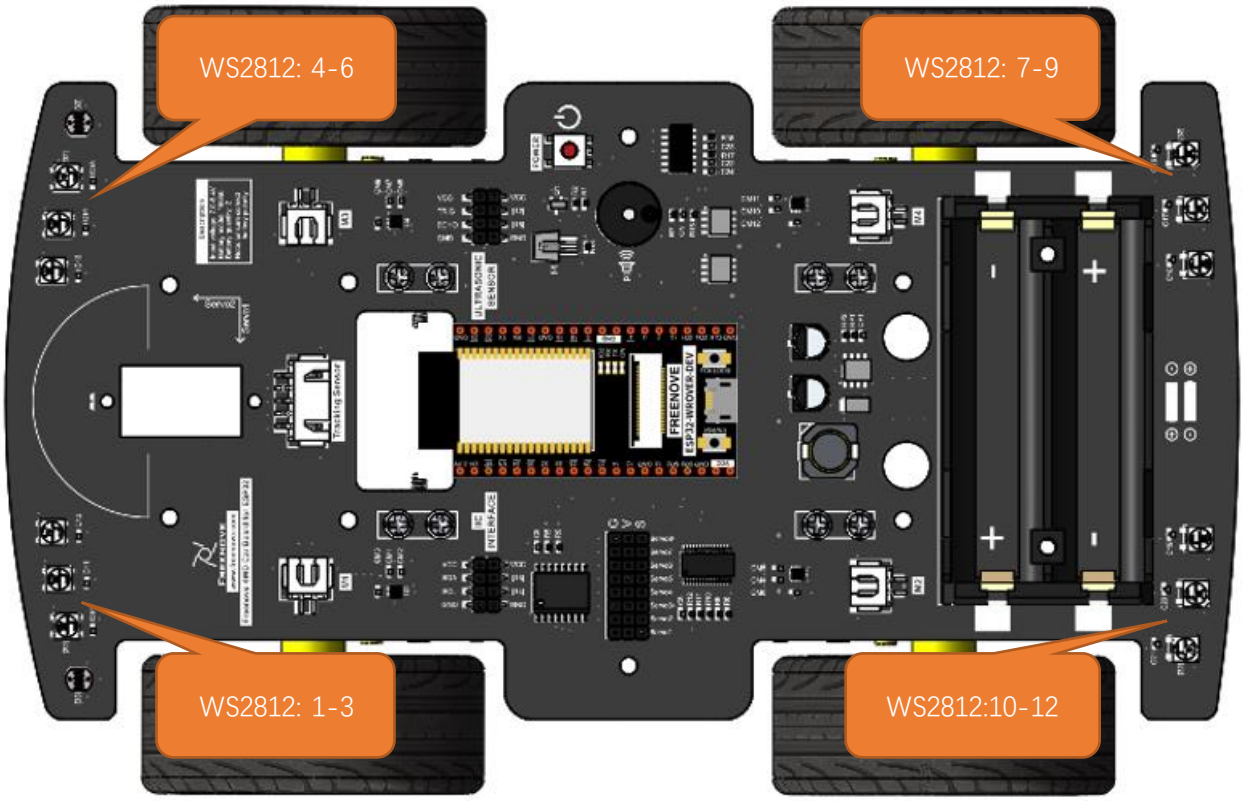
WS2812

Red, green, and blue are called the three primary colors. When you combine these three primary colors of different brightness, it can produce almost all kinds of visible light.



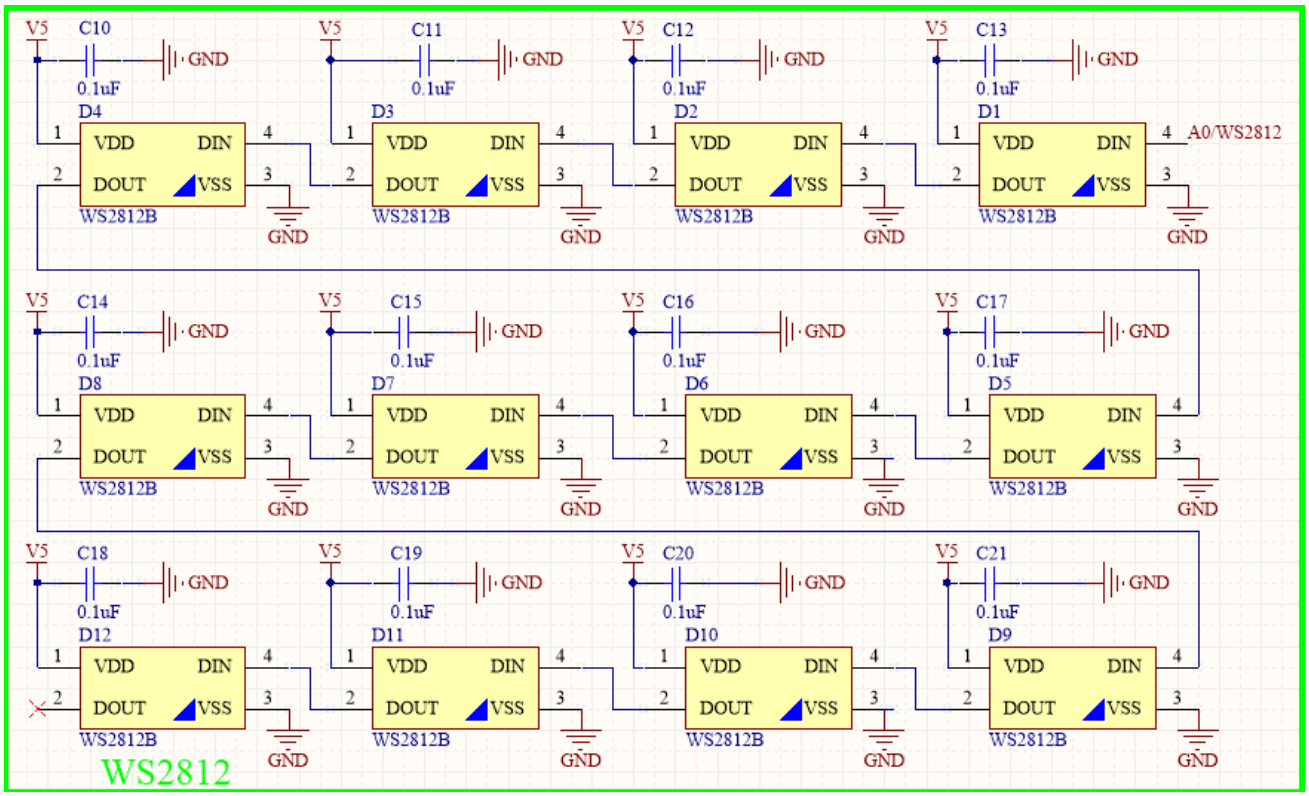
RGB

The LED of the car is composed of 12 WS2812, each of which is controlled by one pin and supports cascading. Each WS2812 can emit three basic colors of red, green and blue, and supports 256-level brightness adjustment, which means that each WS2812 can emit 2^24=16,777,216 different colors.



Schematic

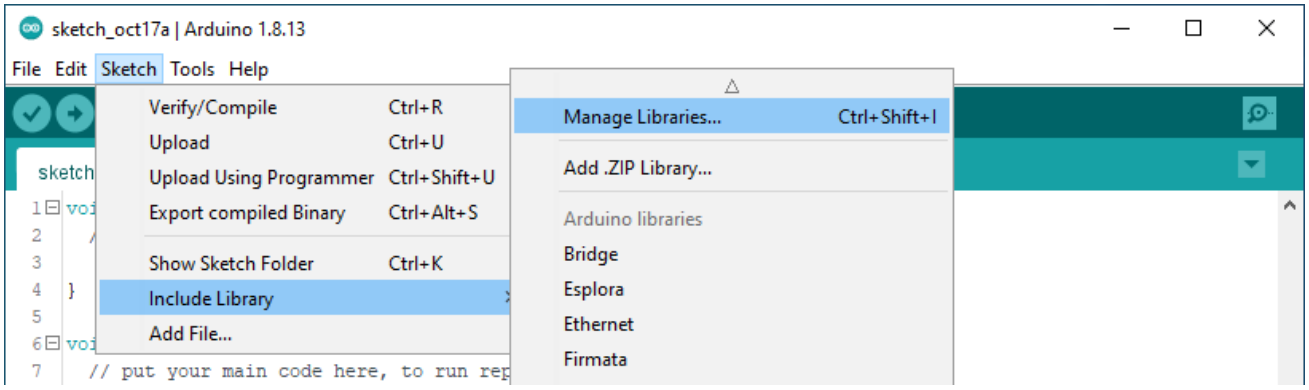
As shown below, the DOUT of each WS2812 is connected with DIN of the next WS2812, and the 12 WS2812 can be controlled to emit colorful colors by inputting control signals through A0/WS2812.



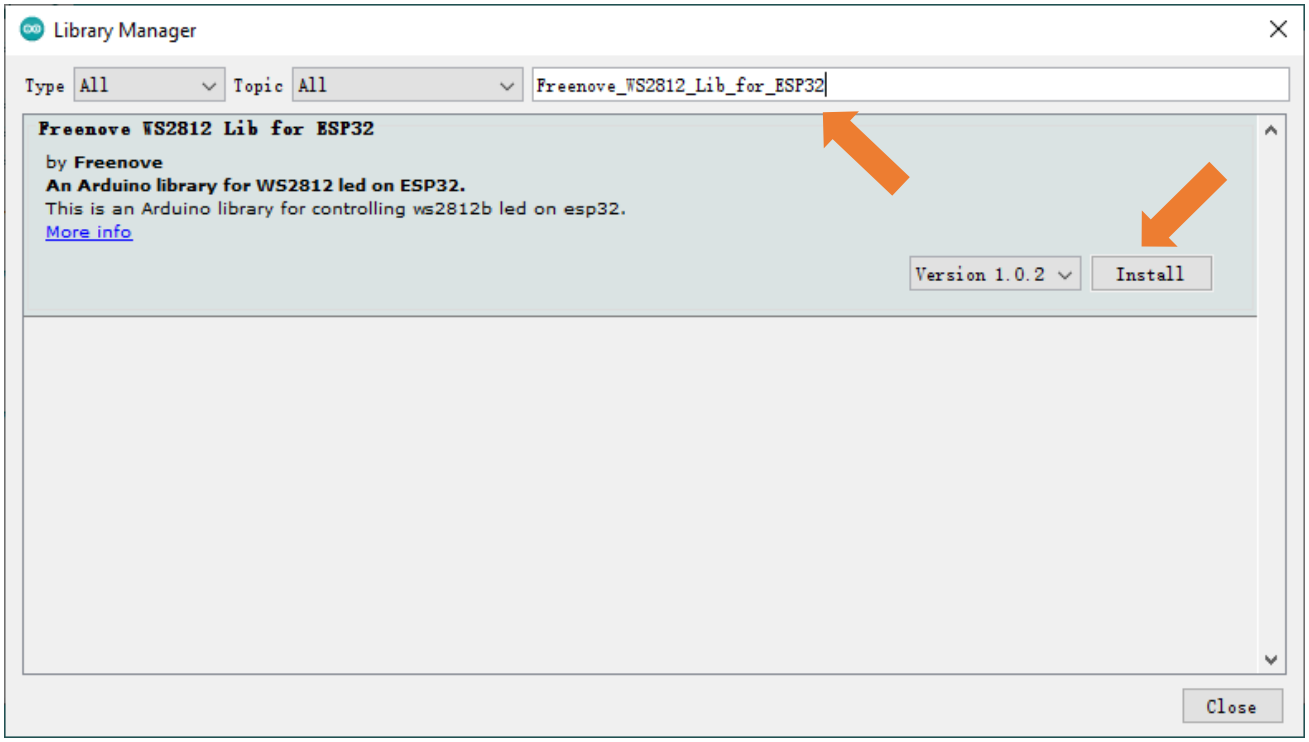
Sketch

Before programming, please make sure the WS2812 driver library has been installed. If not, please install it as follows.

Open Arduino IDE, select Sketch on Menu bar, move the mouse to Include library and click Manage Libraries.

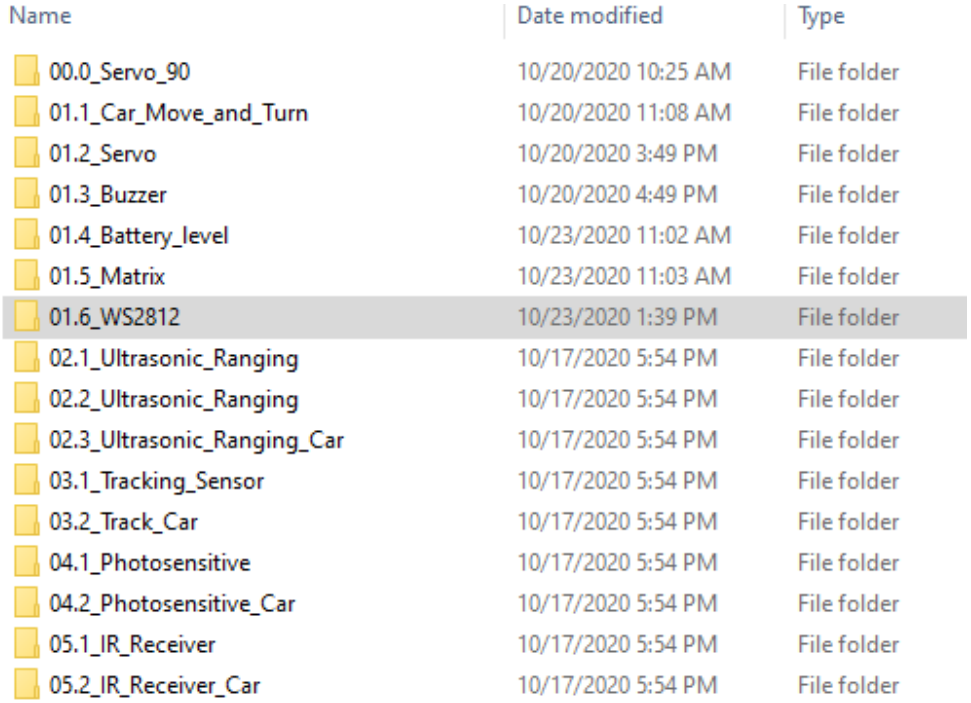


Enter Freenove_WS2812_Lib_for_ESP32 in the input field of the pop-up window, find it and then click Install.



Wait for the installation to finish.

Next we will download the code to ESP32 to test the LED. Open the folder "01.6_WS2812" in the **"Freenove_4WD_Car_Kit_for_ESP32\Sketches"** and then double click "01.6_WS2812.ino".



Code

```
#include "Freenove_WS2812_Lib_for_ESP32.h"

#define LEDS_COUNT  12    //Define the count of WS2812
#define LEDS_PIN    32    //Define the pin number for ESP32
#define CHANNEL     0     //Define the channels that control WS2812
Freenove_ESP32_WS2812 strip = Freenove_ESP32_WS2812(LEDS_COUNT, LEDS_PIN, CHANNEL, TYPE_GRB);

void setup() {
  strip.begin();          //Initialize WS2812
  strip.setBrightness(10);//Set the brightness of WS2812
}

void loop() {
  for (int j = 0; j < 255; j += 2) {
    for (int i = 0; i < LEDS_COUNT; i++) {
      strip.setLedColorData(i, strip.Wheel((i * 256 / LEDS_COUNT + j) & 255));//Set the color
of the WS2812
    }
    strip.show();       //Call WS2812 to display colors
    delay(5);
  }
}
```

Download the code to the ESP32, turn ON the power switch and the WS2812 on the car will emit lights like rainbow.

Code Explanation:

Add the header file of WS2812. Each time before controlling WS2812, please add its header file.

```
#include "Freenove_WS2812_Lib_for_ESP32.h"
```

Set the number of WS2812, define the control pin and channel. Instantiate a WS2812 object and name it strip.

```
#define LEDS_COUNT  12    //Define the count of WS2812
#define LEDS_PIN    32    //Define the pin number for ESP32
#define CHANNEL     0     //Define the channels that control WS2812
Freenove_ESP32_WS2812 strip = Freenove_ESP32_WS2812(LEDS_COUNT, LEDS_PIN, CHANNEL, TYPE_GRB);
```

Initialize WS2812, set their brightness to be 10. The range of brightness is 0-255.

```
  strip.begin();          //Initialize WS2812
  strip.setBrightness(10);//Set the brightness of WS2812
```

Set the color of WS2812. Note: The color will not be displayed immediately after it is set.

```
strip.setLedColorData(i, strip.Wheel((i * 256 / LEDS_COUNT + j) & 255));
```

Display the color of WS2812. After setting the color, you need to call show function to display it.

```
strip.show();
```

Chapter 3 Ultrasonic Obstacle Avoidance Car

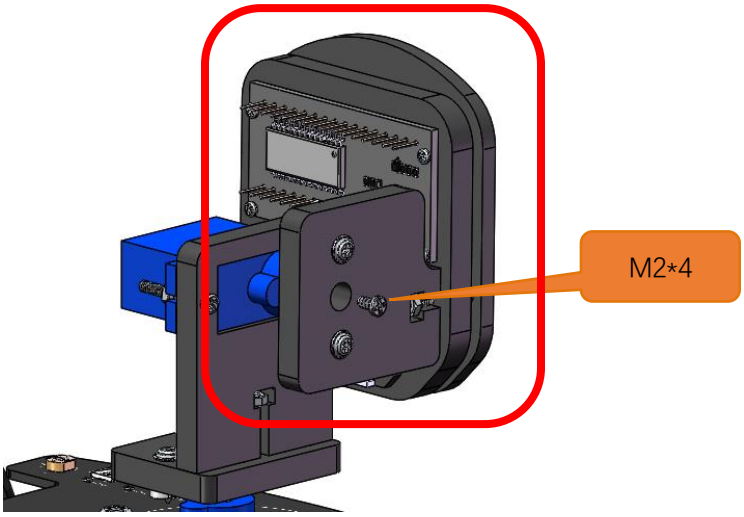
This chapter requires to replace the head pan-tilt. Please follow the instructions below to replace the head of the car with an ultrasonic head before use.

In the whole tutorial, only Chapter 3 uses the ultrasonic pan-tilt. Please replace the head pan-tilt back to the LED matrix after finishing this chapter.

3.1 Ultrasonic Module

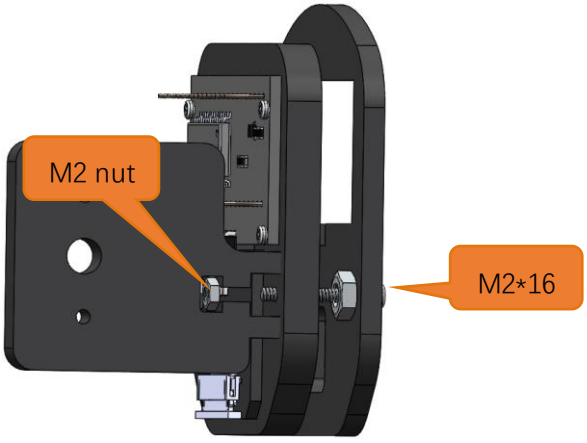
Replace Pan-tilt

Step 1



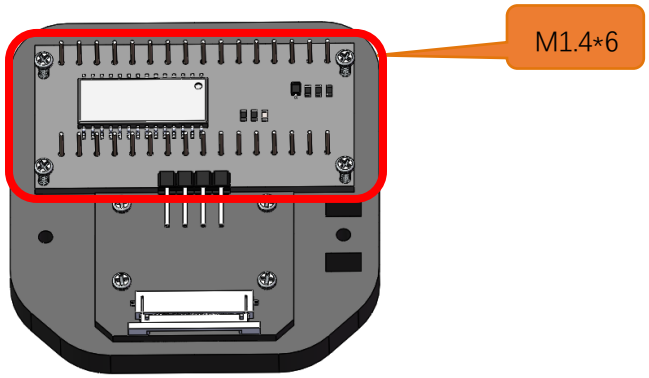
Remove M2*4 screws to uninstall the pan-tilt.

Step 2



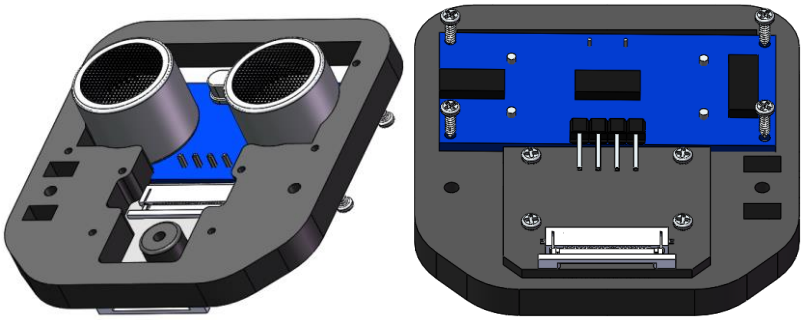
Remove the M2*16 screws and M2 nuts on both sides of the Pan-tilt.

Step 3



Remove the 4 M1.4*6 screws on the LED matrix module.

Step 4

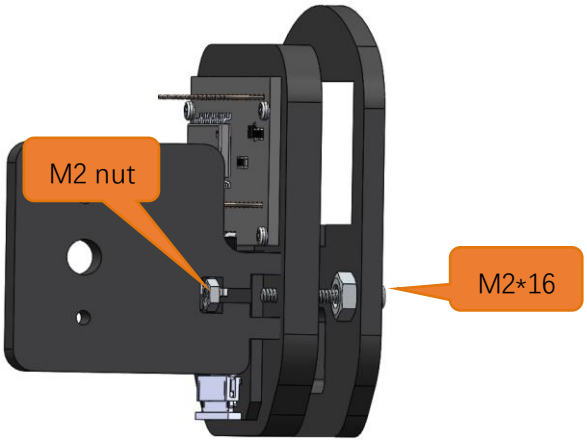


Use 4 M1.4*6 screws to fix the ultrasonic module to the Pan-tilt.

Step 5

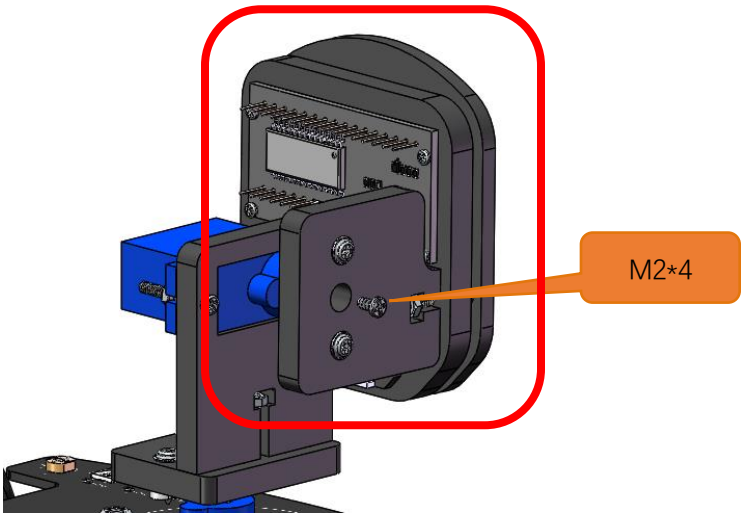
Install M2*16 screws and M2 nuts on both sides of the Pan-tilt.

Step 6



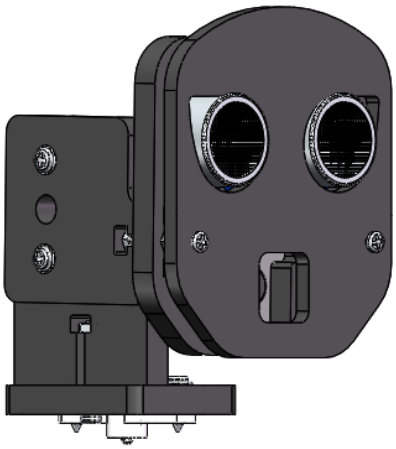
Use 1 M2*16 and 1 M2 nut to fix the three acrylic boards together.

Step 7

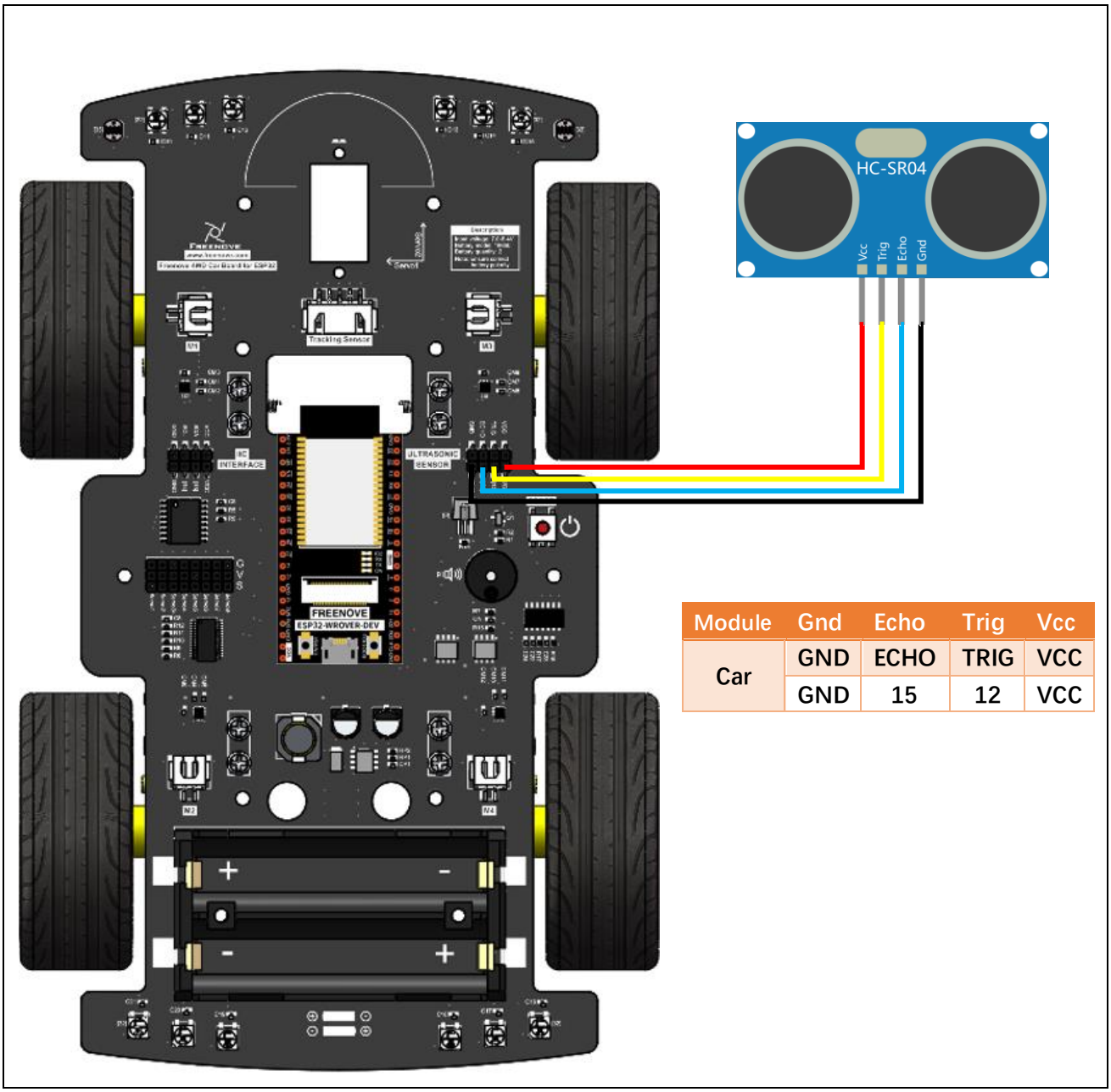


Use M2*4 screws to fix the ultrasonic pan-tilt to the servo motor 2. Please note that you need to adjust the servo motor to a position of 90 degrees before fixing the pan-tilt to it.

After finished



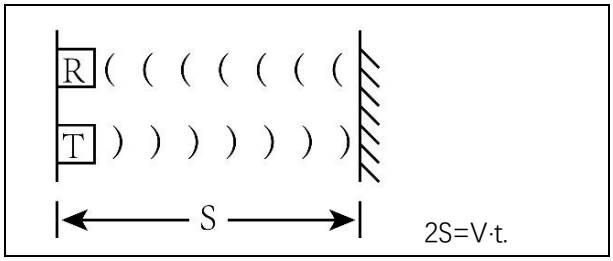
Wiring of the Pan-tilt



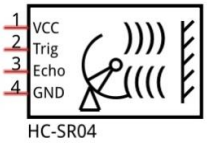
Module	Gnd	Echo	Trig	Vcc
Car	GND	ECHO	TRIG	VCC
	GND	15	12	VCC

Ultrasonic Module

The ultrasonic ranging module uses the principle that ultrasonic waves will be sent back when encounter obstacles. We can measure the distance by counting the time interval between sending and receiving of the ultrasonic waves, and the time difference is the total time of the ultrasonic wave's journey from being transmitted to being received. Because the speed of sound in air is a constant, about v=340m/s, we can calculate the distance between the ultrasonic ranging module and the obstacle: s=vt/2.



The HC-SR04 ultrasonic ranging module integrates both an ultrasonic transmitter and a receiver. The transmitter is used to convert electrical signals (electrical energy) into high frequency (beyond human hearing) sound waves (mechanical energy) and the function of the receiver is opposite of this. The picture and the diagram of the HC SR04 ultrasonic ranging module are shown below:



Pin description:

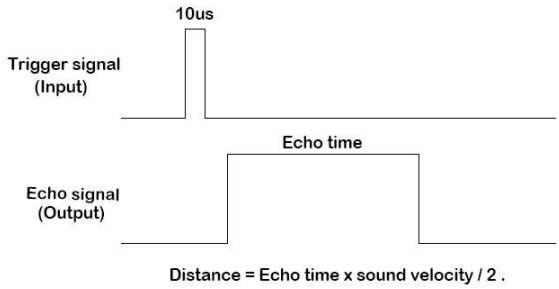
Pin	Description
VCC	Power supply pin
Trig	Trigger pin
Echo	Echo pin
GND	GND

Technical specs:

Working voltage: 5V Working current: 12mA

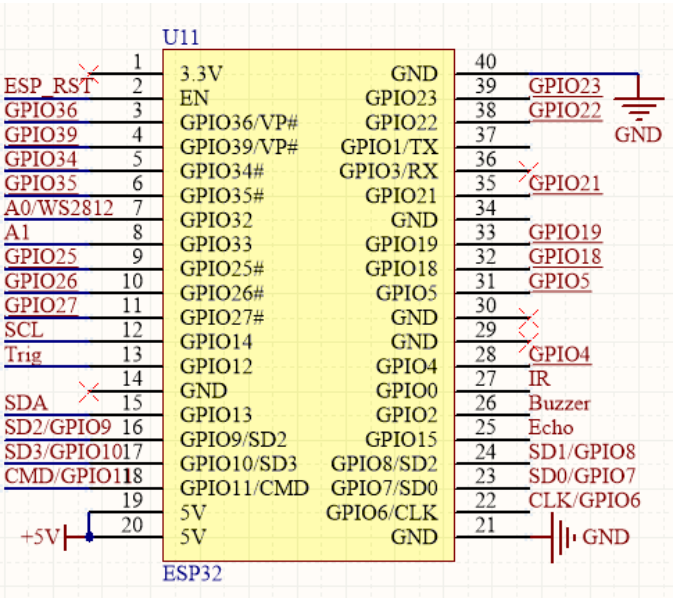
Minimum measured distance: 2cm Maximum measured distance: 200cm

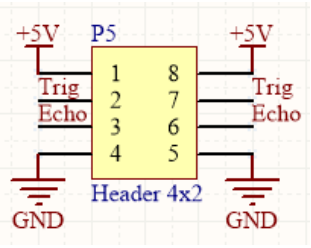
Instructions for use: output a high-level pulse in Trig pin lasting for least 10us, the module begins to transmit ultrasonic waves. At the same time, the Echo pin is pulled up. When the module receives the returned ultrasonic waves from encountering an obstacle, the Echo pin will be pulled down. The duration of high level in the Echo pin is the total time of the ultrasonic wave from transmitting to receiving, s=vt/2.



Schematic

The ultrasonic module is located at the front of the car and is connected to the ESP32 development board by means of wiring. As can be seen from the figure below, ESP32 uses GPIO12 and GPIO15 to control the Trig and Echo pins of the ultrasonic module.





Sketch

When the power of the car is turned ON, every module will be initialized and the servo motor will rotate to 90°. The ultrasonic data will be obtained and printed through serial port as the servo motor rotates.

Open the folder "02.1_Ultrasonic_Ranging" in **"Freenove_4WD_Car_Kit_for_ESP32\Sketches"** and double click "02.1_Ultrasonic_Ranging.ino"

Code

```
#include <Arduino.h>
#include "Freenove_4WD_Car_For_ESP32.h"

void setup() {
  Serial.begin(115200);//Open the serial port and set the baud rate to 115200
  Ultrasonic_Setup();  //Ultrasonic module initialization
  PCA9685_Setup();       //Servo motor initialization
  Servo_1_Angle(90);   //Set the initial value of Servo 1 to 90 degrees
  Servo_2_Angle(90);   //Set the initial value of Servo 2 to 90 degrees
  delay(500);          //Wait for the servo to arrive at the specified location
}

void loop() {
  Servo_1_Angle(150);  //Turn servo 1 to 150 degrees
  Serial.print("Distance: " + String(Get_Sonar()) + "\n");//Print ultrasonic distance
```

```
16    delay(500);
17
18    Servo_1_Angle(90);   //Turn servo 1 to 90 degrees
19    Serial.print("Distance: " + String(Get_Sonar()) + "\n");//Print ultrasonic distance
20    delay(500);
21
22    Servo_1_Angle(30);   //Turn servo 1 to 30 degrees
23    Serial.print("Distance: " + String(Get_Sonar()) + "\n");//Print ultrasonic distance
24    delay(500);
25
26    Servo_1_Angle(90);   //Turn servo 1 to 90 degrees
27    Serial.print("Distance: " + String(Get_Sonar()) + "\n");//Print ultrasonic distance
28    delay(500);
29  }
```

Code Explanation:

Initialize ultrasonic module and PCA9685 chip.

```
6     Ultrasonic_Setup();  //Ultrasonic module initialization
7     PCA9685_Setup();       //Servo motor initialization
```

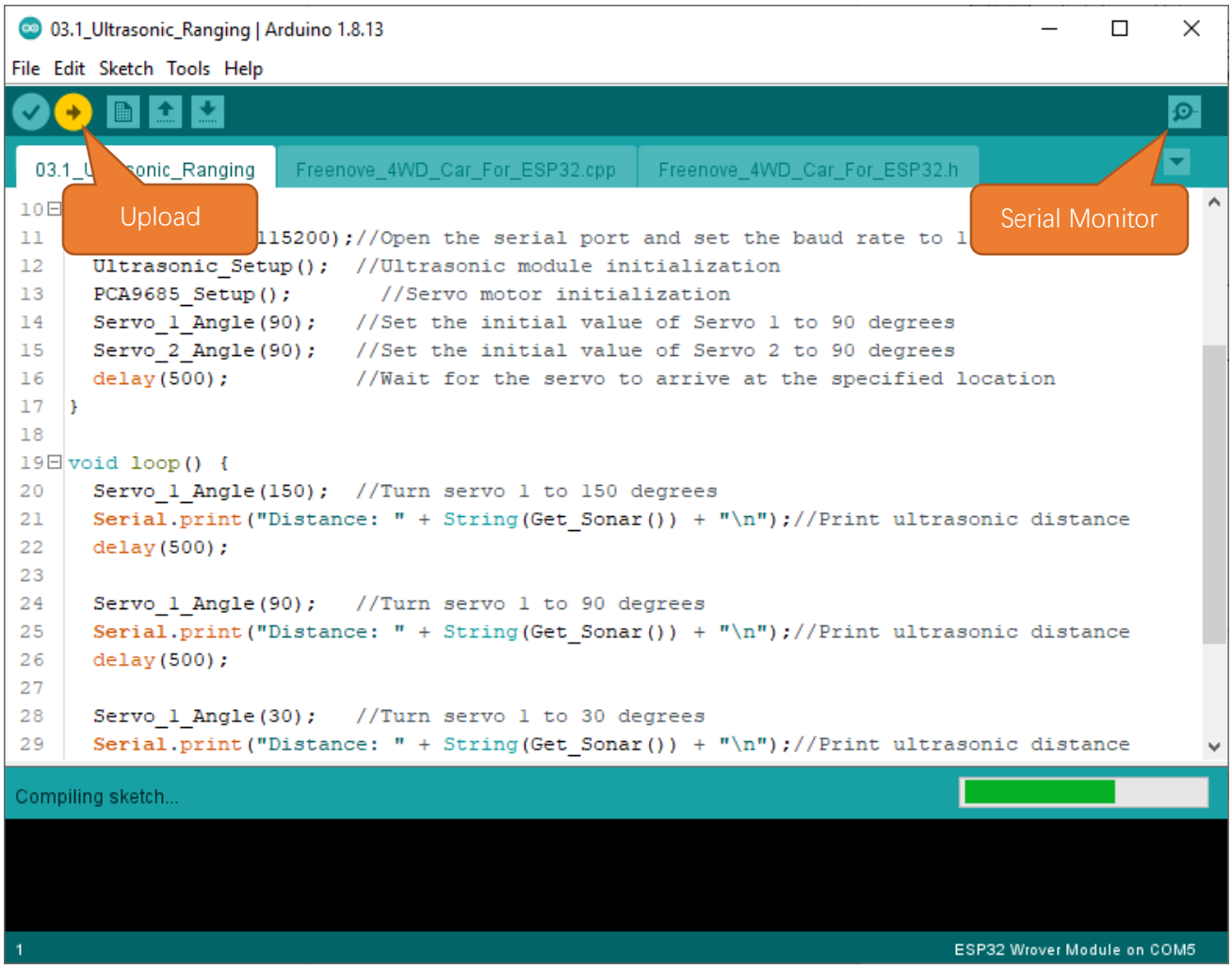
Obtain the distance between ultrasonic module and the obstacle and return a real number data in cm.

```
15    Get_Sonar()
```

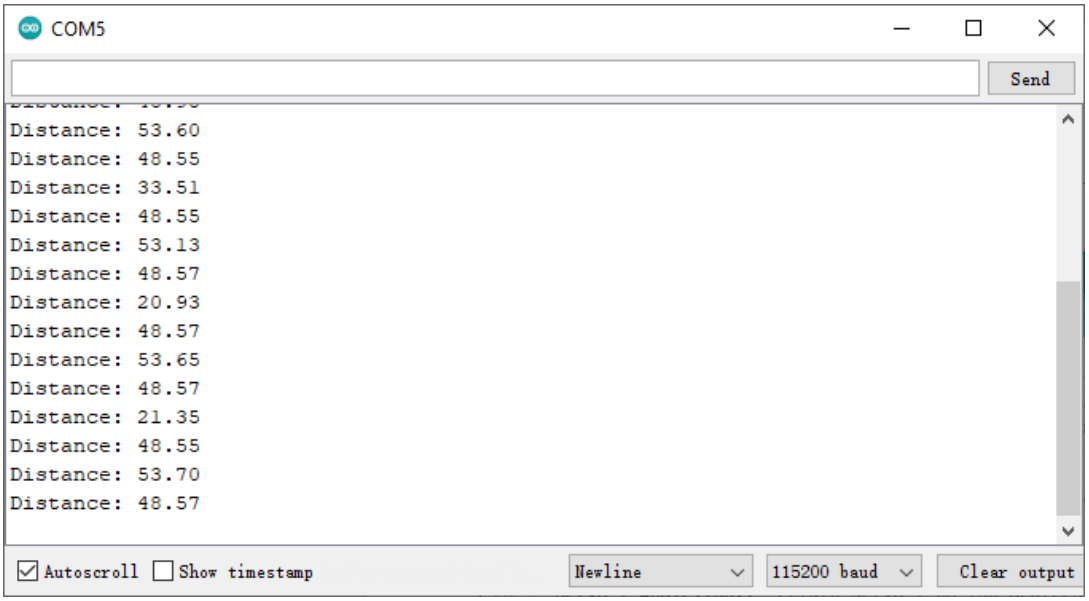
Rotate the angle of the servo motor 1 and cooperate with the ultrasonic module to obtain distance data.

```
14    Servo_1_Angle(150);  //Turn servo 1 to 150 degrees
...   ...
18    Servo_1_Angle(90);   //Turn servo 1 to 90 degrees
...   ...
22    Servo_1_Angle(30);   //Turn servo 1 to 30 degrees
...   ...
26    Servo_1_Angle(90);   //Turn servo 1 to 90 degrees
```

Click “Upload” to upload the code to ESP32. After uploading successfully, click Serial Monitor.



Set the baud rate as 115200.



3.2 Ultrasonic car

After starting the car, the ultrasound acquires data in various directions, makes judgments based on the data in each direction, and controls the car to avoid obstacles.

Sketch

Open the folder "02.2_Ultrasonic_Ranging_Car"in **"Freenove_4WD_Car_Kit_for_ESP32\Sketches"** and double click "02.2_Ultrasonic_Ranging_Car.ino".

Code

```
#include <Arduino.h>
#include "Freenove_4WD_Car_For_ESP32.h"

#define OBSTACLE_DISTANCE      40
#define OBSTACLE_DISTANCE_LOW  20
int distance[4];     //Storage of ultrasonic data

void setup() {
  Ultrasonic_Setup();//Initialize the ultrasonic module
  PCA9685_Setup();   //PCA9685 chip initializes
}

void loop()
{
  get_distance(1);//Get multiple Angle distance data
  //There is no obstacle within 40cm of the front of the car
  if (distance[1] > OBSTACLE_DISTANCE){
    //There is an obstacle on the right
    if (distance[0] >= distance[2] && distance[2] < OBSTACLE_DISTANCE_LOW)
      Motor_Move(-1000, -1000, 2000, 2000);
    //There is an obstacle on the left
    else if (distance[0] < distance[2] && distance[0] < OBSTACLE_DISTANCE_LOW)
      Motor_Move(2000, 2000, -1000, -1000);
    else //There are no obstacles around
      Motor_Move(1000, 1000, 1000, 1000);
    delay(20);
  }
  else if (distance[1] < OBSTACLE_DISTANCE){ //There is an obstacle ahead
    Motor_Move(0, 0, 0, 0);                  //Stop the car to judge the situation
    get_distance(2);
    if (distance[1] < OBSTACLE_DISTANCE_LOW) //The car got too close to the obstacle
      Motor_Move(-1000, -1000, -1000, -1000);
```

```
33      else if (distance[0] < distance[2])      //There is also an obstacle on the left
34        Motor_Move(2000, 2000, -2000, -2000);
35      else if (distance[0] > distance[2])      //There is also an obstacle on the right
36        Motor_Move(-2000, -2000, 2000, 2000);
37      delay(200);
38    }
39  }
40
41  //Get distance values for different angles
42  void get_distance(int car_mode)
43  {
44    int add_angle=30;
45    if(car_mode==1)
46      add_angle=0;
47    else
48      add_angle=30;
49    Servo_2_Angle(90);
50    Servo_1_Angle(120+add_angle);
51    delay(100);
52    distance[0] = Get_Sonar();//Get the data on the left
53    Servo_1_Angle(90);
54    delay(100);
55    distance[1] = Get_Sonar();//Get the data in the middle
56    Servo_1_Angle(60-add_angle);
57    delay(100);
58    distance[2] = Get_Sonar();//Get the data on the right
59    Servo_1_Angle(90);
60    delay(100);
61    distance[1] = Get_Sonar();//Get the data in the middle
62  }
```

Code Explanation:

Define an array to store the distance data of the front left, middle, and front right of the car

```
6   int distance[4];     //Storage of ultrasonic data
```

The function to get the distance between obstacles and cars. When car_mode is 1, the scanning range is 60-120 degrees; when car_mode is 2, the scanning range is 30-150 degrees.

```
41  //Get distance values for different angles
42  void get_distance(int car_mode)
…   {...}
```

Control the movement function of the car, the parameter range is -4095-4095. When the parameter is a positive number, the motor rotates forward, and vice versa. The greater the absolute value of the parameter, the more powerful the car will be.

```
        Motor_Move(M1, M2, M3, M4)
```

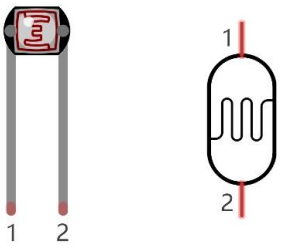
Chapter 4 Light Tracing Car

4.1 Photoresistor ADC

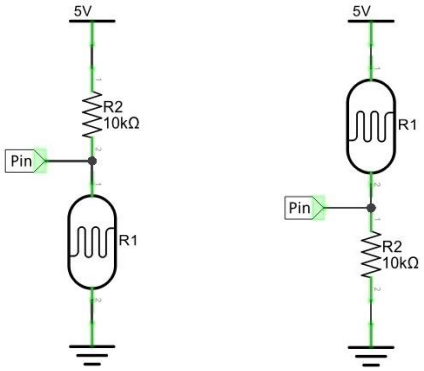
The photoresistor is very sensitive to the amount of light present. We can use this feature to make a light-tracing car. The car is controlled to trun toward the light source by reading the ADC values of the two photoresistors at the head of the car. Before we start, let us learn how to read the photoresistor value.

Photoresistor

A photoresistor is simply a light sensitive resistor. It is an active component that decreases resistance with respect to receiving luminosity (light) on the component's light sensitive surface. A photoresistor's resistance value will change in proportion to the ambient light detected. With this characteristic, we can use a photoresistor to detect light intensity. The photoresistor and its electronic symbol are as follows.



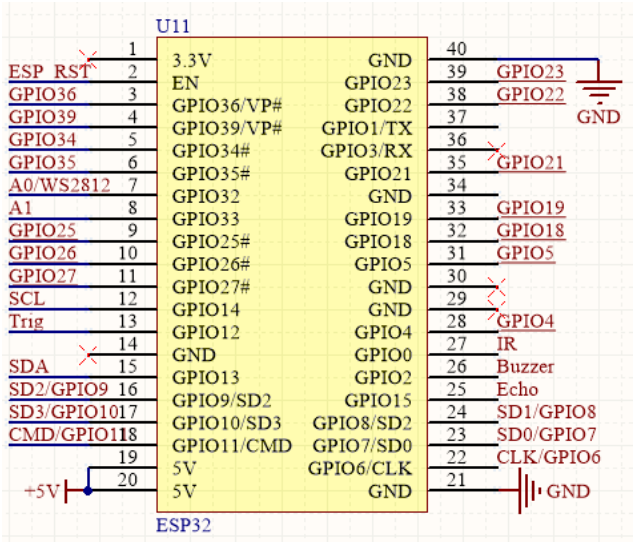
The circuit below is used to detect the change of a photoresistor's resistance value:

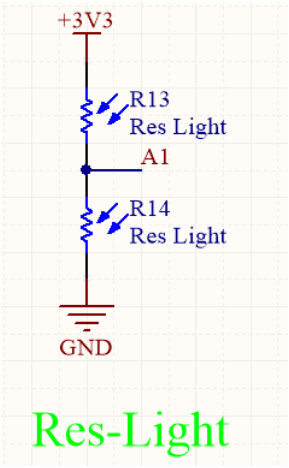


In the above circuit, when a photoresistor's resistance vale changes due to a change in light intensity, the voltage between the photoresistor and resistor R1 will also change. Therefore, the intensity of the light can be obtained by measuring this voltage.

Schematic

The photoresistors are distributed on both sides of the car's head, and the ADC value collection range of A1 is 0-4095. It can be seen from the circuit that when the brightness of the light received by the photoresistor R13 and R14 is the same, the voltage at A1 is 3.3/2*4096, which is 2048. Therefore, when the brightness of the light received by the two photoresistors is different, the value read by the ADC will be greater or less than this value.





Sketch

Next we download the code to ESP32 to test the photoresistors. Open the folder "03.1_Photosensitive" in **"Freenove_4WD_Car_Kit_for_ESP32\Sketches"** and double click "03.1_Photosensitive.ino"

Code

```
#define PHOTOSENSITIVE_PIN 33 //Define the pins that ESP32 reads photosensitive
int photosensitiveADC;        //Defines a variable to store ADC values

void setup()
{
  pinMode(PHOTOSENSITIVE_PIN, INPUT);//Configure the pins for input mode
  Serial.begin(115200); //Initialize the serial port and set the baud rate to 115200
}

void loop()
{
  photosensitiveADC = analogRead(PHOTOSENSITIVE_PIN);//Read the photosensitive resistance value
  Serial.print("photosensitiveADC: ");
  Serial.println(photosensitiveADC);                  //Print photosensitive resistance value
  delay(500);
}
```

Code Explanation:

Define the pin to read photoresistors.

```
1  #define PHOTOSENSITIVE_PIN 33 //Define the pins that ESP32 reads photosensitive
```

Call analogRead to read the differential value of the photoresistors and store it in the photosensitiveADC.

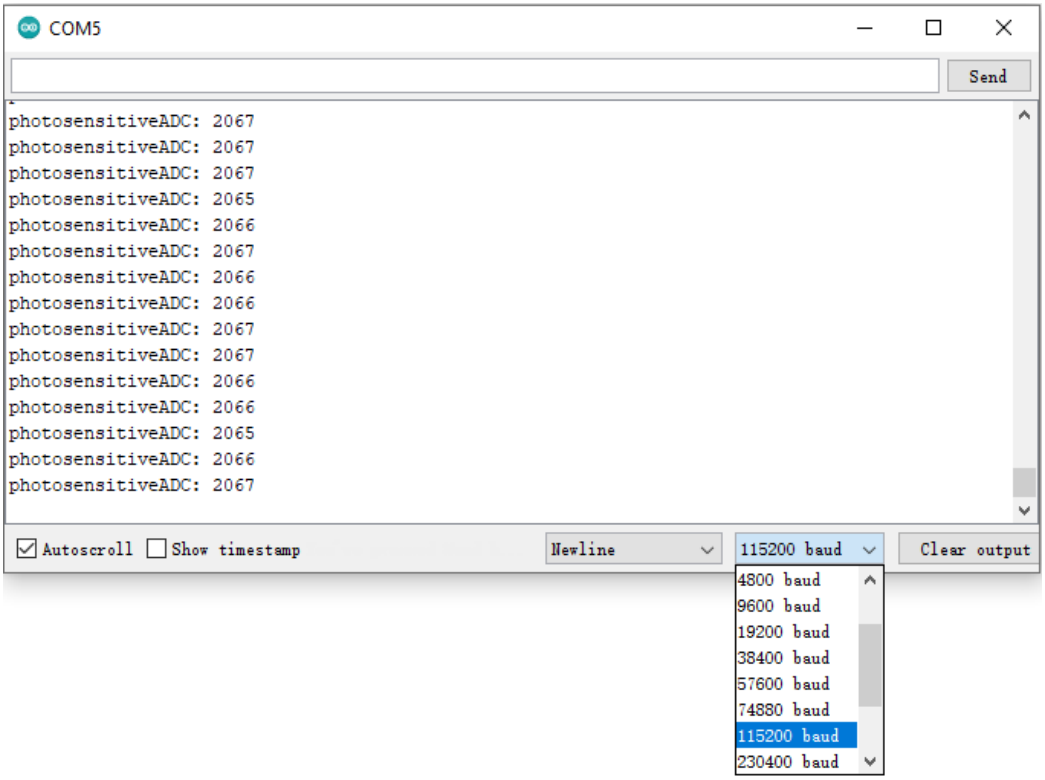
```
12  photosensitiveADC = analogRead(PHOTOSENSITIVE_PIN);//Read the photosensitive resistance value
```

Print the photoresistor value through the serial port.

```
13    Serial.print("photosensitiveADC: ");
14    Serial.println(photosensitiveADC);                    //Print photosensitive resistance value
```

Click "Upload" to upload the code to ESP32. When finishes uploading, click Serial Monitor.

Set baud rate as 115200.

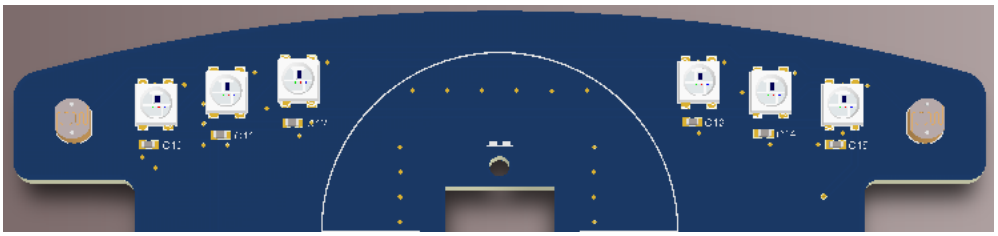


4.2 Light Tracing Car

When light shines on R13, the resistance value of R13 becomes smaller, and the ADC value of photoresistor becomes larger; when light shines on R14, the resistance value of R14 becomes smaller, and the ADC value of photoresistor becomes smaller. Based on that, when the ADC value is detected to be less than that, it means that the light source is on the left of the car, and when the ADC value is greater than that, it means that the light source is on the right of the car.

R14

R13



Sketch

When car is powered ON, the ADC value of the current environment will be obtained. After initialization, the buzzer will sound once to remind users to test the light-tracing function. When users approach the car with light source the car, the car will turn with the light source.

Open the folder "03.2_Photosensitive_Car" in **"Freenove_4WD_Car_Kit_for_ESP32\Sketches"** and double click "03.2_Photosensitive_Car.ino"

Code

```
#include <Arduino.h>
#include "Freenove_4WD_Car_For_ESP32.h"

int photosensitive_sensitivity = 100;//Set the sensitivity of the photosensitive
int photosensitive_init_value = 0;   //Set the car's initial environment ADC value

void setup(){
  Emotion_Setup();          //Emotion initialization
  Buzzer_Setup();           //Buzzer initialization
  Photosensitive_Setup();   //Photosensitive initialization
  PCA9685_Setup();          //Initialize PCA9685 to control motor
  photosensitive_init_value = Get_Photosensitive();
  Buzzer_Alert(1,1);        //The buzzer beeps to prompt the user to start playing
}

void loop(){
  //There is a light source on the left side of the car
  if (Get_Photosensitive() < (photosensitive_init_value - photosensitive_sensitivity)){
    wheel(2,100);
    Motor_Move(-2000, -2000, 2000, 2000);
```

```
21      }
22      //There is a light source on the right side of the car
23      else if (Get_Photosensitive() > (photosensitive_init_value + photosensitive_sensitivity)){
24        wheel(1,100);
25        Motor_Move(2000, 2000, -2000, -2000);
26      }
27      //The light is in the middle of the car
28      else{
29        eyesBlink(100);
30        Motor_Move(0, 0, 0, 0);
31      }
32  }
```

Code Explanation:

Set the sensitivity of the photoresistor. You can adjust the value according to the change of the environment. The value is recommended to be between 30-300.

```
4   int photosensitive_sensitivity = 100;//Set the sensitivity of the photosensitive
```

Initialize Emotion led matrix, buzzer, photoresistor and motor.

```
8     Emotion_Setup();          //Emotion initialization
9     Buzzer_Setup();           //Buzzer initialization
10    Photosensitive_Setup();   //Photosensitive initialization
11    PCA9685_Setup();          //Initialize PCA9685 to control motor
```

Get the ADC value of ambient light during initialization.

```
12  photosensitive_init_value = Get_Photosensitive();
```

Determine the relative position of the light source and the car.

```
18  if (Get_Photosensitive() < (photosensitive_init_value - photosensitive_sensitivity))
…   ...
23  else if (Get_Photosensitive() > (photosensitive_init_value + photosensitive_sensitivity))
…   ...
28  else
```

Chapter 5 Line Tracking Car

5.1 Line tracking sensor

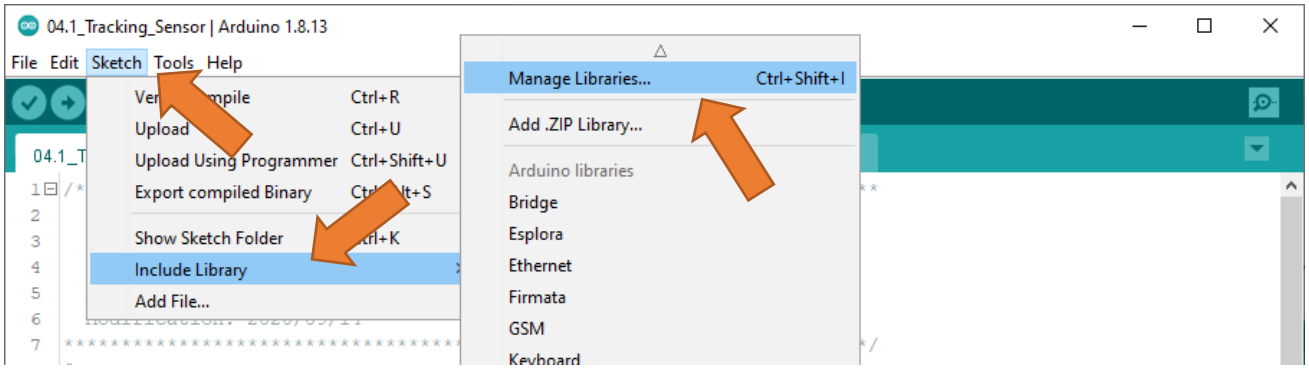
Track Sensor

There are three Reflective Optical Sensors on this car. When the infrared light emitted by infrared diode shines on the surface of different objects, the sensor will receive light with different intensities after reflection.
As we know, black objects absorb light better. So when black lines are drawn on the white plane, the sensor can detect the difference. The sensor can also be called Line Tracking Sensor.
Warning:
Reflective Optical Sensor (including Line Tracking Sensor) should be avoided using in environment with infrared interference, like sunlight. Sunlight contains a lot of invisible light such as infrared and ultraviolet. Under environment with intense sunlight, Reflective Optical Sensor cannot work normally.
The following table shows the values of all cases when three Tracking Sensors detect objects of different colors. Among them, black objects or no objects were detected to represent 1, and white objects were detected to represent 0.

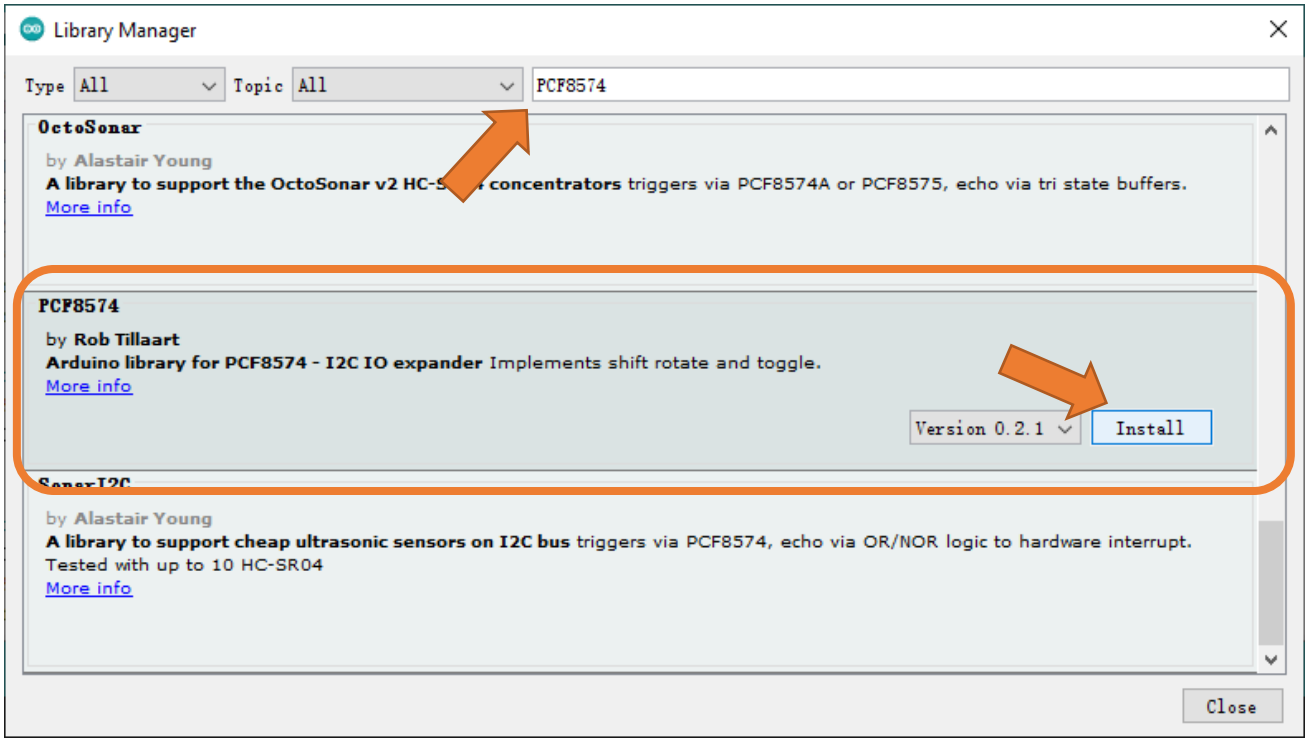
Left	Middle	Right	Value(binary)	Value(decimal)
0	0	0	000	0
0	0	1	001	1
0	1	0	010	2
0	1	1	011	3
1	0	0	100	4
1	0	1	101	5
1	1	0	110	6
1	1	1	111	7

Sketch

The car calls the PCF8574 library file, if you haven't installed it yet, please install it first.
Click "Sketch", select "Include Library" and then select "Manage Libraries".



Input “PCF8574” in the searching field and find the library marked below, click “Install”.



The line tracking module is connected to the PCF8574. The ESP32 obtains whether the three channels of the line tracking module are triggered by reading the IO value of the PCF8574, and prints it out through the serial port.

Open the folder “04.1_Tracking_Sensor” in **“Freenove_4WD_Car_Kit_for_ESP32\Sketches”** and double click “04.1_Tracking_Sensor.ino”.

Code

```
#include <Arduino.h>
#include "Freenove_4WD_Car_For_ESP32.h"

void setup() {
  Serial.begin(115200); //set baud rate
  Track_Setup();
}

void loop() {
  Track_Read();
  Serial.print("Sensor Value (L / M / R / ALL) : ");
  for (int i = 0; i < 4; i++) {
    Serial.print(sensorValue[i]);
    Serial.print('\t');
  }
  Serial.print('\n');
  delay(500);
}
```

Code Explanation:

Initialize PCF8574 chip.

```
6    Track_Setup();
```

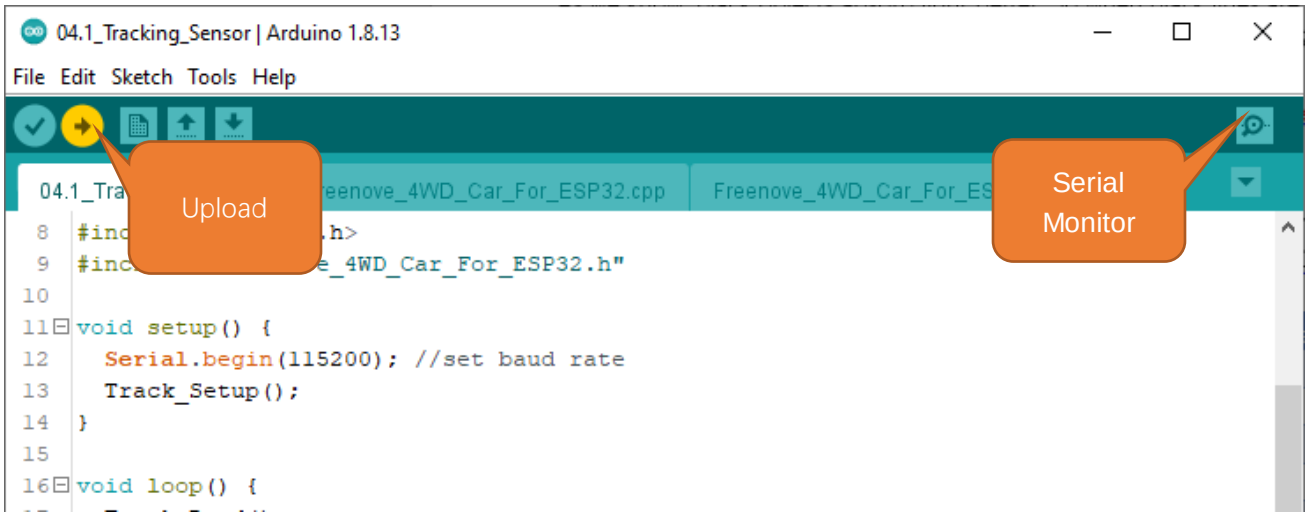
The function that obtains the tracking module feedback value. After calling this function, ESP32 will store the data in the array sensorValue[].

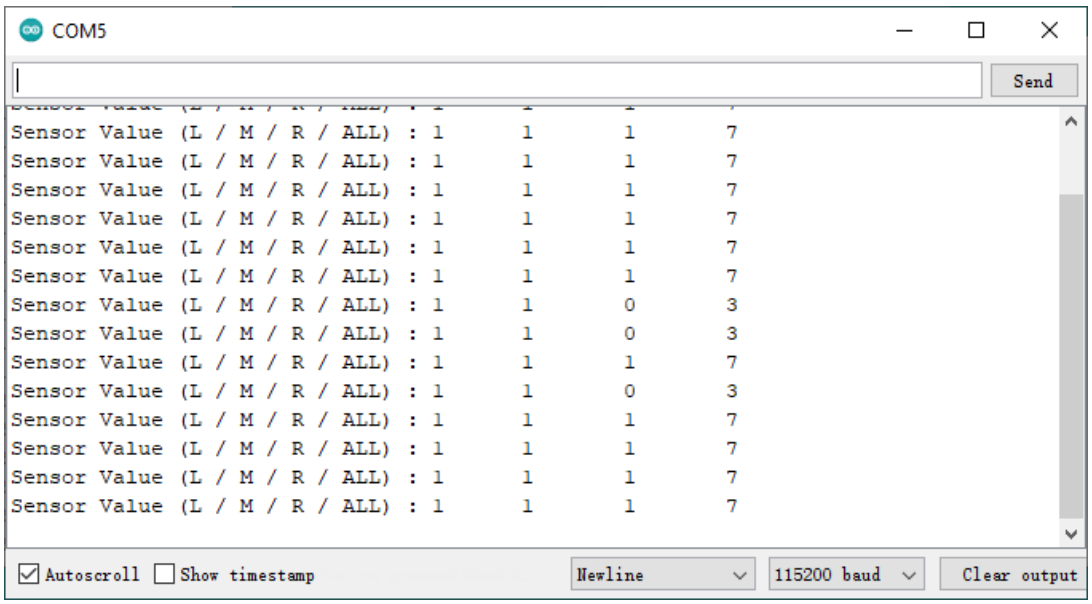
```
10   Track_Read();
```

Print the obtained feedback value through serial port.

```
11   Serial.print("Sensor Value (L / M / R / ALL) : ");
12   for (int i = 0; i < 4; i++) {
13     Serial.print(sensorValue[i]);
14     Serial.print('\t');
15   }
16   Serial.print('\n');
```

Click "Upload" to upload to code to ESP32 development board. After uploading successfully, click "Serial Monitor".



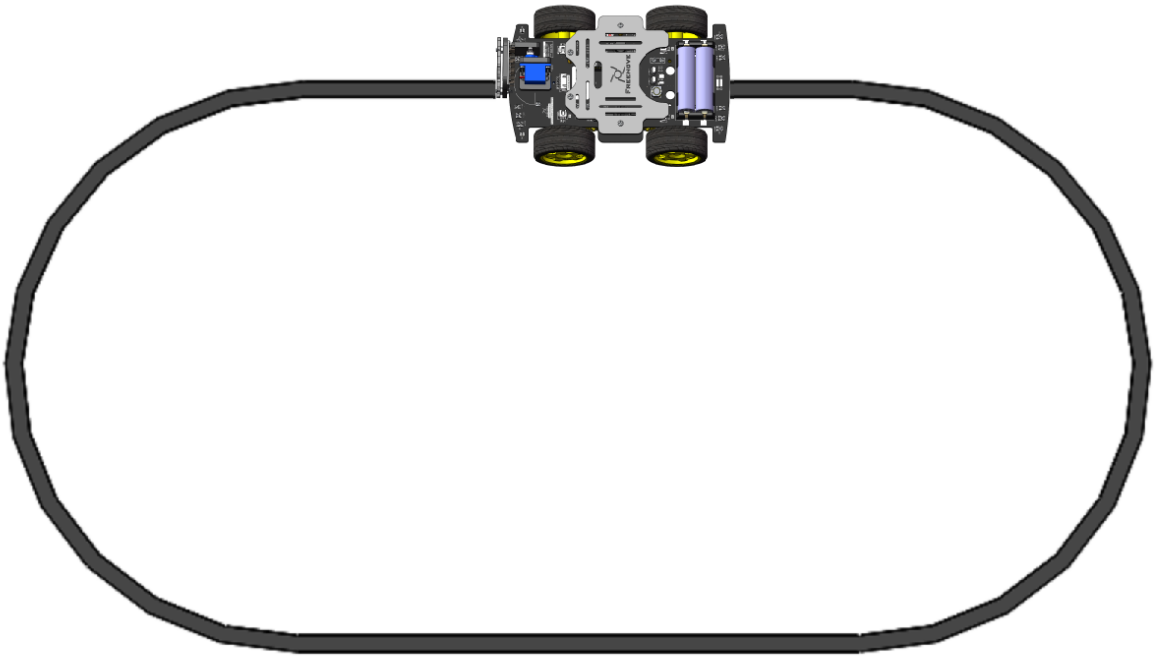


5.2 Line Tracking Car

The car will make different actions according to the value transmitted by the line-tracking sensor.

Left	Middle	Right	Value(binary)	Value(decimal)	Action
0	0	0	000	0	Stop
0	0	1	001	1	Turn Left
0	1	0	010	2	Move Forward
0	1	1	011	3	Turn Left
1	0	0	100	4	Turn Right
1	0	1	101	5	Move Forward
1	1	0	110	6	Turn Right
1	1	1	111	7	Stop

Turn on the power. Use a black tape to build a line and then put your car on it as below.



Sketch

Open the folder "04.2_Track_Car" in **"Freenove_4WD_Car_Kit_for_ESP32\Sketches"** and double click "04.2_Track_Car.ino".

Code

```
#include <Arduino.h>
#include "Freenove_4WD_Car_For_ESP32.h"

#define SPEED_LV4    ( 4000 )
```

```
#define SPEED_LV3   ( 3000 )
#define SPEED_LV2   ( 2500 )
#define SPEED_LV1   ( 1500 )

void setup(){
  Track_Setup();  //Trace module initialization
  PCA9685_Setup();//Motor drive initialization
  Emotion_Setup();//Emotion initialization
}

void loop(){
  Track_Read();
  switch (sensorValue[3])
  {
    case 2:   //010
    case 5:   //101
      showArrow(1, 100);
      Motor_Move(SPEED_LV1, SPEED_LV1, SPEED_LV1, SPEED_LV1);    //Move Forward
      break;
    case 0:   //000
    case 7:   //111
      eyesBlink1(100);
      Motor_Move(0, 0, 0, 0);                                    //Stop
      break;
    case 4:   //100
    case 6:   //110
      wheel(2, 100);
      Motor_Move(SPEED_LV4, SPEED_LV4 , - SPEED_LV3, -SPEED_LV3);//Turn Right
      break;
    case 1:   //001
    case 3:   //011
      wheel(1, 100);
      Motor_Move(-SPEED_LV3, -SPEED_LV3, SPEED_LV4, SPEED_LV4);  //Turn Left
      break;
    default:
      break;
  }
}
```

Code Explanation:

Initialize PCF8574 chip, PCA9685 chip, Emotion module.

```
  Track_Setup();  //Trace module initialization
  PCA9685_Setup();//Motor drive initialization
  Emotion_Setup();//Emotion initialization
```

The function that obtains the tracking module feedback value. After calling this function, ESP32 will store the data in the array sensorValue[].

```
16    Track_Read();
```

Control the car to move forward, turn left, turn right, stop, etc. based on the value of line tracking module.

```
17    switch (sensorValue[3])
18    {
19      case 2:   //010
20      case 5:   //101
21        showArrow(1, 100);
22        Motor_Move(SPEED_LV1, SPEED_LV1, SPEED_LV1, SPEED_LV1);     //Move Forward
23        break;
24      case 0:   //000
25      case 7:   //111
26        eyesBlink1(100);
27        Motor_Move(0, 0, 0, 0);                                     //Stop
28        break;
29      case 4:   //100
30      case 6:   //110
31        wheel(2, 100);
32        Motor_Move(SPEED_LV4, SPEED_LV4 , - SPEED_LV3, -SPEED_LV3);//Turn Right
33        break;
34      case 1:   //001
35      case 3:   //011
36        wheel(1, 100);
37        Motor_Move(-SPEED_LV3, -SPEED_LV3, SPEED_LV4, SPEED_LV4);  //Turn Left
38        break;
39      default:
40        break;
41    }
```

Note: The tracks made of black tape vary from different people. If your car cannot move along the black tape, you can modify the parameters in Motor_Move().

Chapter 6 Infrared Car

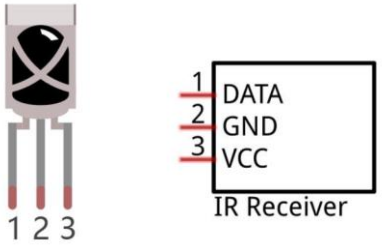
6.1 Introduction of infrared reception function

Infrared Remote

An infrared(IR) remote control is a device with a certain number of buttons. Pressing down different buttons will make the infrared emission tube, which is located in the front of the remote control, send infrared ray with different command. Infrared remote control technology is widely used in electronic products such as TV, air conditioning, etc. Thus making it possible for you to switch TV programs and adjust the temperature of the air conditioning when away from them. The remote control we use is shown below:

Infrared receiver

An infrared(IR) receiver is a component which can receive the infrared light, so we can use it to detect the signal emitted by the infrared remote control. DATA pin here outputs the received infrared signal.



When you use the infrared remote control, the infrared remote control sends a key value to the receiving circuit according to the pressed keys. We can program the ESP32-WROVER to do things like lighting, when a key value is received.

The following is the key value that the receiving circuit will receive when each key of the infrared remote control is pressed.

ICON	KEY Value	ICON	KEY Value
	FFA25D	C	FFB04F
MENU	FFE21D	1	FF30CF
TEST	FF22DD	2	FF18E7
+	FF02FD	3	FF7A85
	FFC23D	4	FF10EF
	FFE01F	5	FF38C7
	FFA857	6	FF5AA5
	FF906F	7	FF42BD
0	FF6897	8	FF4AB5
-	FF9867	9	FF52AD

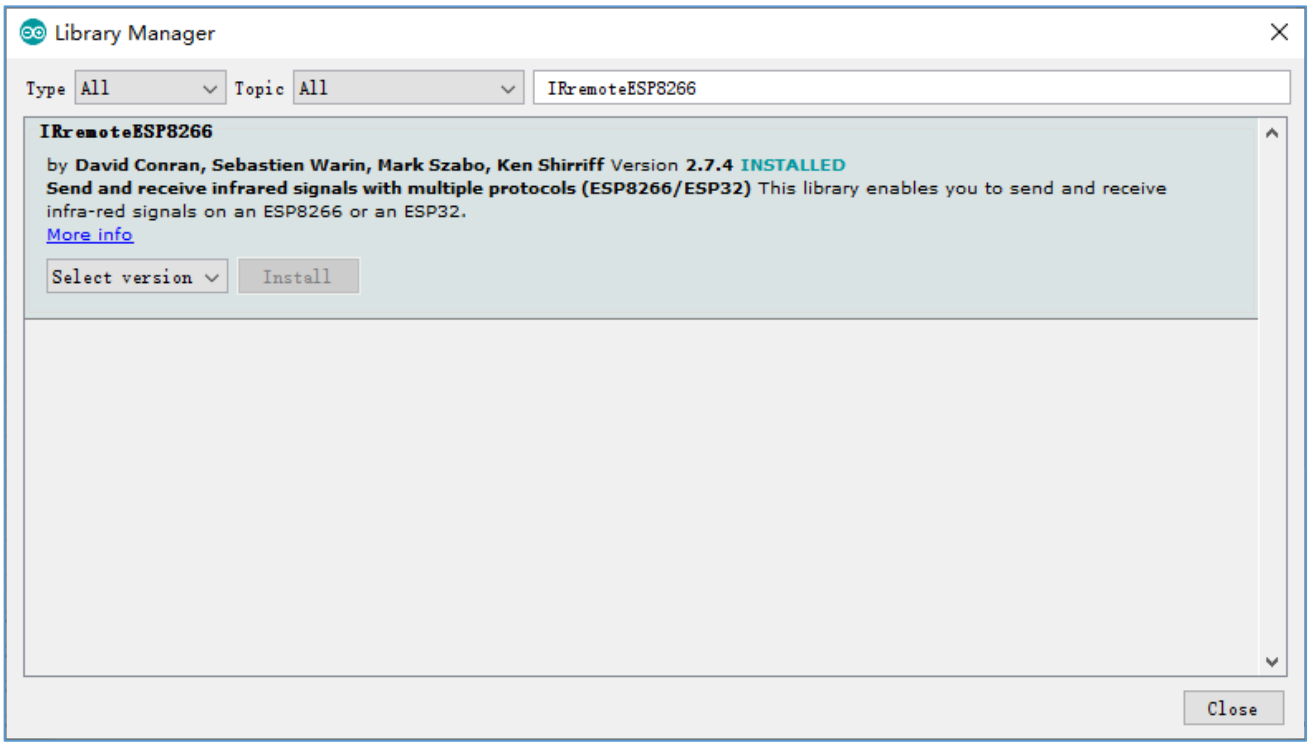
This sketch uses the infrared receiving tube to receive the value sent form the infrared remote control, and print it out via the serial port.

Sketch

We use the third party library IRremoteESP8266. If you haven't installed it yet, please do so first.

The steps to add third-party Libraries are as follows: open arduino->Sketch->Include library-> Manage libraries. Enter "IRremoteESP8266" in the search bar and select " IRremoteESP8266" for installation.

Refer to the following operations:



Each time when you press the infrared remote control, the car will print the received infrared coding value through serial port.

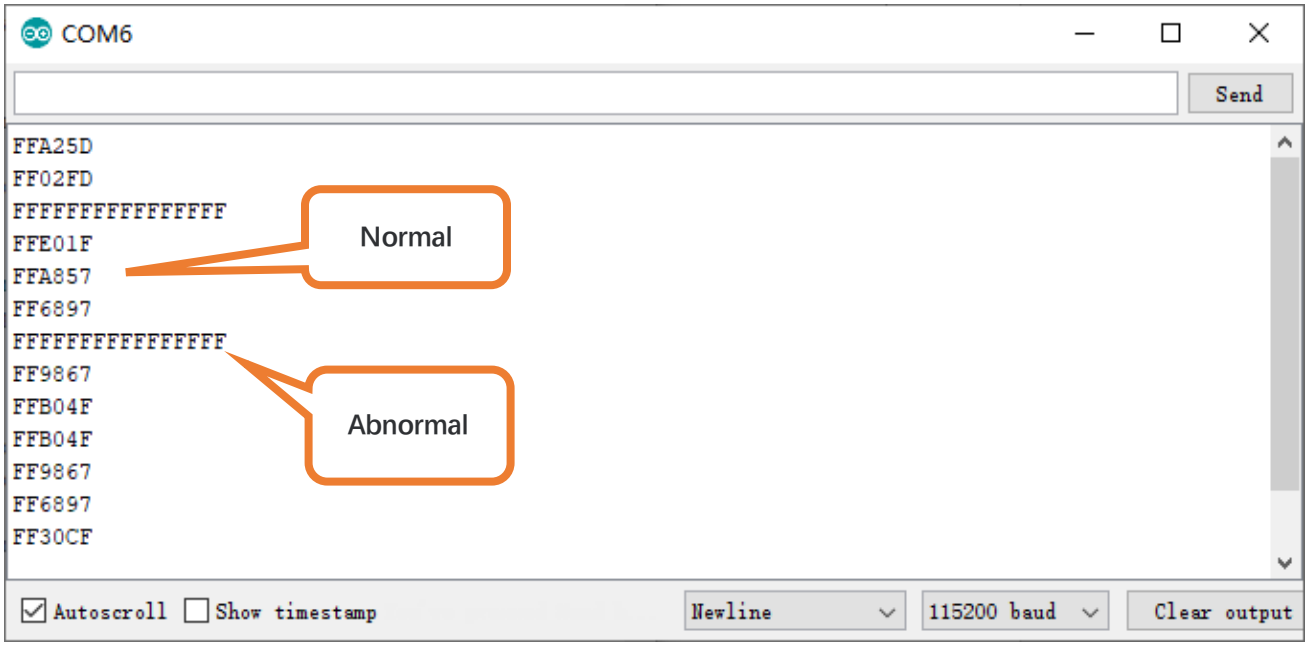
Open the folder "05.1_IR_Receiver" in **"Freenove_4WD_Car_Kit_for_ESP32\Sketches"** and double click "05.1_IR_Receiver.ino".

Code

```
#include <Arduino.h>
#include <IRremoteESP8266.h>
#include <IRrecv.h>
#include <IRutils.h>

#define RECV_PIN     0         // Infrared receiving pin
IRrecv irrecv(RECV_PIN);       // Create a class object used to receive class
decode_results results;        // Create a decoding results class object

void setup()
{
  Serial.begin(115200);        // Initialize the serial port and set the baud rate to 115200
  irrecv.enableIRIn();         // Start the receiver
  Serial.print("IRrecvDemo is now running and waiting for IR message on Pin ");
  Serial.println(RECV_PIN);    //print the infrared receiving pin
}

void loop()
{
```

```
20    if (irrecv.decode(&results)) {          // Waiting for decoding
21      unsigned long value = results.value;
22      Serial.println(value,HEX);            // Print out the decoded results
23      irrecv.resume();                      // Release the IRremote. Receive the next value
24    }
25    delay(100);
26  }
```

Download the code to ESP32-WROVER, open the serial port monitor, set the baud rate to 115200, press the IR remote control, the pressed keys value will be printed out through the serial port. As shown in the following figure: (Note that when the remote control button is pressed for a long time, the infrared receiving circuit receives a continuous high level, that is, it receives a hexadecimal "F")



First, include header file. Each time you use the infrared library, you need to include the header file at the beginning of the program.

```
1  #include <Arduino.h>
2  #include <IRremoteESP8266.h>
3  #include <IRrecv.h>
4  #include <IRutils.h>
```

Second, define an infrared receive pin and associates it with the receive class. Apply a decode_results to decode the received infrared value.

```
6  #define RECV_PIN     0          // Infrared receiving pin
7  IRrecv irrecv(RECV_PIN);        // Create a class object used to receive class
8  decode_results results;         // Create a decoding results class object
```

Third, enable infrared reception function, if you do not use this function, you won't receive the value from the infrared remote control.

```
13  irrecv.enableIRIn();          // Start the receiver
```

Finally, put the received data into the results class and print out the data through the serial port. Note that you must use **resume**() to release the infrared receive function every time when you receive data, otherwise you can only use the infrared receive function once and cannot receive the data next time.

```
18  void loop() {
19    if (irrecv.decode(&results)) {        // Waiting for decoding
20      serialPrintUint64(results.value, HEX); // Print out the decoded results
21      Serial.println("");
22      irrecv.resume();                    // Release the IRremote. Receive the next value
23    }
24    delay(1000);
25  }
```

Reference

class IRrecv	You need to add the library each time you use the Infrared Reception.
IRrecv irrecv(Pin): Create a class object used to receive class, and associated with **Pin**.	
enableIRIn(): Before using the infrared decoding function, enable the infrared receiving function. Otherwise the correct data will not be received.	
decode(&results): Determine whether the infrared has received data, and if so, return true and store the data in the decode_results class. If no data is received, false is returned.	
resume(): Release the IRremote. Or, the infrared reception and decoding function cannot be used again.	

For more information about Infrared Remote Control, please visit:
https://github.com/crankyoldgit/IRremoteESP8266/tree/master/src

6.2 Infrared Car

On the basis of the previous section, we use the infrared remote control to control the car. Press the black button on the infrared remote control to control the car to move forward, backward, turn left, and turn right. Press the other buttons and the cart stops moving.

Sketch

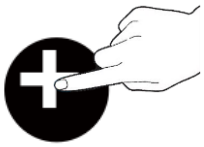
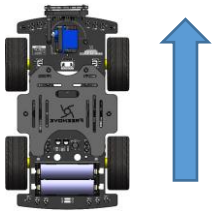
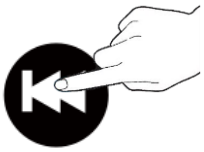
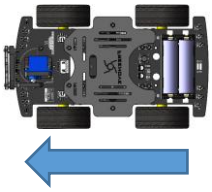
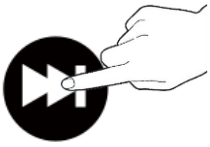
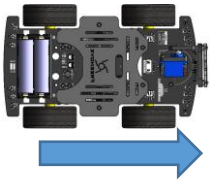
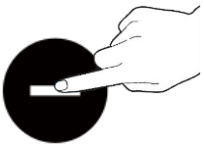
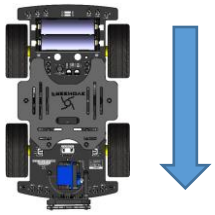
Open the folder "05.2_IR_Receiver_Car" in the **"Freenove_4WD_Car_Kit_for_ESP32\Sketches"** and double click "05.2_IR_Receiver_Car.ino".

Code

```
#include <Arduino.h>
#include <IRremoteESP8266.h>
#include <IRrecv.h>
#include <IRutils.h>
#include "Freenove_4WD_Car_For_ESP32.h"

#define RECV_PIN     0         // Infrared receiving pin
IRrecv irrecv(RECV_PIN);       // Create a class object used to receive class
decode_results results;        // Create a decoding results class object
int motor_speed = 3000;
int emotionMode = 4;
int motor_flag = 0;

void setup(){
  irrecv.enableIRIn();         //Initialize the infrared receiving pin
  PCA9685_Setup();             //Initializes the motor control chip
  Emotion_Setup();             //Initializes Emotion module
}

void loop(){
  if (irrecv.decode(&results)) {  // Waiting for decoding
    handleControl(results.value); // Handle the commands from remote control
    irrecv.resume();
  }
  showEmotion(emotionMode);
}

void handleControl(unsigned long value) {
  // Handle the commands
  switch (value) {
    case 0xFF02FD:// Receive the number '+'
```

```
32          motor_flag = 1;
33          Motor_Move(motor_speed, motor_speed, motor_speed, motor_speed);     // Go forward
34          delay(200);
35          Motor_Move(0, 0, 0, 0);
36          break;
37        case 0xFF9867:// Receive the number '-'
38          motor_flag = 2;
39          Motor_Move(-motor_speed, -motor_speed, -motor_speed, -motor_speed); // Back up
40          delay(200);
41          Motor_Move(0, 0, 0, 0);
42          break;
43        case 0xFFE01F:// Receive the number '|<<'
44          motor_flag = 3;
45          Motor_Move(-motor_speed, -motor_speed, motor_speed, motor_speed);   // Turn left
46          delay(200);
47          Motor_Move(0, 0, 0, 0);
48          break;
49        case 0xFF906F:// Receive the number '>>|'
50          motor_flag = 4;
51          Motor_Move(motor_speed, motor_speed, -motor_speed, -motor_speed);   // Turn right
52          delay(200);
53          Motor_Move(0, 0, 0, 0);
54          break;
55        case 0xFF6897:// Receive the number '0'
56          emotionMode = millis() % 10;
57          break;
58        case 0xFFFFFFFF:// Remain unchanged
59          if (motor_flag == 1)
60            Motor_Move(motor_speed, motor_speed, motor_speed, motor_speed);     // Go forward
61          else if (motor_flag == 2)
62            Motor_Move(-motor_speed, -motor_speed, -motor_speed, -motor_speed); // Back up
63          else if (motor_flag == 3)
64            Motor_Move(-motor_speed, -motor_speed, motor_speed, motor_speed);   // Turn left
65          else if (motor_flag == 4)
66            Motor_Move(motor_speed, motor_speed, -motor_speed, -motor_speed);   // Turn right
67          else
68            Motor_Move(0, 0, 0, 0);
69          break;
70        default:        // Control the car to stop moving
71          motor_flag = 0;
72          Motor_Move(0, 0, 0, 0); //stop
73          break;
74      }
75  }
```

Compile and upload the code to the ESP32-WROVER. When pressing "0", "1", "2", "3" of the infrared remote control, the buzzer will sound once, and the brightness of the LED light will change correspondingly.

		**Move forward**
		**Turn left**
		**Turn right**
		**Move back**

Code Explanation:

Call the decode function to obtain the infrared remote control coding information, and call the handleControl function to execute actions corresponding to different code values. After each execution of the program, call the resume function to release the infrared pin. If this function is not called, the infrared receiving and decoding functions cannot be used again.

```
20  void loop(){
21    if (irrecv.decode(&results)) {  // Waiting for decoding
22      handleControl(results.value); // Handle the commands from remote control
23      irrecv.resume();
24    }
25    showEmotion(emotionMode);
26  }
```

Infrared key code value processing function, received the infrared remote control sent instructions, the execution of the corresponding program.

```
28  void handleControl(unsigned long value) {
29    // Handle the commands
30    switch (value) {
31      case 0xFF02FD:// Receive the number '+'
...       ...
37      case 0xFF9867:// Receive the number '-'
```

```
...     ...
43      case 0xFFE01F:// Receive the number'|<<'
...      ...
49      case 0xFF906F:// Receive the number'>>|'
...      ...
55      case 0xFF6897:
...      ...
58      case 0xFFFFFFFF:// Remain unchanged
...      ...
70      default:        // Control the car to stop moving
...      ...
74    }
75  }
```

6.3 Multi-Functional Infrared Car

On the basis of the last section, we integrate other functions of the car into the infrared car, and most of the functions of the car can be controlled by the infrared remote control.

Sketch

Open the folder "05.3_Multi_Functional_Car" in the **"Freenove_4WD_Car_Kit_for_ESP32\Sketches"** and double click "05.3_Multi_Functional_Car.ino".

Code

```
#include <Arduino.h>
#include <IRremoteESP8266.h>
#include <IRrecv.h>
#include <IRutils.h>
#include "Freenove_4WD_Car_For_ESP32.h"
#include "Freenove_4WD_Car_Emotion.h"
#include "Freenove_WS2812_Lib_for_ESP32.h"

Freenove_ESP32_WS2812 strip = Freenove_ESP32_WS2812(12, 32, 0, TYPE_GRB);
u8 m_color[5][3] = { {255, 0, 0}, {0, 255, 0}, {0, 0, 255}, {255, 255, 255}, {0, 0, 0} };
#define RECV_PIN     0        // Infrared receiving pin
IRrecv irrecv(RECV_PIN);      // Create a class object used to receive class
decode_results results;       // Create a decoding results class object

static int servo_1_angle=90;
static int servo_2_angle=90;
static int emotion_flag=0;
static int ws2812_flag=0;

void setup(){
  Serial.begin(115200);
  PCA9685_Setup();
  Buzzer_Setup();
  Emotion_Setup();
  strip.begin();
  strip.setBrightness(10);
  irrecv.enableIRIn();            //Initialize the infrared receiving pin
  Servo_1_Angle(servo_1_angle);
  Servo_2_Angle(servo_2_angle);
}

void loop(){
```

```
33    if (irrecv.decode(&results)) {  // Waiting for decoding
34      handleControl(results.value); // Handle the commands from remote control
35      unsigned long value=results.value;
36      Serial.print(value,HEX);
37      Serial.println();
38      irrecv.resume();              // Release infrared decoding function
39    }
40  }
41
42  void handleControl(unsigned long value) {
43    // Handle the commands
44    switch (value) {
45      case 0xFF02FD:// Receive the number '+'
46        Motor_Move(2000,2000,2000,2000);
47        break;
48      case 0xFF9867:// Receive the number '-'
49        Motor_Move(-2000,-2000,-2000,-2000);
50        break;
51      case 0xFFE01F:// Receive the number '|<<'
52        Motor_Move(-2000,-2000,2000,2000);
53        delay(200);
54        Motor_Move(0,0,0,0);
55        break;
56      case 0xFF906F:// Receive the number '>>|'
57        Motor_Move(2000,2000,-2000,-2000);
58        delay(200);
59        Motor_Move(0,0,0,0);
60        break;
61      case 0xFFA857:// Receive the number '▶'
62        Motor_Move(0,0,0,0);
63        break;
64      case 0xFF6897:// Receive the number '0'
65        servo_1_angle=servo_1_angle+10;
66        Servo_1_Angle(servo_1_angle);
67        break;
68      case 0xFF30CF:// Receive the number '1'
69        servo_1_angle=servo_1_angle-10;
70        Servo_1_Angle(servo_1_angle);
71        break;
72      case 0xFF10EF:// Receive the number '4'
73        servo_1_angle=90;
74        Servo_1_Angle(servo_1_angle);
75        break;
76      case 0xFFB04F:// Receive the number 'C'
```

```
77          servo_2_angle=servo_2_angle+10;
78          Servo_2_Angle(servo_2_angle);
79          break;
80        case 0xFF7A85:// Receive the number '3'
81          servo_2_angle=servo_2_angle-10;
82          Servo_2_Angle(servo_2_angle);
83          break;
84        case 0xFF5AA5:// Receive the number '6'
85          servo_2_angle=90;
86          Servo_2_Angle(servo_2_angle);
87          break;
88        case 0xFF22DD:// Receive the number 'TEST'
89          Buzzer_Alert(1,1);
90          break;
91        case 0xFF18E7:// Receive the number '2'
92          emotion_flag=millis()%21;
93          staticEmtions(emotion_flag);
94          break;
95        case 0xFF38C7:// Receive the number '5'
96          clearEmtions();
97          break;
98        case 0xFF42BD:// Receive the number '7'
99          ws2812_flag=millis()%4;
100         WS2812_Show();
101         break;
102       case 0xFF4AB5:// Receive the number '8'
103         ws2812_flag=4;
104         WS2812_Show();
105         break;
106       case 0xFFFFFFFF:// Remain unchanged
107         break;
108       default:
109         break;
110     }
111 }
112
113 void WS2812_Show(){
114   for (int i = 0; i < 12; i++)
115     strip.setLedColorData(i, m_color[ws2812_flag][0], m_color[ws2812_flag][1],
    m_color[ws2812_flag][2]);
116     strip.show();
117 }
```

After the code is successfully uploaded, turn on the power of the car and use the infrared remote control to control the car and other functions. The corresponding keys and their functions are shown in the following table:

ICON	KEY Value	Function	ICON	KEY Value	Function
+	FF02FD	Move forward	TEST	FF22DD	Control the buzzer
⏮	FFE01F	Turn left	2	FF18E7	Random emoticons
⏭	FF906F	Turn light	5	FF38C7	Turn off emoticons
−	FF9867	Move back	7	FF42BD	Random display of WS2812
▶	FFA857	Stop the car	8	FF4AB5	Turn off WS2812 display
0	FF6897	Control servo 1 turn left	C	FFB04F	Control servo 2 turn left
1	FF30CF	Control servo 1 turn right	3	FF7A85	Control servo 2 turn right
4	FF10EF	Control servo 1 turn to 90°	6	FF5AA5	Control servo 2 turn to 90°

Code Explanation:

Add the header file for the car.

```
#include <Arduino.h>
#include <IRremoteESP8266.h>
#include <IRrecv.h>
#include <IRutils.h>
#include "Freenove_4WD_Car_For_ESP32.h"
#include "Freenove_4WD_Car_Emotion.h"
#include "Freenove_WS2812_Lib_for_ESP32.h"
```

Initialize each function of the car.

```
  Serial.begin(115200);    // Turn on the serial port monitor and set the baud rate to 115200
  PCA9685_Setup();         // Initialize the PCA9685 chip
  Buzzer_Setup();          //Initialize the buzzer
  Emotion_Setup();         //Initializes the expression module
  strip.begin();           //Initialize the WS2812
  strip.setBrightness(10);//Set the brightness of the WS2812 to 10
  irrecv.enableIRIn();     //Initialize the infrared receiving pin
  Servo_1_Angle(servo_1_angle);
  Servo_2_Angle(servo_2_angle);
```

Set the color of WS2812 and have it display the corresponding color.

```
void WS2812_Show(){
  for (int i = 0; i < 12; i++)
    strip.setLedColorData(i, m_color[ws2812_flag][0], m_color[ws2812_flag][1],
m_color[ws2812_flag][2]);
    strip.show();
}
```

Infrared key code value processing function, received the infrared remote control sent instructions, the execution of the corresponding program.

```
void handleControl(unsigned long value) {
  // Handle the commands
  switch (value) {
    case 0xFF02FD:// Receive the number '+'
    ...
    case 0xFF9867:// Receive the number '-'
    ...
    case 0xFFE01F:// Receive the number '|<<'
    ...
    case 0xFF906F:// Receive the number '>>|'
    ...
    case 0xFFA857:// Receive the number '▶'
    ...
    case 0xFF6897:// Receive the number '0'
    ...
    case 0xFF30CF:// Receive the number '1'
```

```
...     ...
72      case 0xFF10EF://Receive the number '4'
...     ...
76      case 0xFFB04F://Receive the number 'C'
...     ...
80      case 0xFF7A85://Receive the number '3'
...     ...
84      case 0xFF5AA5://Receive the number '6'
...     ...
88      case 0xFF22DD://Receive the number 'TEST'
...     ...
91      case 0xFF18E7://Receive the number '2'
...     ...
95      case 0xFF38C7://Receive the number '5'
...     ...
98      case 0xFF42BD://Receive the number '7'
...     ...
102     case 0xFF4AB5://Receive the number '8'
...     ...
106     case 0xFFFFFFFF://Remain unchanged
...     ...
108     default:
109       break;
110   }
111 }
```

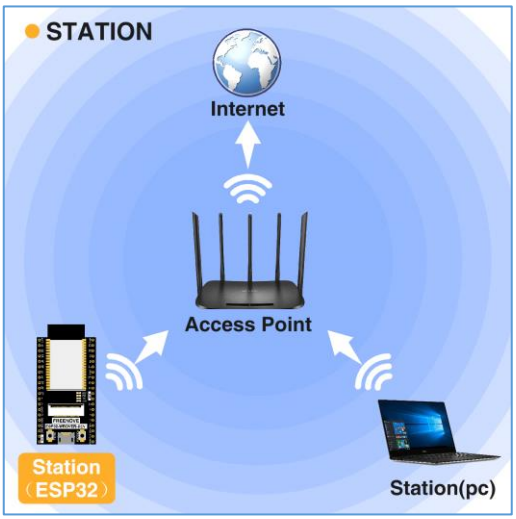
Chapter 7 WiFi Car

7.1 WiFi Sending and Receiving Data

Before programming, we need to have a basic understanding of WiFi.

Station mode

When ESP32 selects Station mode, it acts as a WiFi client. It can connect to the router network and communicate with other devices on the router via WiFi connection. As shown below, the PC is connected to the router, and if ESP32 wants to communicate with the PC, it needs to be connected to the router.



AP mode

When ESP32 selects AP mode, it creates a hotspot network that is separate from the Internet and waits for other WiFi devices to connect. As shown in the figure below, ESP32 is used as a hotspot. If a mobile phone or PC wants to communicate with ESP32, it must be connected to the hotspot of ESP32. Only after a connection is established with ESP32 can they communicate.

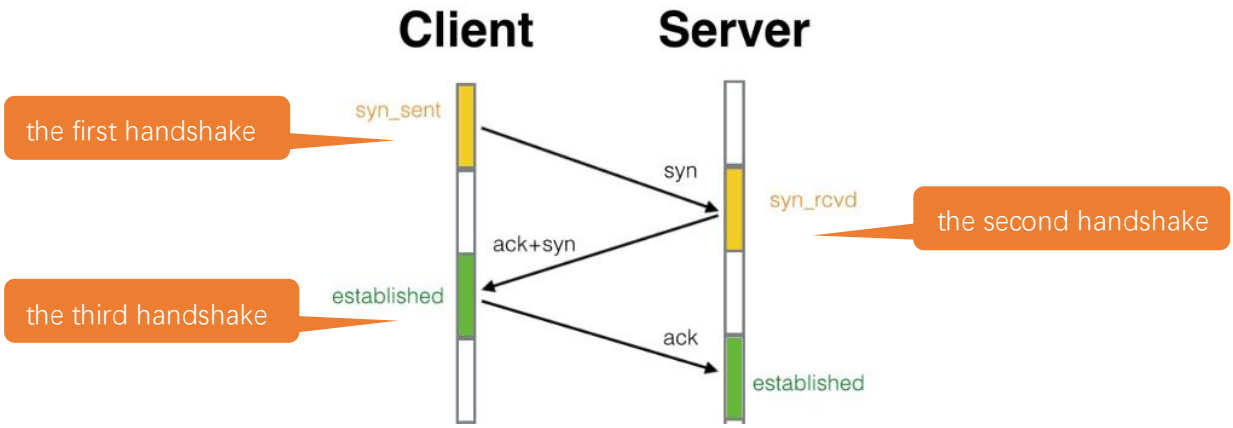
AP+Station mode

In addition to AP mode and station mode, ESP32 can also use AP mode and station mode at the same time. This mode contains the functions of the previous two modes. Turn on ESP32's station mode, connect it to the router network, and it can communicate with the Internet via the router. At the same time, turn on its AP mode to create a hotspot network. Other WiFi devices can choose to connect to the router network or the hotspot network to communicate with ESP32.

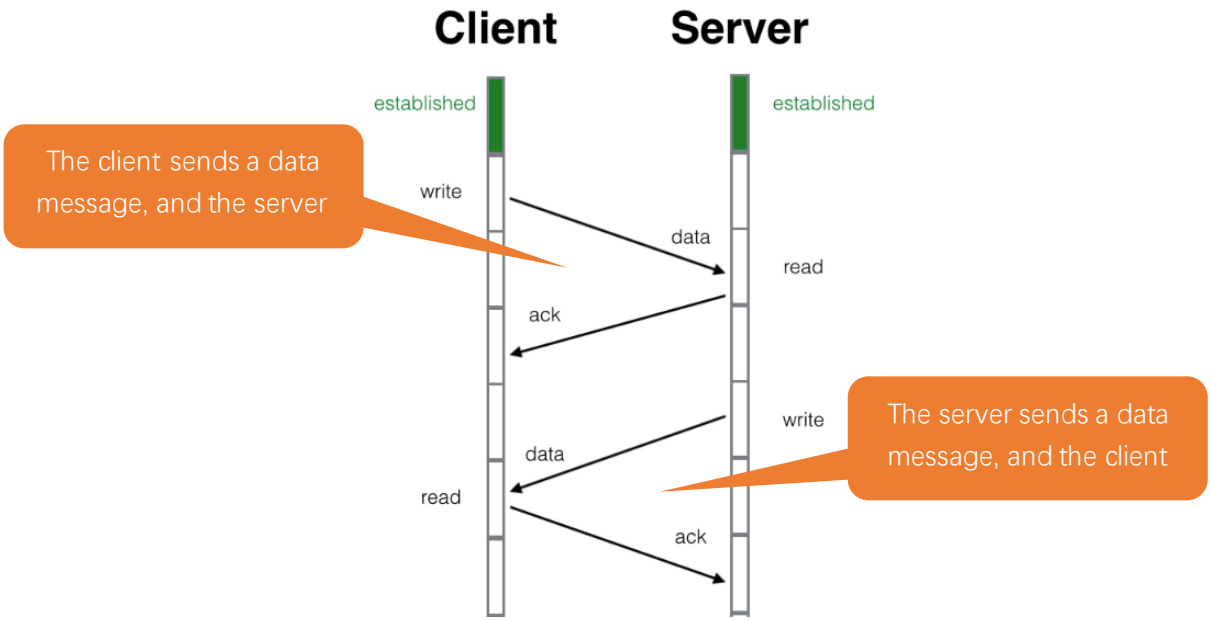
TCP connection

Before transmitting data, TCP needs to establish a logical connection between the sending end and the receiving end. It provides reliable and error-free data transmission between the two computers. In the TCP connection, the client and the server must be clarified. The client sends a connection request to the server, and each time such a request is proposed, a "three-times handshake" is required.

Three-times handshake: In the TCP protocol, during the preparation phase of sending data, the client and the server interact three times to ensure the reliability of the connection, which is called "three-times handshake".
The first handshake, the client sends a connection request to the server and waits for the server to confirm.
The second handshake, the server sends a response back to the client informing that it has received the connection request.
The third handshake, the client sends a confirmation message to the server again to confirm the connection.



TCP is a connection-oriented, low-level transmission control protocol. After TCP establishes a connection, the client and server can send and receive messages to each other, and the connection will always exist as long as the client or server does not initiate disconnection. Each time one party sends a message, the other party will reply with an ack signal.

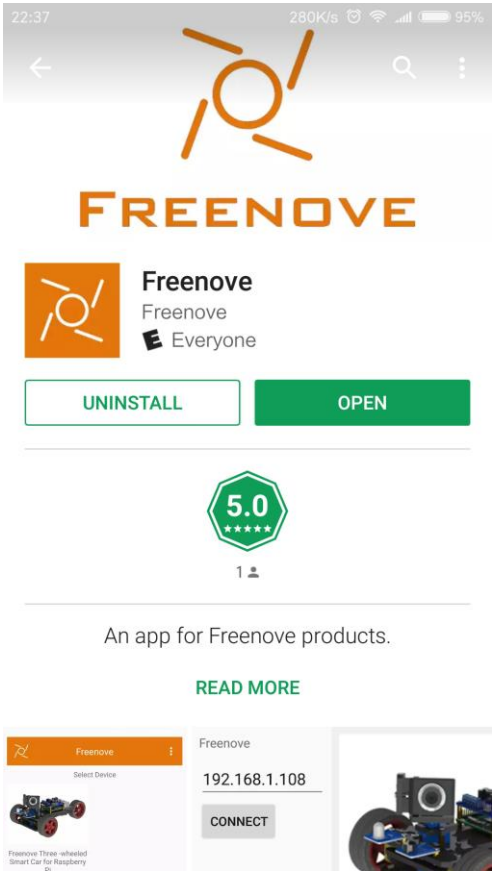


Install Freenove app

There are three ways to install app, you can choose any one.

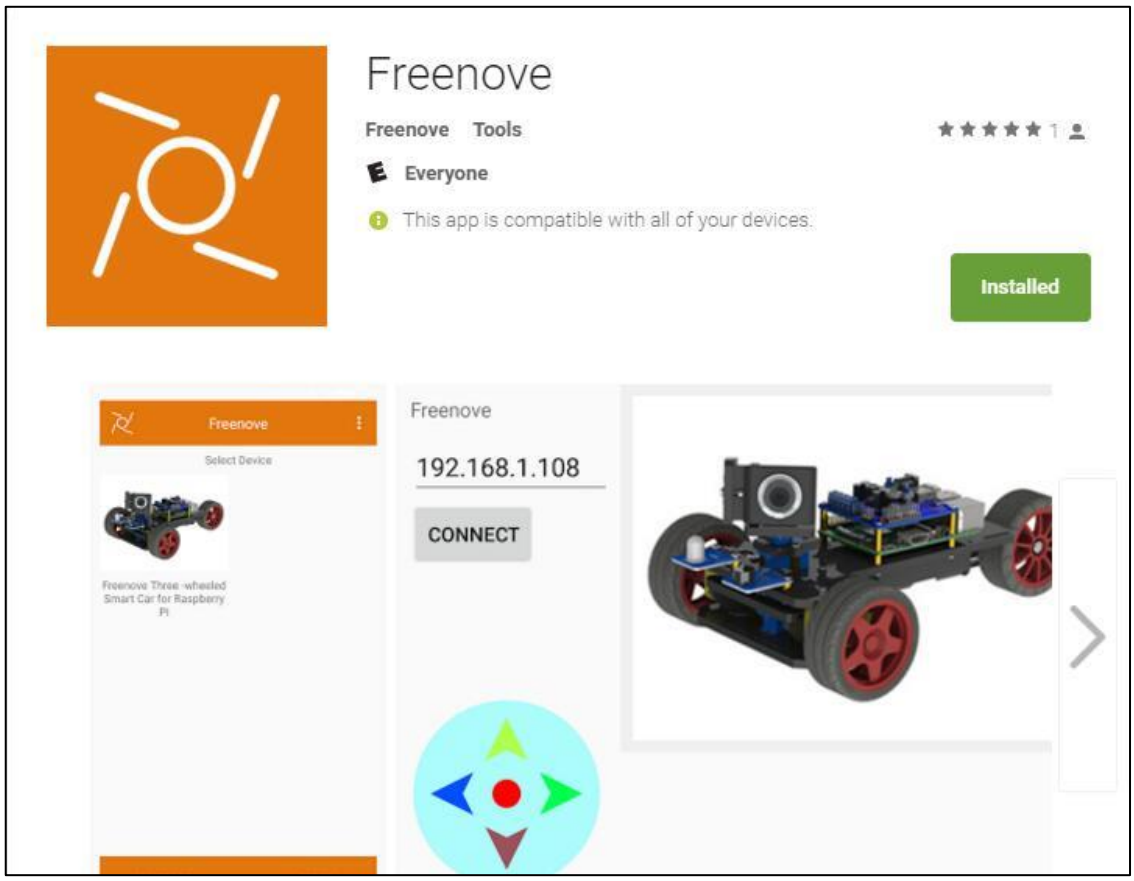
Method 1

Use Google play to search “Freenove”, download and install.



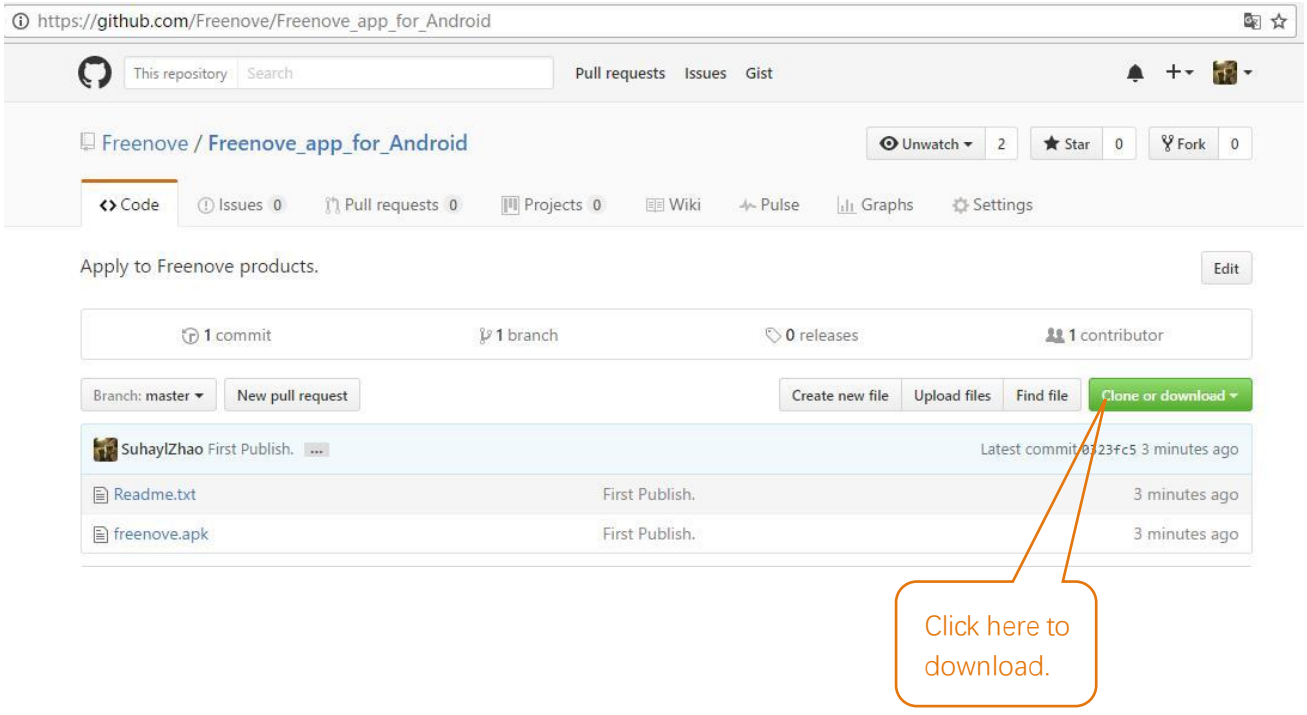
Method 2

Visit https://play.google.com/store/apps/details?id=com.freenove.suhayl.Freenove, and click install.



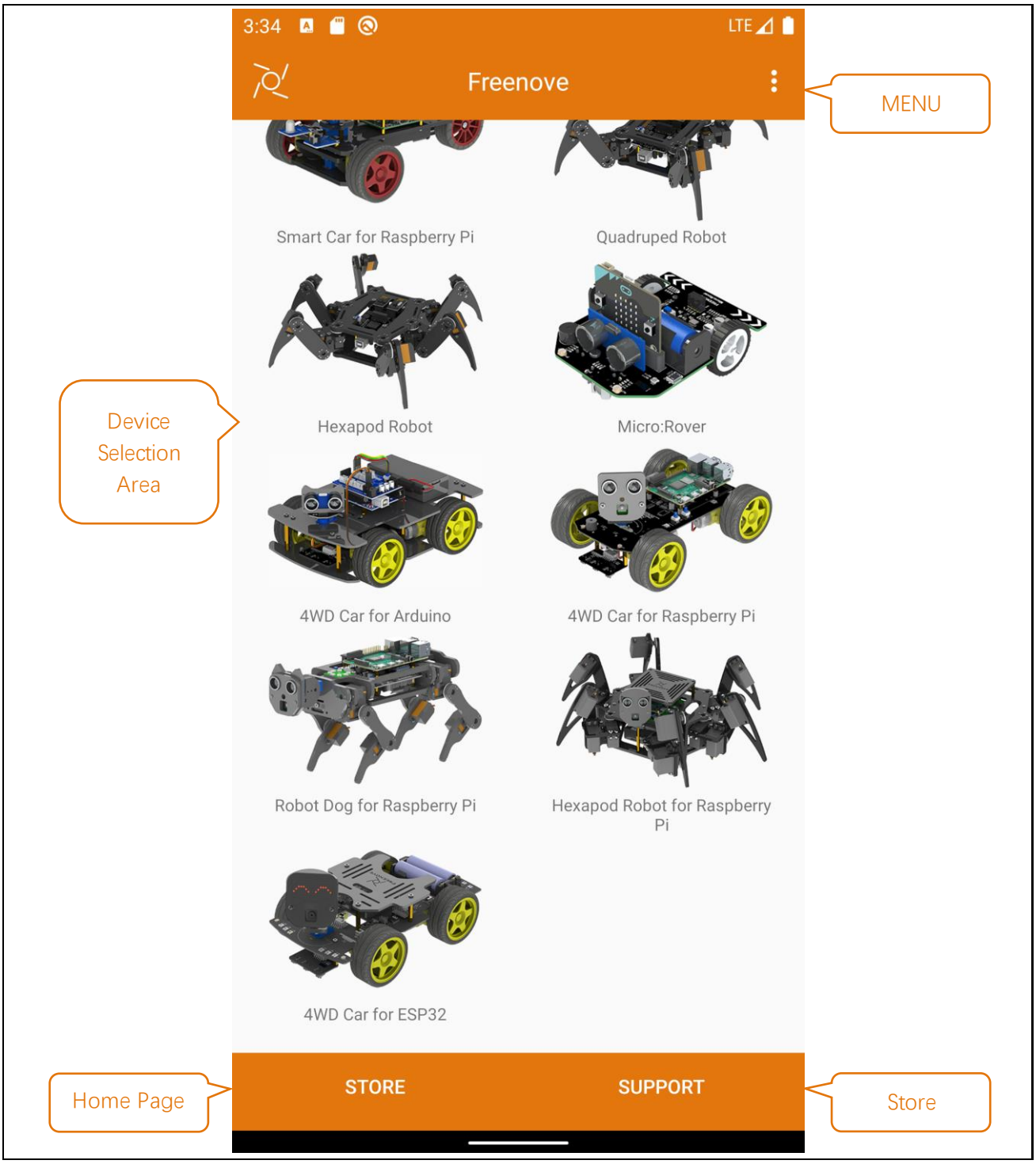
Method 3

Visit https://github.com/Freenove/Freenove_app_for_Android, download the files in this library, and install freenove.apk to your Android phone manually.



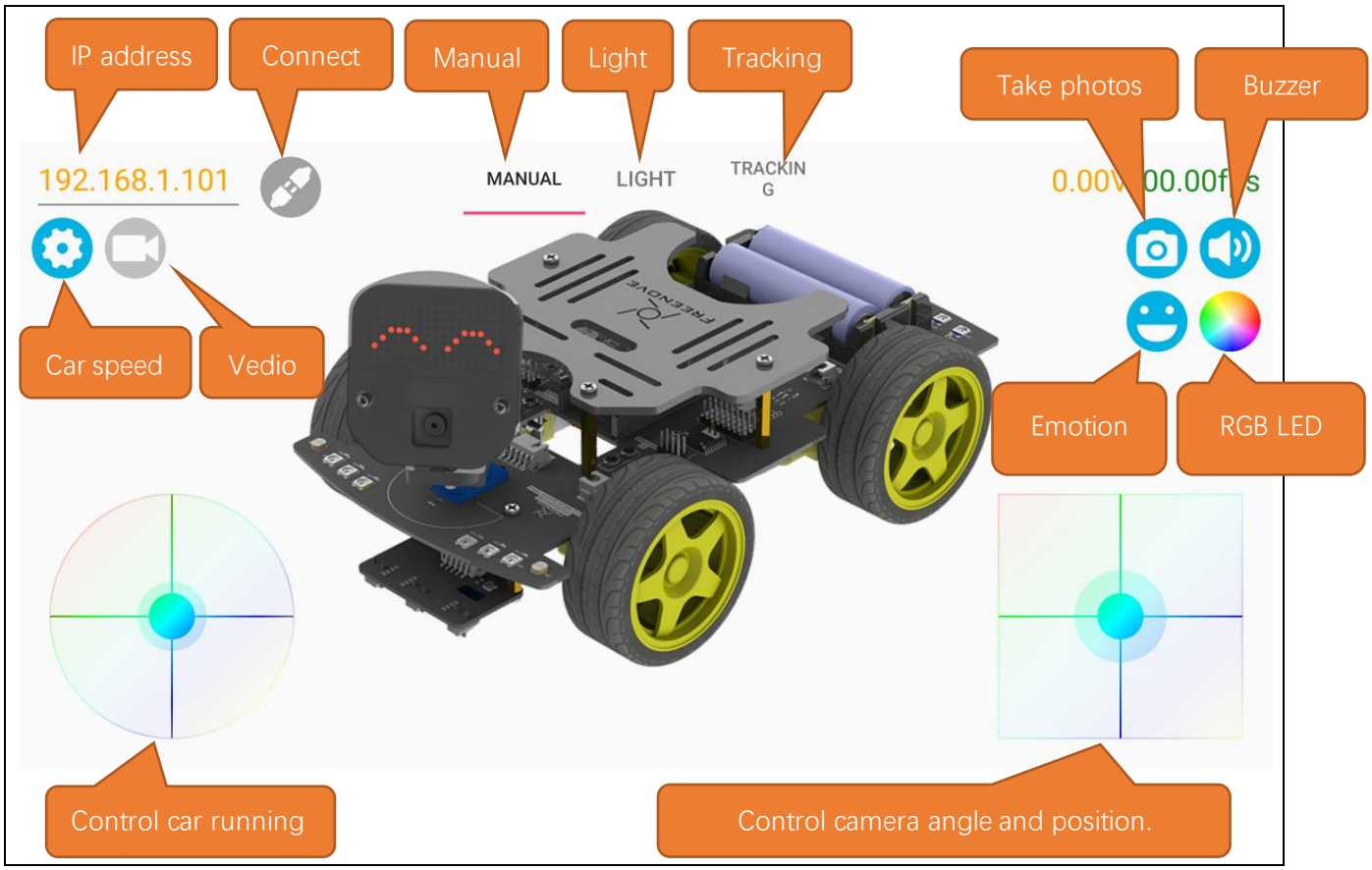
Menu

Open application "Freenove", as shown below:



Introduction of the APP

In this chapter, we use Freenove 4WD Car for ESP32, so it is necessary to understand the interface of this mode.



Sketch

Open "06.1_WiFi_APP_TcpServer" in **"Freenove_4WD_Car_Kit_for_ESP32\Sketches"** and double click "06.1_WiFi_APP_TcpServer.ino".

Before uploading the code, you need to modify it based on the name and password of your router.

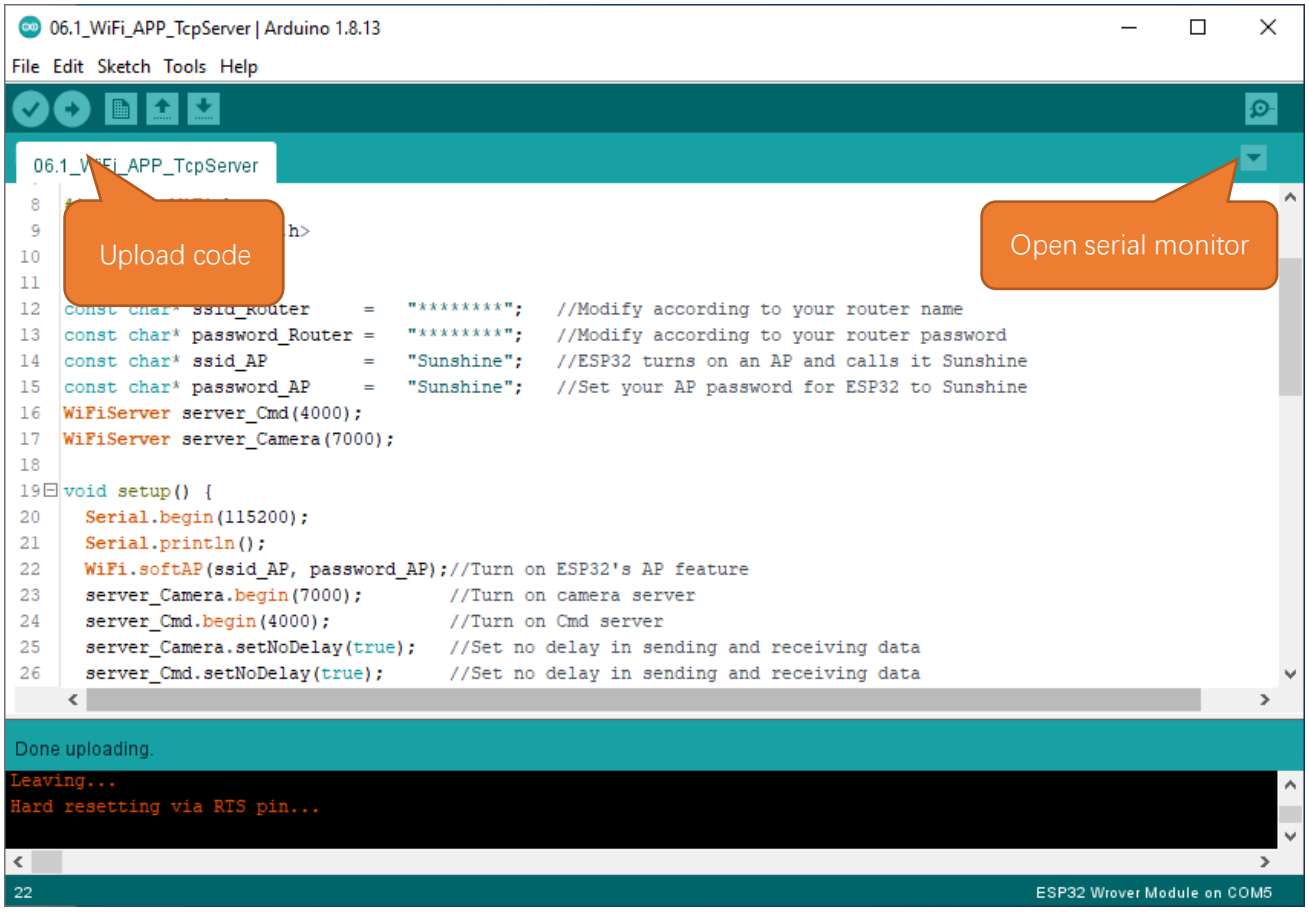
Code

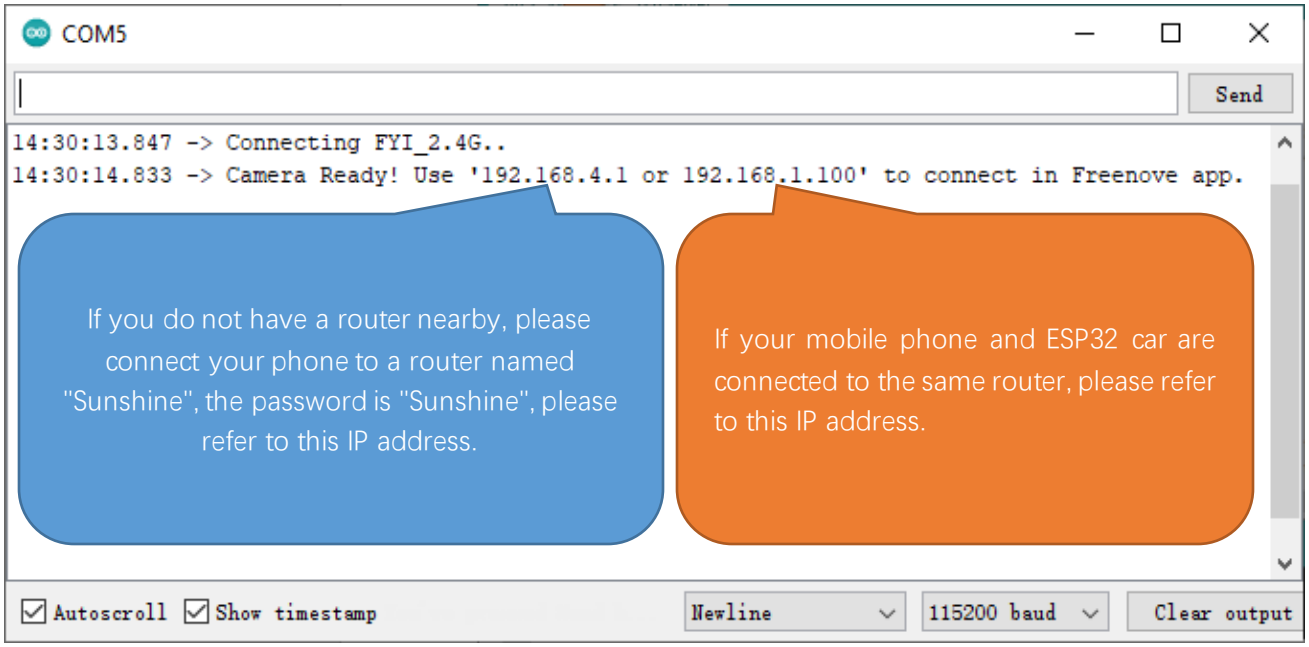
```
#include <WiFi.h>
#include <WiFiClient.h>
#include <WiFiAP.h>
const char* ssid_Router     =   "********";//Modify according to your router name
const char* password_Router =   "********";//Modify according to your router password
const char* ssid_AP         =   "Sunshine";//ESP32 turns on an AP and calls it Sunshine
const char* password_AP     =   "Sunshine";//Set your AP password for ESP32 to Sunshine
WiFiServer server_Cmd(4000);
WiFiServer server_Camera(7000);

void setup() {
```

Modify based on the name and password of your router.

```
12    Serial.begin(115200);
13    Serial.println();
14    WiFi.softAP(ssid_AP, password_AP);//Turn on ESP32's AP feature
15    server_Camera.begin(7000);         //Turn on camera server
16    server_Cmd.begin(4000);            //Turn on Cmd server
17    server_Camera.setNoDelay(true);    //Set no delay in sending and receiving data
18    server_Cmd.setNoDelay(true);       //Set no delay in sending and receiving data
19  ////////////////////////////////////////////////////
20    int timeout=0;
21    while (WiFi.status() != WL_CONNECTED) { //If the connection fails, wait half a second for
      another connection request
22      delay(500);
23      Serial.print(".");
24      timeout++;
25      WiFi.begin(ssid_Router, password_Router);
26      if(timeout==10)
27        break;
28    }
29  ////////////////////////////////////////////////////
30    Serial.print("\nCamera Ready! Use '");
31    Serial.print(WiFi.softAPIP());
32    Serial.print(" or ");
33    Serial.print(WiFi.localIP());
34    Serial.println("' to connect in Freenove app.");
35  }
36
37  void loop() {
38    WiFiClient client = server_Cmd.available();//listen for incoming clients
39    if (client) {                              //if you get a client,
40      Serial.println("Command Server connected to a client.");
41      while (client.connected()) {//loop while the client's connected
42        if (client.available()) { //if there's bytes to read from the client,
43          String dataBuffer = client.readStringUntil('\n') + String("\n"); //read data
44          Serial.print(dataBuffer);//print it out the serial monitor
45        }
46      }
47      client.stop();// close the connection:
48      Serial.println("Command Client Disconnected.");
49    }
50  }
```

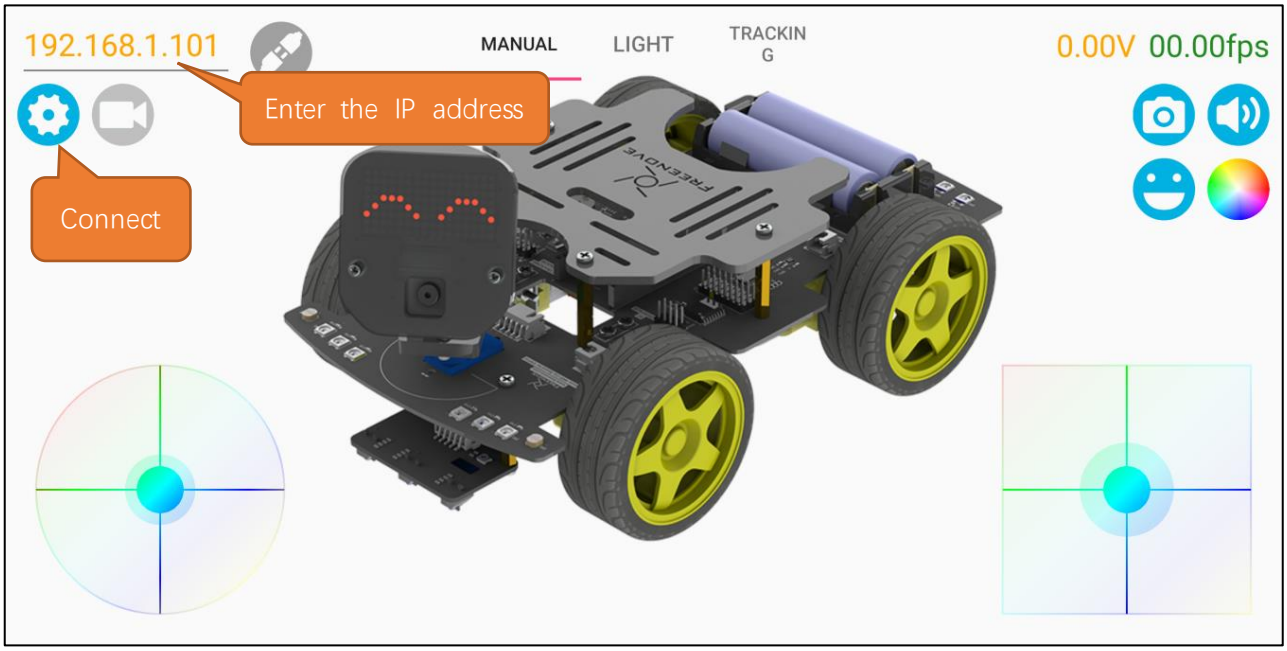
Uplode the code to ESP32 car and open serial monitor.

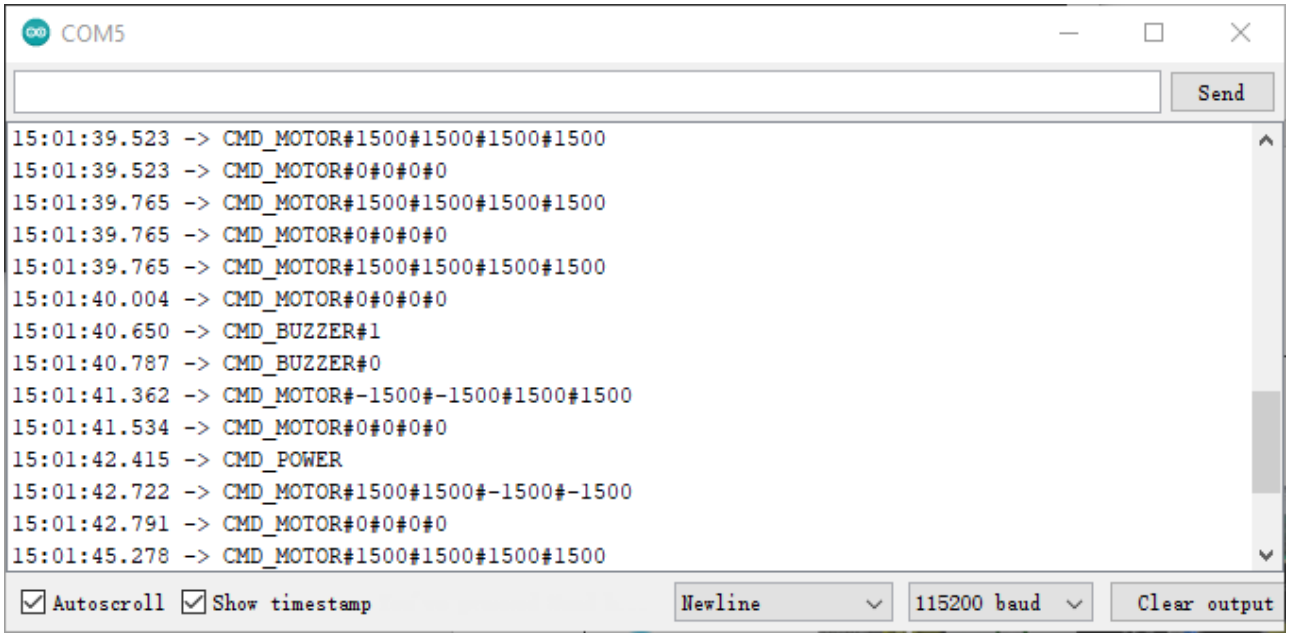




Open mobliephone app and select Freenove 4WD Car for ESP32.

Make sure your mobile phone and ESP32 car are connected to the same router. According to the aforementioned IP address, enter the corresponding IP address, and then click Connect. Tap the button on the screen and you can see the data on the serial monitor.





Code Explanation:

Add the library functions of WiFi. Each time before using WiFi, please add these library functions.

```
#include <WiFi.h>
#include <WiFiClient.h>
#include <WiFiAP.h>
```

Define four pointer variables to store information.

```
const char* ssid_Router     =   "********";//Modify according to your router name
const char* password_Router =   "********";//Modify according to your router password
const char* ssid_AP         =   "Sunshine";//ESP32 turns on an AP and calls it Sunshine
const char* password_AP     =   "Sunshine";//Set your AP password for ESP32 to Sunshine
```

Turn ON AP feature of ESP32. When the function is called, you can search a WiFi signal named “Sunshine” in your phone.

```
  WiFi.softAP(ssid_AP, password_AP);//Turn on ESP32's AP feature
```

The mobile app uses two servers, one for transmitting images and one for transmitting commands. Set ESP32 to start the two servers. The server with port 7000 is used to receive image data (not used in this section for the time being), and the server with port 4000 is used to transmit commands. Call the setNoDelay function to send and receive data directly.

```
  server_Camera.begin(7000);        //Turn on camera server
  server_Cmd.begin(4000);           //Turn on Cmd server
  server_Camera.setNoDelay(true);   //Set no delay in sending and receiving data
  server_Cmd.setNoDelay(true);      //Set no delay in sending and receiving data
```

Connect ESP32 to the router. After the router is successfully connected, ESP32 will print the IP address information to the serial monitor. If there is no router near you, or the router fails to connect, you can delete this code.

```
//////////////////////////////////////////////////////
  int timeout=0;
  while (WiFi.status() != WL_CONNECTED) { //If the connection fails, wait half a second for another connection request
```

```
22      delay(500);
23      Serial.print(".");
24      timeout++;
25      WiFi.begin(ssid_Router, password_Router);
26      if(timeout==10)
27        break;
28    }
29  ////////////////////////////////////////////////////
```

Define a WiFi client object to monitor whether the server has a client request access signal.

```
38  WiFiClient client = server_Cmd.available();//listen for incoming clients
```

Determine whether there is a client connected to the server.

```
42    while (client.connected()) {//loop while the client's connected
```

Call available function to query whether the client has sent data to the server.

```
43      if (client.available()) { //if there's bytes to read from the client,
```

Call readStringUntil function to read a line of data from the client and print it to the serial monitor.

```
44          String dataBuffer = client.readStringUntil('\n') + String("\n"); //read data
45          Serial.print(dataBuffer);//print it out the serial monitor
```

Call stop() function to close the client connection.

```
48      client.stop();// close the connection:
```

Reference

Class Station

Every time when using WiFi, you need to include header file "WiFi.h.".

begin(ssid, password,channel, bssid, connect): ESP32 is used as Station to connect hotspot.

ssid: WiFi hotspot name

password: WiFi hotspot password

channel: WiFi hotspot channel number; communicating through specified channel; optional parameter

bssid: mac address of WiFi hotspot, optional parameter

connect: blloean optional parameter, defaulting to true. If set as false, then ESP32 won't connect WiFi.

config(local_ip, gateway, subnet, dns1, dns2): set static local IP address.

local_ip: station fixed IP address.

subnet: subnet mask

dns1,dns2: optional parameter. define IP address of domain name server

status: obtain the connection status of WiFI

local IP(): obtian IP address in Station mode

disconnect(): disconnect wifi

setAutoConnect(boolen): set automatic connection Every time ESP32 is power on, it will connect WiFi aitomatically.

setAutoReconnect(boolen): set automatic reconnection Every time ESP32 disconnects WiFi, it will reconnect to WiFi automatically.

Class AP

Every time when using WiFi, you need to include header file "WiFi.h.".

softAP(ssid, password, channel, ssid_hidden, max_connection):

ssid: WiFi hotspot name

password: WiFi hotspot password

channel: Number of WiFi connection channels, range 1-13. The default is 1.

ssid_hidden: Whether to hide WiFi name from scanning by other devices. The default is not hide.

max_connection: Maximum number of WiFi connected devices. The range is 1-4. The default is 4.

softAPConfig(local_ip, gateway, subnet): set static local IP address.

local_ip: station fixed IP address.

Gateway: gateway IP address

subnet: subnet mask

softAP(): obtian IP address in AP mode

softAPdisconnect (): disconnect AP mode.

Class Client

Every time when using Client, you need to include header file "WiFi.h."

connect(ip, port, timeout)/connect(*host, port, timeout): establish a TCP connection.

ip, *host: ip address of target server

port: port number of target server

timeout: connection timeout

connected(): judge whether client is connecting. If return value is 1, then connect successfully; If return value is 0, then fail to connect.

stop(): stop tcp connection

print(): send data to server connecting to client

available(): return to the number of bytes readable in receive buffer, if no, return to 0 or -1.

read(): read one byte of data in receive buffer

readString(): read string in receive buffer

Class Server

Every time use Server functionality, we need to include header file"WiFi.h".

WiFiServer(uint16_t port=80, uint8_t max_clients=4): create a TCP Server.

port: ports of Server; range from 0 to 65535 with the default number as 80.

max_clients: maximum number of clients with default number as 4.

begin(port): start the TCP Server.

port: ports of Server; range from 0 to 65535 with the default number as 0.

setNoDelay(bool nodelay): whether to turn off the delay sending functionality.

nodelay: true stands for forbidden Nagle algorithm.

close(): close tcp connection.

stop(): stop tcp connection.

7.2 WiFi Transmitting Images

Earlier we used the WiFi to transmit command. In this section, we use WiFi to transmit image data and display the picture taken by the ESP32 car on the mobile APP.

Sketch

Open the folder "06.2_WiFi_Cam_TcpServer" in the **"Freenove_4WD_Car_Kit_for_ESP32\Sketches"** and double click "06.2_WiFi_Cam_TcpServer.ino".

Before uploading the code, please modify the name and password of Router. Other operation is the same as Section 7.1.

Code

```
#include "esp_camera.h"
#include <WiFi.h>
#include <WiFiClient.h>
#include <WiFiAP.h>

#define CAMERA_MODEL_WROVER_KIT
#include "camera_pins.h"

const char* ssid_Router     =   "********";//Modify according to your router name
const char* password_Router =   "********";//Modify according to your router password
const char* ssid_AP         =   "Sunshine";//ESP32 turns on an AP and calls it Sunshine
const char* password_AP     =   "Sunshine";//Set your AP password for ESP32 to Sunshine

WiFiServer server_Cmd(4000);
WiFiServer server_Camera(7000);
extern TaskHandle_t loopTaskHandle;

void setup() {
    Serial.begin(115200);
    Serial.setDebugOutput(false);
    Serial.println();
    cameraSetup();

    WiFi.softAP(ssid_AP, password_AP);
    server_Camera.begin(7000);
    server_Cmd.begin(4000);
    ////////////////////////////////////////////////
    WiFi.begin(ssid_Router, password_Router);
    Serial.print("Connecting");
```

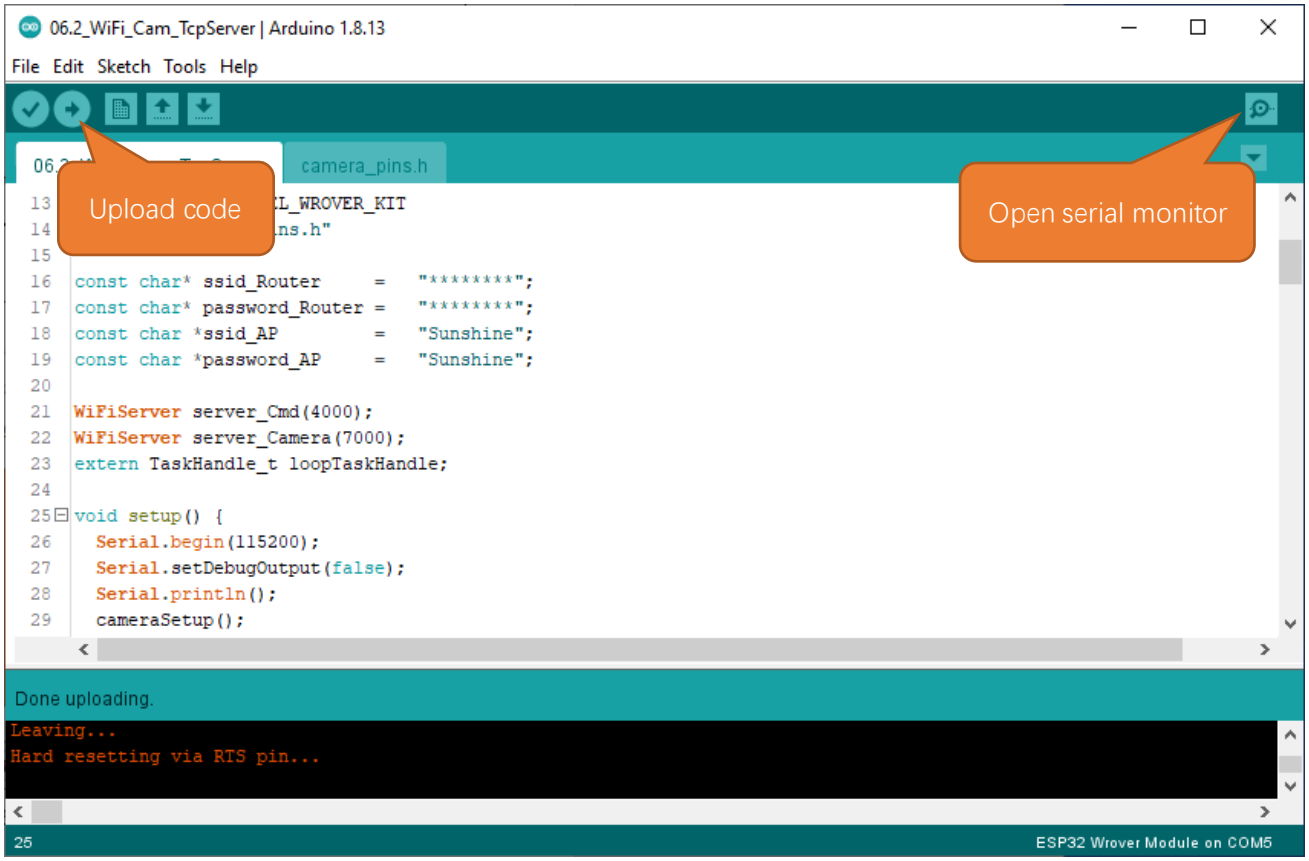
Please modify the name and password of Router.

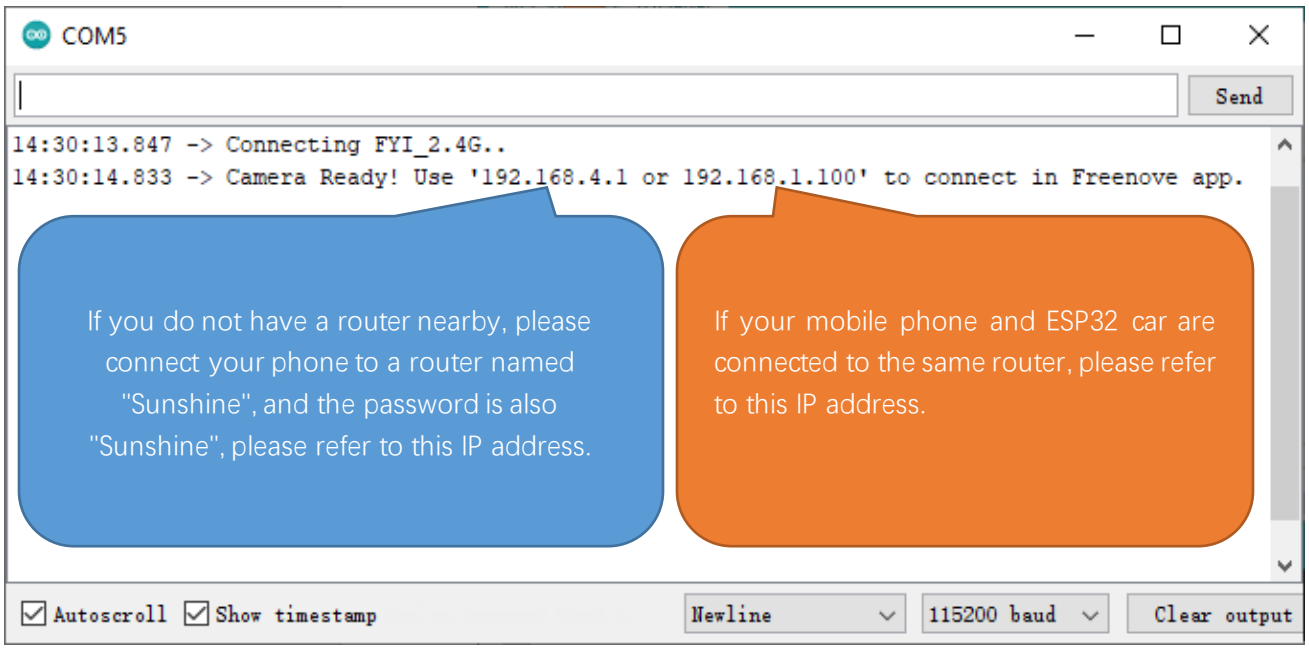
```
30      Serial.print(ssid_Router);
31      int timeout=0;
32      while (WiFi.status() != WL_CONNECTED) {
33          delay(500);
34          Serial.print(".");
35          WiFi.begin(ssid_Router, password_Router);
36      timeout++;
37      if(timeout>=10)
38      break;
39      }
40      Serial.println("");
41      Serial.println("WiFi connected");
42      ////////////////////////////////////////////////
43      Serial.print("Camera Ready! Use '");
44      Serial.print(WiFi.softAPIP());
45      Serial.print(" or ");
46      Serial.print(WiFi.localIP());
47      Serial.println("' to connect in Freenove app.");
48
49      disableCore0WDT();
50      //loopTask_Cmd uses core 0.
51      xTaskCreateUniversal(loopTask_Cmd, "loopTask_Cmd", 8192, NULL, 1, &loopTaskHandle, 0);
52  }
53
54  //task loop uses core 1.
55  void loop() {
56      WiFiClient client = server_Camera.available();// listen for incoming clients
57      if (client) {                                 // if you get a client,
58          Serial.println("Camera Server connected to a client.");
59          String currentLine = "";       // make a String to hold incoming data from the client
60          while (client.connected()) {   // loop while the client's connected
61              camera_fb_t * fb = NULL;
62              while (client.connected()) {
63                  fb = esp_camera_fb_get();
64                  if (fb != NULL) {
65                      uint8_t slen[4];
66                      slen[0] = fb->len >> 0;
67                      slen[1] = fb->len >> 8;
68                      slen[2] = fb->len >> 16;
69                      slen[3] = fb->len >> 24;
70                      client.write(slen, 4);
71                      client.write(fb->buf, fb->len);
72                  }
73                  else {
```

```
74                      Serial.println("Camera Error");
75                  }
76              }
77          }
78          // close the connection:
79          client.stop();
80          Serial.println("Camera Client Disconnected.");
81      }
82  }
83
84  void loopTask_Cmd(void *pvParameters) {
85      Serial.println("Task Cmd_Server is starting... ");
86      while (1) {
87          WiFiClient client = server_Cmd.available();//listen for incoming clients
88          if (client)                                 //if you get a client,
89          {
90              Serial.println("Command Server connected to a client.");
91              while (client.connected()) {//loop while the client's connected
92                  if (client.available()) { //if there's bytes to read from the client,
93                      String dataBuffer = client.readStringUntil('\n') + String("\n");
94                      Serial.print(dataBuffer);//print it out the serial monitor
95                  }
96              }
97              client.stop();// close the connection:
98              Serial.println("Command Client Disconnected.");
99          }
100     }
101 }
102
103 void cameraSetup() {
104     camera_config_t config;
105     config.ledc_channel = LEDC_CHANNEL_0;
106     config.ledc_timer = LEDC_TIMER_0;
107     config.pin_d0 = Y2_GPIO_NUM;
108     config.pin_d1 = Y3_GPIO_NUM;
109     config.pin_d2 = Y4_GPIO_NUM;
110     config.pin_d3 = Y5_GPIO_NUM;
111     config.pin_d4 = Y6_GPIO_NUM;
112     config.pin_d5 = Y7_GPIO_NUM;
113     config.pin_d6 = Y8_GPIO_NUM;
114     config.pin_d7 = Y9_GPIO_NUM;
115     config.pin_xclk = XCLK_GPIO_NUM;
116     config.pin_pclk = PCLK_GPIO_NUM;
117     config.pin_vsync = VSYNC_GPIO_NUM;
```

```
118       config.pin_href = HREF_GPIO_NUM;
119       config.pin_sscb_sda = SIOD_GPIO_NUM;
120       config.pin_sscb_scl = SIOC_GPIO_NUM;
121       config.pin_pwdn = PWDN_GPIO_NUM;
122       config.pin_reset = RESET_GPIO_NUM;
123       config.xclk_freq_hz = 20000000;
124       config.pixel_format = PIXFORMAT_JPEG;
125       config.frame_size = FRAMESIZE_VGA;   //clear
126   //  config.frame_size = FRAMESIZE_QVGA;  //ordinary
127   //  config.frame_size = FRAMESIZE_QQVGA;//concision
128       config.jpeg_quality = 10;
129       config.fb_count = 1;
130
131       // camera init
132       esp_err_t err = esp_camera_init(&config);
133       if (err != ESP_OK) {
134           Serial.printf("Camera init failed with error 0x%x", err);
135           return;
136       }
137       Serial.println("Camera configuration complete!");
138   }
```

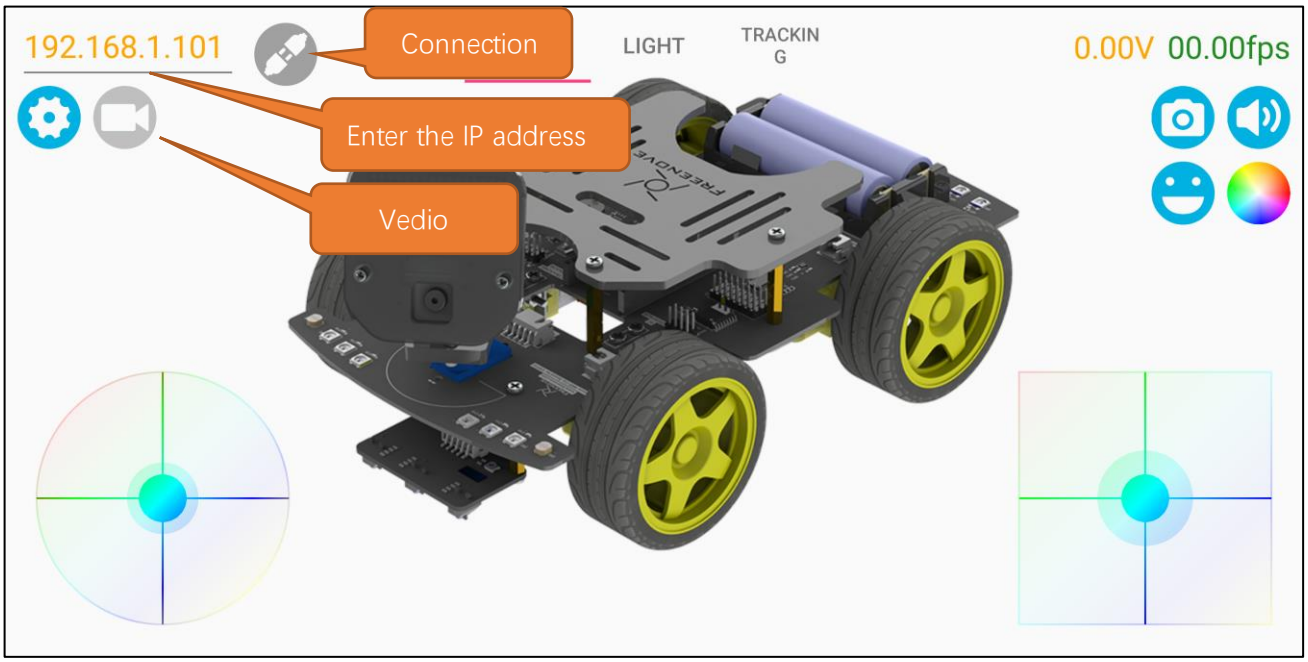
Upload code to ESP32 car and open serial monitor.

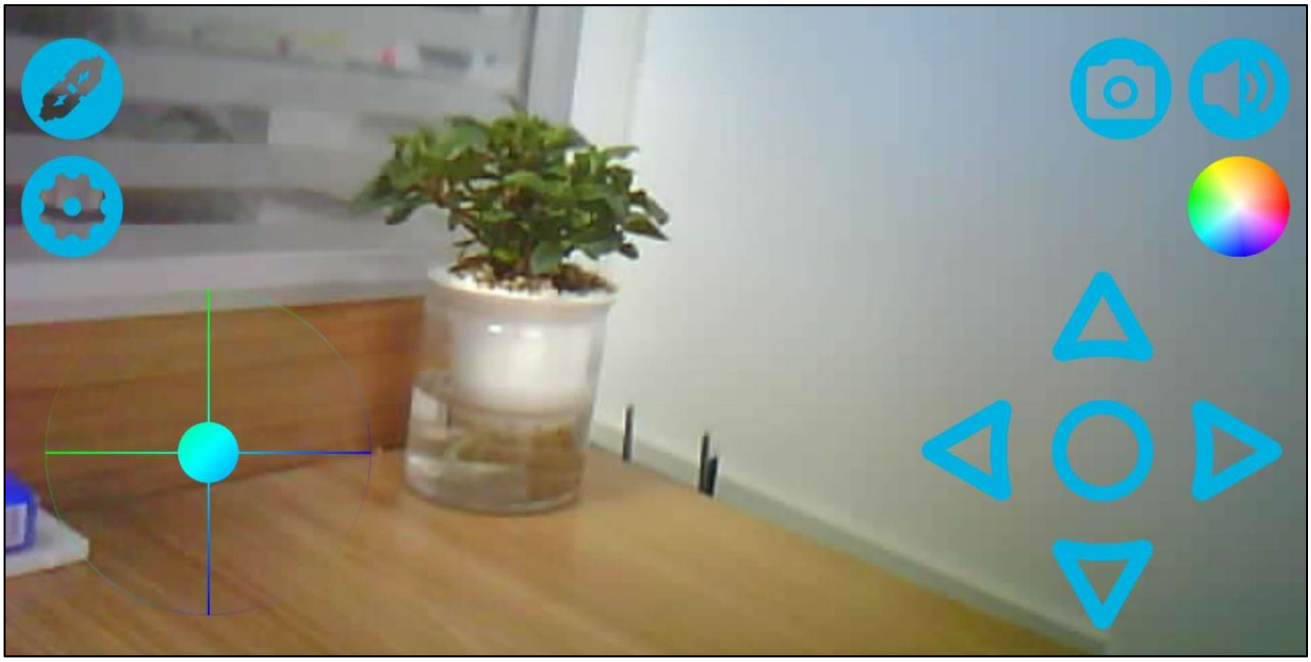




Open mobile APP and select Freenove 4WD Car for ESP32.

Make sure your mobile phone and ESP32 car are connected to the same router. According to the aforementioned IP address, enter the corresponding IP address, and then click Connect. Tap the button on the screen and you can see the data on the serial monitor.





Code Explanation:

Before using camera, please add two header files esp_camera.h and camera_pins.h .

```
#include "esp_camera.h"
#include <WiFi.h>
#include <WiFiClient.h>
#include <WiFiAP.h>

#define CAMERA_MODEL_WROVER_KIT
#include "camera_pins.h"
```

Define four pointer variables to store information. Each time, before using, please modify name and password of Router.

```
5  const char* ssid_Router     =   "********";//Modify according to your router name
6  const char* password_Router =   "********";//Modify according to your router password
7  const char* ssid_AP         =   "Sunshine";//ESP32 turns on an AP and calls it Sunshine
8  const char* password_AP     =   "Sunshine";//Set your AP password for ESP32 to Sunshine
```

Camara initialization function that assign pins to camera and sets the camera clock frequency, picture quality, picture size and other information.

```
103 void cameraSetup() {
104     camera_config_t config;
105     config.ledc_channel = LEDC_CHANNEL_0;
106     config.ledc_timer = LEDC_TIMER_0;
107     config.pin_d0 = Y2_GPIO_NUM;
108     config.pin_d1 = Y3_GPIO_NUM;
109     config.pin_d2 = Y4_GPIO_NUM;
110     config.pin_d3 = Y5_GPIO_NUM;
111     config.pin_d4 = Y6_GPIO_NUM;
112     config.pin_d5 = Y7_GPIO_NUM;
113     config.pin_d6 = Y8_GPIO_NUM;
114     config.pin_d7 = Y9_GPIO_NUM;
115     config.pin_xclk = XCLK_GPIO_NUM;
116     config.pin_pclk = PCLK_GPIO_NUM;
117     config.pin_vsync = VSYNC_GPIO_NUM;
118     config.pin_href = HREF_GPIO_NUM;
119     config.pin_sscb_sda = SIOD_GPIO_NUM;
120     config.pin_sscb_scl = SIOC_GPIO_NUM;
121     config.pin_pwdn = PWDN_GPIO_NUM;
122     config.pin_reset = RESET_GPIO_NUM;
123     config.xclk_freq_hz = 20000000;
124     config.pixel_format = PIXFORMAT_JPEG;
125     config.frame_size = FRAMESIZE_VGA;   //clear
126 //  config.frame_size = FRAMESIZE_QVGA;  //ordinary
127 //  config.frame_size = FRAMESIZE_QQVGA; //concision
128     config.jpeg_quality = 10;
129     config.fb_count = 1;
130     esp_err_t err = esp_camera_init(&config); // camera init
131     if (err != ESP_OK) {
132         Serial.printf("Camera init failed with error 0x%x", err);
133         return;
134     }
135     Serial.println("Camera configuration complete!");
136   }
```

Set frame_size. VGA, QVGA, and QQVGA correspond to different definitions of the pictures.

ESP32 is a dual-core processor, which can handle different things at the same time. Usually the loop function of ESP32 runs in the first core area. In this code, we make the command processing run in the 0th core area and the camera data processing in the first core area. Call the disableCore0WDT() function to turn off the watchdog in the 0th core area to avoid program reset caused by the watchdog. Call the xTaskCreateUniversal function to apply to create a task function and let the task function run in the 0th core area.

```
49      disableCore0WDT();
50      //loopTask_Cmd uses core 0.
51      xTaskCreateUniversal(loopTask_Cmd, "loopTask_Cmd", 8192, NULL, 1, &loopTaskHandle, 0);
```

Define a camera image data storage object to store image data.

```
61    camera_fb_t * fb = NULL;
```

Call the esp_camera_fb_get() function to get the image data and store it in fb. Call the write() function to send the data to the mobile app.

```
63      fb = esp_camera_fb_get();
64      if (fb != NULL) {
65          uint8_t slen[4];
66          slen[0] = fb->len >> 0;
67          slen[1] = fb->len >> 8;
68          slen[2] = fb->len >> 16;
69          slen[3] = fb->len >> 24;
70          client.write(slen, 4);
71          client.write(fb->buf, fb->len);
72      }
```

Reference

Image resolution	Sharpness	Image resolution	Sharpness
FRAMESIZE_QQVGA	160x120	FRAMESIZE_VGA	640x480
FRAMESIZE_QQVGA2	128x160	FRAMESIZE_SVGA	800x600
FRAMESIZE_QCIF	176x144	FRAMESIZE_XGA	1024x768
FRAMESIZE_HQVGA	240x176	FRAMESIZE_SXGA	1280x1024
FRAMESIZE_QVGA	320x240	FRAMESIZE_UXGA	1600x1200
FRAMESIZE_CIF	400x296	FRAMESIZE_QXGA	2048x1536

7.3 WiFi Video Car by APP

In this section, we will combine image transmission with controlling the car. You can control the ESP32 car through mobile APP while observing the images that the camera transmits.

Sketch

Open the folder "06.3_Multi_Functional_Car" in Freenove_4WD_Car_Kit_for_ESP32\Sketches and double click "06.3_Multi_Functional_Car.ino".
Before compiling the code, please modify the code according to the name and password of your WiFi.
If you want to enable ESP32's own router functionality, WiFi_Setup(1); If you want ESP32 to connect to the router, WiFi_Setup(0).

Code

```
#include <Arduino.h>
#include <WiFi.h>
#include <WiFiClient.h>
#include <WiFiAP.h>
#include "esp_camera.h"
#include "Freenove_4WD_Car_WiFi.h"
#include "Freenove_4WD_Car_Emotion.h"
#include "Freenove_4WD_Car_WS2812.h"
#include "Freenove_4WD_Car_For_ESP32.h"

String CmdArray[8];
int paramters[8];
bool videoFlag = 0;

void WiFi_Init(){
  ssid_Router     =  "********";    //Modify according to your router name
  password_Router =  "********";    //Modify according to your router password
  ssid_AP         =  "Sunshine";    //ESP32 turns on an AP and calls it Sunshine
  password_AP     =  "Sunshine";    //Set your AP password for ESP32 to Sunshine
  frame_size      =   FRAMESIZE_CIF;//400*296
}

WiFiServer server_Cmd(4000);
WiFiServer server_Camera(7000);

void setup(){
  Buzzer_Setup();           //Buzzer initialization
  Serial.begin(115200);
  Serial.setDebugOutput(true);
```

Please modify the name and password of Router.

```
30    WiFi_Init();                  //WiFi paramters initialization
31    WiFi_Setup(1);                //Start AP Mode
32    server_Camera.begin(7000);//Turn on the camera server
33    server_Cmd.begin(4000);    //Start the command server
34
35    cameraSetup();                //Camera initialization
36    Emotion_Setup();              //Emotion initialization
37    WS2812_Setup();               //WS2812 initialization
38    PCA9685_Setup();              //PCA9685 initialization
39    Light_Setup();                //Light initialization
40    Track_Setup();                //Track initialization
41
42    disableCoreOWDT();            //Turn off the watchdog function in kernel 0
43    xTaskCreateUniversal(loopTask_Camera, "loopTask_Camera", 8192, NULL, 0, NULL, 0);
44    xTaskCreateUniversal(loopTask_WTD, "loopTask_WTD", 8192, NULL, 0, NULL, 0);
45  }
46
47  void loop() {
48    WiFiClient client = server_Cmd.available();//listen for incoming clients
49    if (client) {                               //if you get a client
50      Serial.println("Cmd_Server connected to a client.");
51      while (client.connected()) {              //loop while the client's connected
52        if (client.available()) {               //if there's bytes to read from the client
53          String inputStringTemp = client.readStringUntil('\n');//Read the command by WiFi
54          Serial.println(inputStringTemp);      //Print out the command received by WiFi
55          Get_Command(inputStringTemp);
56
57          if (CmdArray[0] == CMD_LED_MOD)//Set the display mode of car colored lights
58            WS2812_SetMode(paramters[1]);
59          if (CmdArray[0] == CMD_LED) //Set the color and brightness of the car lights
60            WS2812_Set_Color_1(paramters[1], paramters[3], paramters[2], paramters[4]);
61          if (CmdArray[0] == CMD_MATRIX_MOD)//Set the display mode of the LED matrix
62            Emotion_SetMode(paramters[1]);
63          if (CmdArray[0] == CMD_VIDEO)//Video transmission command
64            videoFlag = paramters[1];
65          if (CmdArray[0] == CMD_BUZZER) //Buzzer control command
66            Buzzer_Variable(paramters[1], paramters[2]);
67          if (CmdArray[0] == CMD_POWER) {//Power query command
68            float battery_voltage = Get_Battery_Voltage();
69            client.print(CMD_POWER);
70            client.print(INTERVAL_CHAR);
71            client.print(battery_voltage);
72            client.print(ENTER);
73          }
```

```
74          if (CmdArray[0] == CMD_MOTOR) {//Network control car movement command
75            Car_SetMode(0);
76            if (paramters[1] == 0 && paramters[3] == 0)
77              Motor_Move(0, 0, 0, 0);//Stop the car
78            else //If the parameters are not equal to 0
79              Motor_Move(paramters[1], paramters[1], paramters[3], paramters[3]);
80          }
81          if (CmdArray[0] == CMD_SERVO) {//Network control servo motor movement command
82            if (paramters[1] == 0)
83              Servo_1_Angle(paramters[2]);
84            else if (paramters[1] == 1)
85              Servo_2_Angle(paramters[2]);
86          }
87          if (CmdArray[0] == CMD_CAMERA) {//Network control servo motor movement command
88            Servo_1_Angle(paramters[1]);
89            Servo_2_Angle(paramters[2]);
90          }
91          if (CmdArray[0] == CMD_LIGHT) { //Light seeking car command
92            if (paramters[1] == 1)
93              Car_SetMode(1);
94            else if (paramters[1] == 0)
95              Car_SetMode(0);
96          }
97          else if (CmdArray[0] == CMD_TRACK) { //Tracking car command
98            if (paramters[1] == 1)
99              Car_SetMode(2);
100           else if (paramters[1] == 0)
101             Car_SetMode(0);
102         }
103         if (CmdArray[0] == CMD_CAR_MODE) { //Car command Mode
104           Car_SetMode(paramters[1]);
105         }
106         //Clears the command array and parameter array
107         memset(CmdArray, 0, sizeof(CmdArray));
108         memset(paramters, 0, sizeof(paramters));
109       }
110       Emotion_Show(emotion_task_mode);//Led matrix display function
111       WS2812_Show(ws2812_task_mode);//Car color lights display function
112       Car_Select(carFlag);//ESP32 Car mode selection function
113     }
114     client.stop();//close the connection:
115     Serial.println("Command Client Disconnected.");
116     ESP.restart();
117   }
```

```
}
void loopTask_Camera(void *pvParameters) {
  while (1) {
    WiFiClient client = server_Camera.available();//listen for incoming clients
    if (client) {//if you get a client
      Serial.println("Camera_Server connected to a client.");
      if (client.connected()) {
        camera_fb_t * fb = NULL;
        while (client.connected()) {//loop while the client's connected
          if (videoFlag == 1) {
            fb = esp_camera_fb_get();
            if (fb != NULL) {
              uint8_t slen[4];
              slen[0] = fb->len >> 0;
              slen[1] = fb->len >> 8;
              slen[2] = fb->len >> 16;
              slen[3] = fb->len >> 24;
              client.write(slen, 4);
              client.write(fb->buf, fb->len);
              Serial.println("Camera send");
            }
          }
        }
        //close the connection:
        client.stop();
        Serial.println("Camera Client Disconnected.");
        ESP.restart();
      }
    }
  }
}
void Get_Command(String inputStringTemp){
  int string_length = inputStringTemp.length();
  for (int i = 0; i < 8; i++) {//Parse the command received by WiFi
    int index = inputStringTemp.indexOf(INTERVAL_CHAR);
    if (index < 0) {
      if (string_length > 0) {
        CmdArray[i] = inputStringTemp;          //Get command
        paramters[i] = inputStringTemp.toInt();//Get parameters
      }
      break;
    }
    else {
      string_length -= index;                                  //Count the remaining words
```

```
        CmdArray[i] = inputStringTemp.substring(0, index);      //Get command
        paramters[i] = CmdArray[i].toInt();                     //Get parameters
        inputStringTemp = inputStringTemp.substring(index + 1);//Update string
      }
    }
}
```

Code Explanation:

Add the driver headr files of the car.

The three header files starting with "WiFi" contain the WiFi driver configuration function of ESP32.

The esp_camera.h file contains the control function of ESP32 camera.

Freenove_4WD_Car_WiFi.h contains the command information when WiFi transmits signal and the ESP32 pin configuration of the camera.

Freenove_4WD_Car_Emotion.h contains the driver configuration function of the LED matrix module.

Freenove_4WD_Car_WS2812.h contains the driver configuration function of the RGB LEDs.

Freenove_4WD_Car_For_ESP32.h contains the driver configuration function of the car.

```
#include <Arduino.h>
#include <WiFi.h>
#include <WiFiClient.h>
#include <WiFiAP.h>
#include "esp_camera.h"
#include "Freenove_4WD_Car_WiFi.h"
#include "Freenove_4WD_Car_Emotion.h"
#include "Freenove_4WD_Car_WS2812.h"
#include "Freenove_4WD_Car_For_ESP32.h"
```

Before you upload the code each time, you can change the ssid_Router and password_Router below based on your WiFi name and password. The program uses AP mode by default. ESP32 will create a WiFi called "Sunshine" based on the WiFi configuration.

Note: "Sunshine" is not connected to the Internet. So if you choose to connect to it, your phone will be disconnected from the Internet. “frame_size” is used to configure the pixel size of the camera. If you don’t know its parameters, you can click here.

```
  ssid_Router     =   "********";    //Modify according to your router name
  password_Router =   "********";    //Modify according to your router password
  ssid_AP         =   "Sunshine";    //ESP32 turns on an AP and calls it Sunshine
  password_AP     =   "Sunshine";    //Set your AP password for ESP32 to Sunshine
  frame_size      =    FRAMESIZE_CIF;//400*296
```

Turn ON two TCP server ports. Port 4000 is used to receive the command from the car and port 7000 is to receive the image data collected by camera.

```
WiFiServer server_Cmd(4000);
WiFiServer server_Camera(7000);
```

Call the WiFi_Init() to initialize the WiFi parameters. Call the WiFi_Setup() to initialize the ESP32's WiFi.

AP mode starts when the WiFi_Setup parameter is 1. When the parameter WiFi_Setup is 0, the STA mode is started. After ESP32 starts WIFI, it prints the IP information to the serial monitor.

```
  WiFi_Init();              //WiFi paramters initialization
  WiFi_Setup(1);            //Start AP Mode
```

Initialize the car's buzzer, LED matrix module, RGB LEDs, WiFi, camera, motor motor, servo, photoresistor and line tracking modules.

```
26  void setup(){
27    Buzzer_Setup();           //Buzzer initialization
28    Serial.begin(115200);
29    Serial.setDebugOutput(true);
30    WiFi_Init();              //WiFi paramters initialization
31    WiFi_Setup();             //WiFi initialization
32    server_Camera.begin(7000);//Turn on the camera server
33    server_Cmd.begin(4000);   //Start the command server
34
35    cameraSetup();            //Camera initialization
36    Emotion_Setup();          //Emotion initialization
37    WS2812_Setup();           //WS2812 initialization
38    PCA9685_Setup();          //PCA9685 initialization
39    Light_Setup();            //Light initialization
40    Track_Setup();            //Track initialization
41
42    disableCore0WDT();        //Turn off the watchdog function in kernel 0
43    xTaskCreateUniversal(loopTask_Camera, "loopTask_Camera", 8192, NULL, 0, NULL, 0);
44    xTaskCreateUniversal(loopTask_WTD, "loopTask_WTD", 8192, NULL, 0, NULL, 0);
45  }
```

Monitor TCP server port 4000. If a client connects to this port, ESP32 will print prompt messages of WiFi connection on the serial port. After the client connects to this port, it receives information from the client until the client disconnects. When ESP32 detects that the client is disconnected, it closes the TCP connection and prints a WiFi disconnect prompt message.

```
45    WiFiClient client = server_Cmd.available();//listen for incoming clients
46    if (client) {//if you get a client
47      Serial.println("Cmd_Server connected to a client.");
48      while (client.connected()) {//loop while the client's connected
...       ...
114     }
115     client.stop();//close the connection:
116     Serial.println("Command Client Disconnected.");
117     ESP.restart();
118   }
```

Get_Command(), the command parsing function. Every time this function is used, the command received by TCP will be sent to this function through inputStringTemp, and the command and parameters will be parsed and stored in the global variable array CmdArray and paramters.

```
149 void Get_Command(String inputStringTemp){
150   int string_length = inputStringTemp.length();
151   for (int i = 0; i < 8; i++) {//Parse the command received by WiFi
152     int index = inputStringTemp.indexOf(INTERVAL_CHAR);
153     if (index < 0) {
```

```
154         if (string_length > 0) {
155           CmdArray[i] = inputStringTemp;          //Get command
156           paramters[i] = inputStringTemp.toInt();//Get parameters
157         }
158         break;
159       }
160       else {
161         string_length -= index;                                  //Count the remaining words
162         CmdArray[i] = inputStringTemp.substring(0, index);       //Get command
163         paramters[i] = CmdArray[i].toInt();                      //Get parameters
164         inputStringTemp = inputStringTemp.substring(index + 1);//Update string
165       }
166     }
167 }
```

Judge the client's data and use different functions of the car according to different commands.

```
57          if (CmdArray[0] == CMD_LED_MOD)//Set the display mode of car colored lights
…             …
59          if (CmdArray[0] == CMD_LED) {//Set the color and brightness of the car lights
…             …
61          if (CmdArray[0] == CMD_MATRIX_MOD)//Set the display mode of the LED matrix
…             …
63          if (CmdArray[0] == CMD_VIDEO)//Video transmission command
…             …
65          if(CmdArray[0] == CMD_BUZZER)//Buzzer control command
…             …
74          if (CmdArray[0] == CMD_POWER) {//Power query command
…             …
81          if (CmdArray[0] == CMD_MOTOR) {//Network control car movement command
…             …
87          if (CmdArray[0] == CMD_SERVO) {//Network control servo motor movement command
…             …
91          if(CmdArray[0] == CMD_LIGHT) {//Light seeking car command
…             …
97          if(CmdArray[0] == CMD_TRACK) {//Tracking car command
…             …
103         if (CmdArray[0] == CMD_CAR_MODE) { //Car command Mode
…             …
```

The following three functions run in a non-blocking manner. Call the Emotion_Show() function to control the LED matrix, call the WS2812_Show() function to control the RGB LEDs of the car, and call the Car_Select() function to control the mode of the car.

```
100         Emotion_Show(emotion_task_mode);//Led matrix display function
101         WS2812_Show(ws2812_task_mode);//Car color lights display function
102         Car_Select(carFlag);//ESP32 Car mode selection function
```

The camera thread callback function. Each time a connection is established with the client, if a video

transmission command (videoFlag=1) is received, the image will be obtained and sent to the client.

```
119 void loopTask_Camera(void *pvParameters) {
120   while (1) {
121     WiFiClient client = server_Camera.available();//listen for incoming clients
122     if (client) {//if you get a client
123       Serial.println("Camera_Server connected to a client.");
124       if (client.connected()) {
125         camera_fb_t * fb = NULL;
126         while (client.connected()) {//loop while the client's connected
127           if (videoFlag == 1) {
128             fb = esp_camera_fb_get();
129             if (fb != NULL) {
130               uint8_t slen[4];
131               slen[0] = fb->len >> 0;
132               slen[1] = fb->len >> 8;
133               slen[2] = fb->len >> 16;
134               slen[3] = fb->len >> 24;
135               client.write(slen, 4);
136               client.write(fb->buf, fb->len);
137               Serial.println("Camera send");
138             }
139           }
140         }
141         //close the connection:
142         client.stop();
143         Serial.println("Camera Client Disconnected.");
144         ESP.restart();
145       }
146     }
147   }
148 }
```

7.4 WiFi Video Car by PC

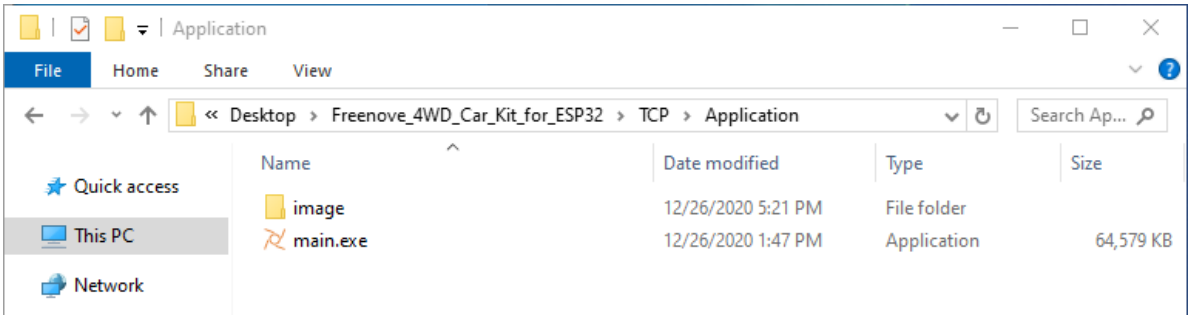
In addition to using a mobile phone to control the WiFi video car, we also provide users with a host computer to control the car. In this chapter, the car still uses the code in section 7.3

Run client on windows system

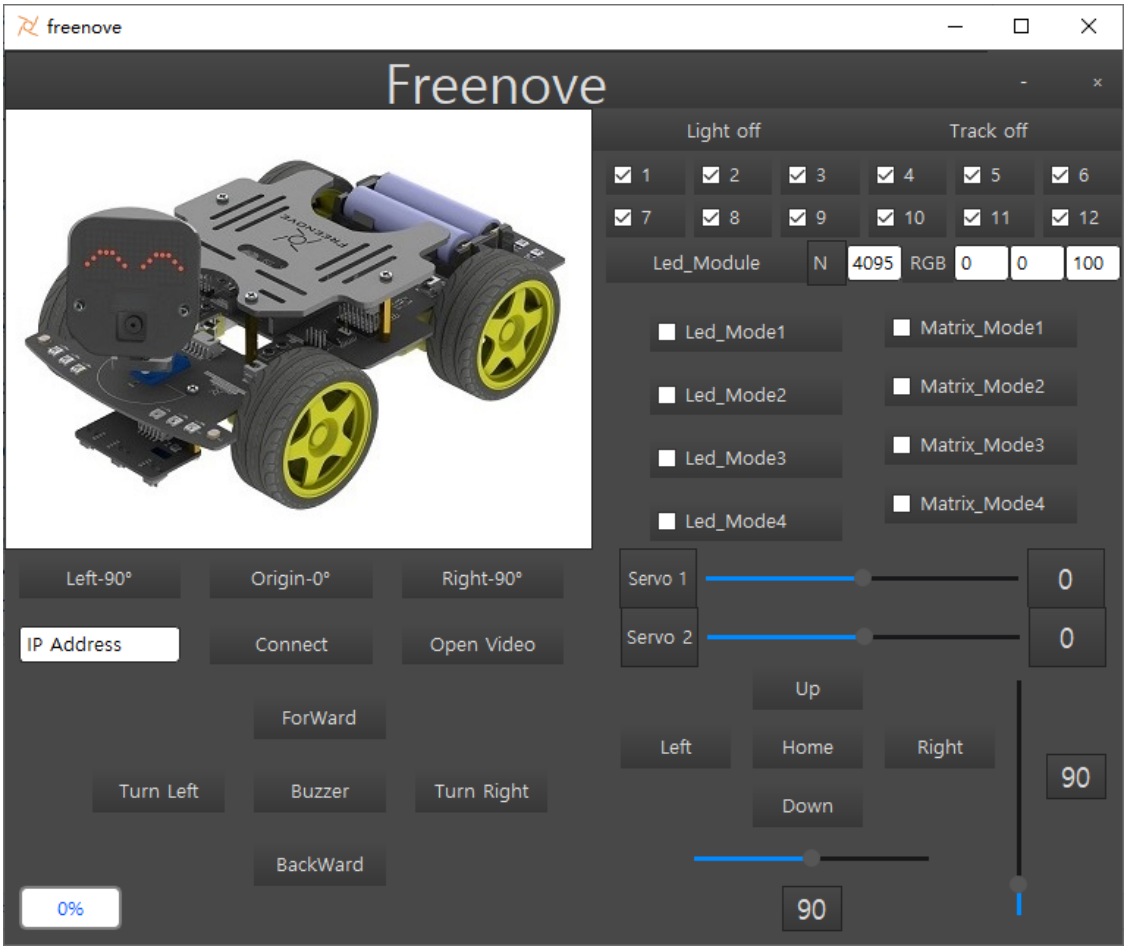
There are two ways to run client on Windows.

Option 1 Run the executable file directly.

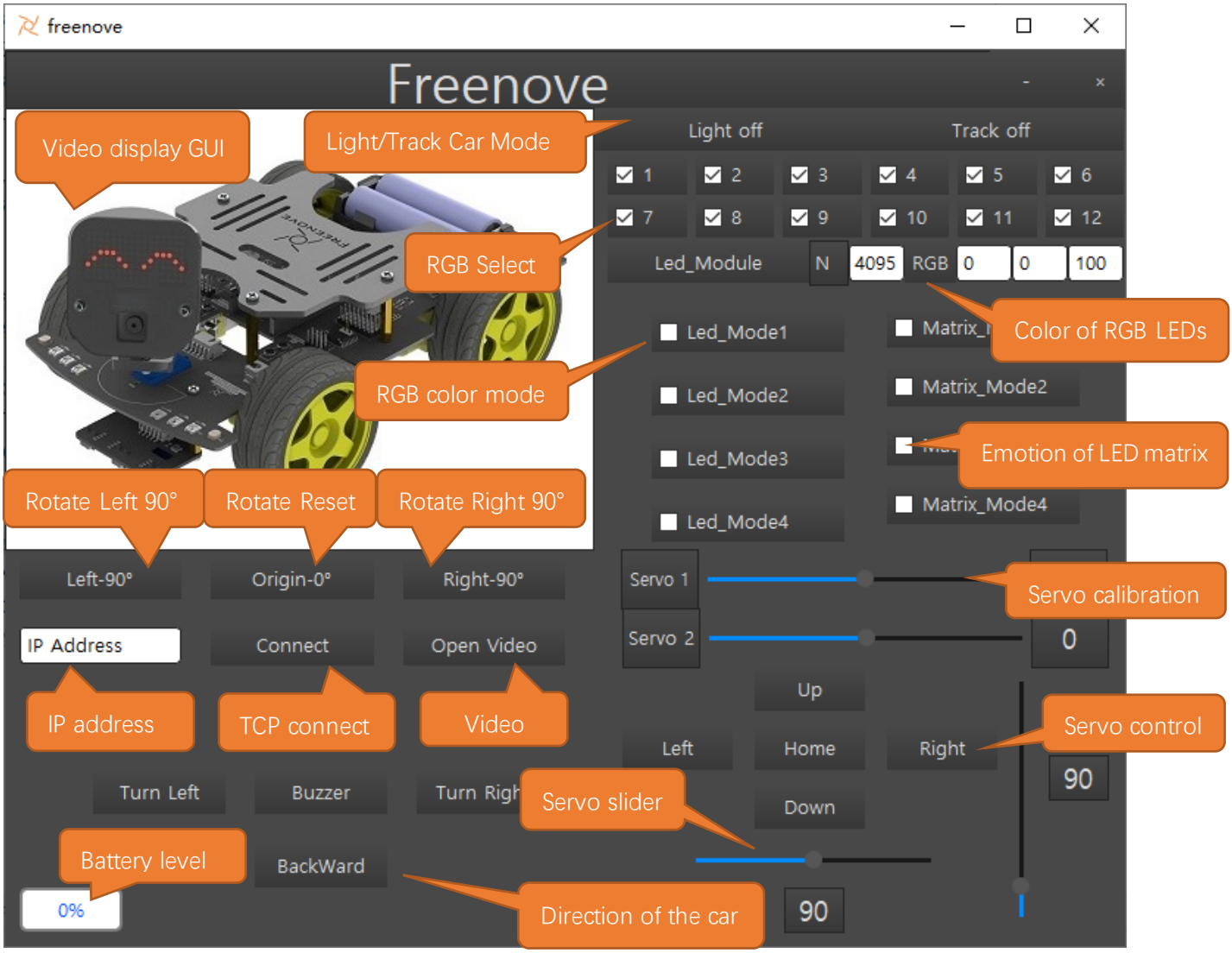
Open main.exe in Freenove_4WD_Car_Kit_for_ESP32\TCP\Application.



The interface of client is as follows:



The function of the client is as follows：



According to the prompt message printed on the ESP32 serial monitor, enter the IP address into "IP Address" and click "Connect", and you can control the ESP32 car through this client

Option 2 Install python3and some related python libraries to run client.

If you want to modify the client, please following this section.

Install python3

Download and install Python3 package.

https://www.python.org/downloads/windows/

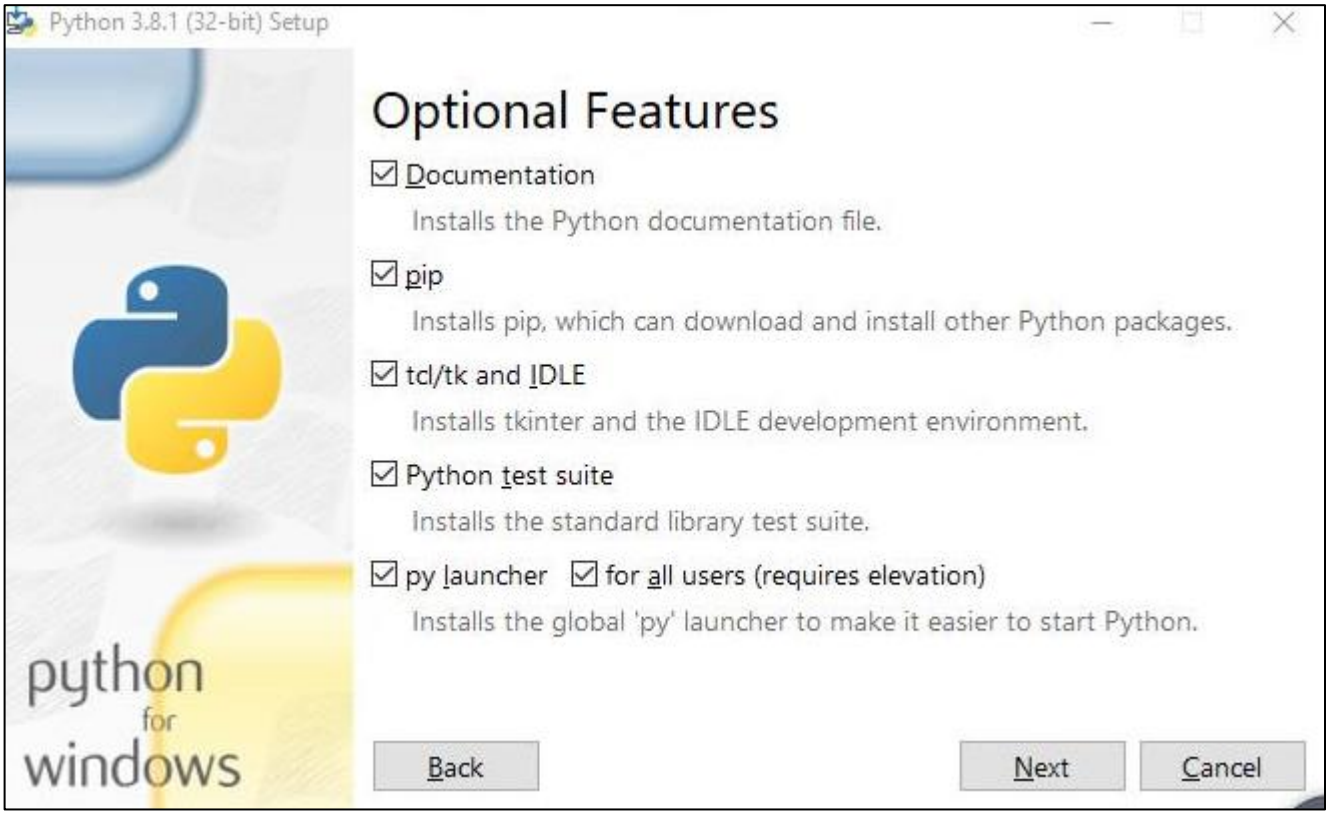
Select Python3.8.1.

Version	Operating System	Description
Gzipped source tarball	Source release	
XZ compressed source tarball	Source release	
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later
Windows help file	Windows	
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64
Windows x86 embeddable zip file	Windows	
Windows x86 executable installer	Windows	
Windows x86 web-based installer	Windows	

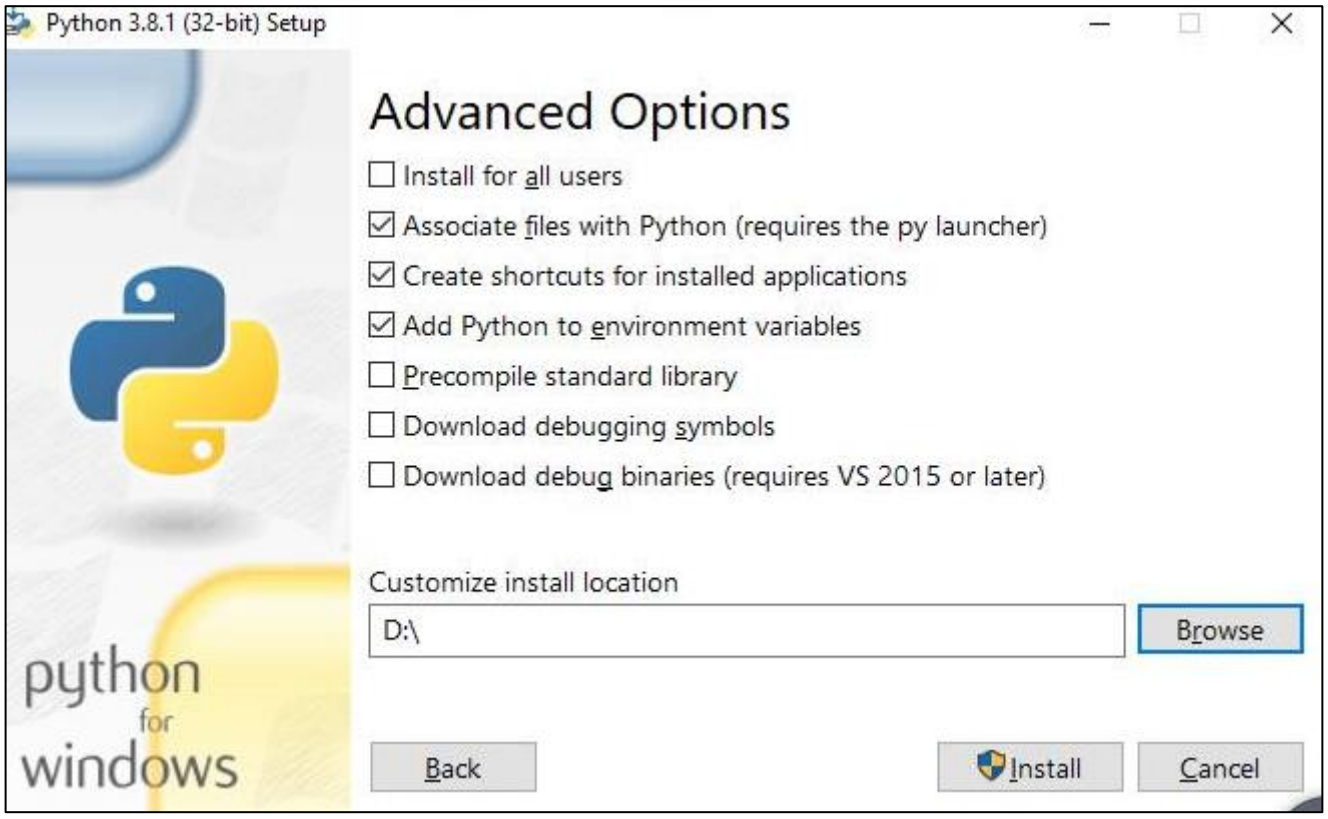
Click Windows x86 executable installerto download. After downloading, click to install.

Please note that "Add Python 3.8 to PATH" MUST be check.

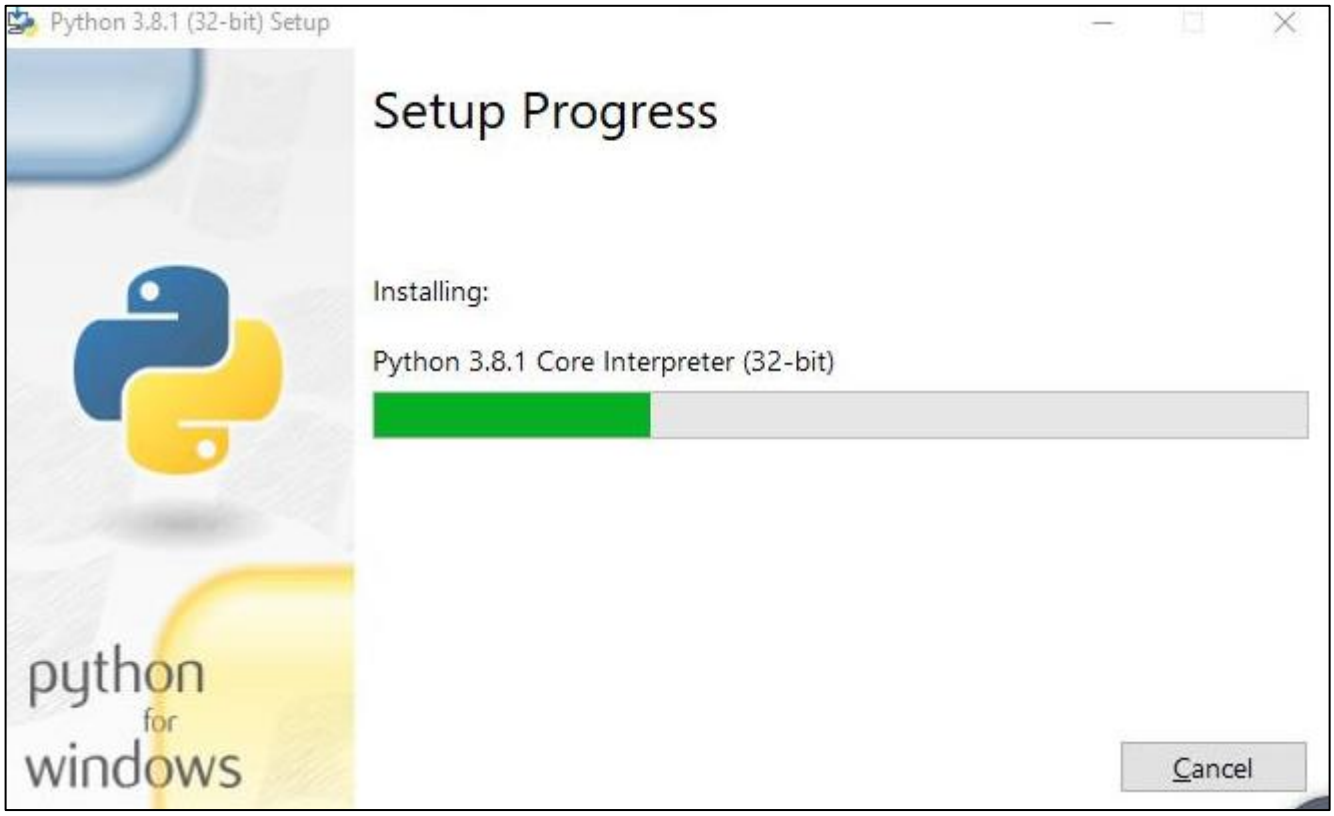
Check all the options and then click "Next".



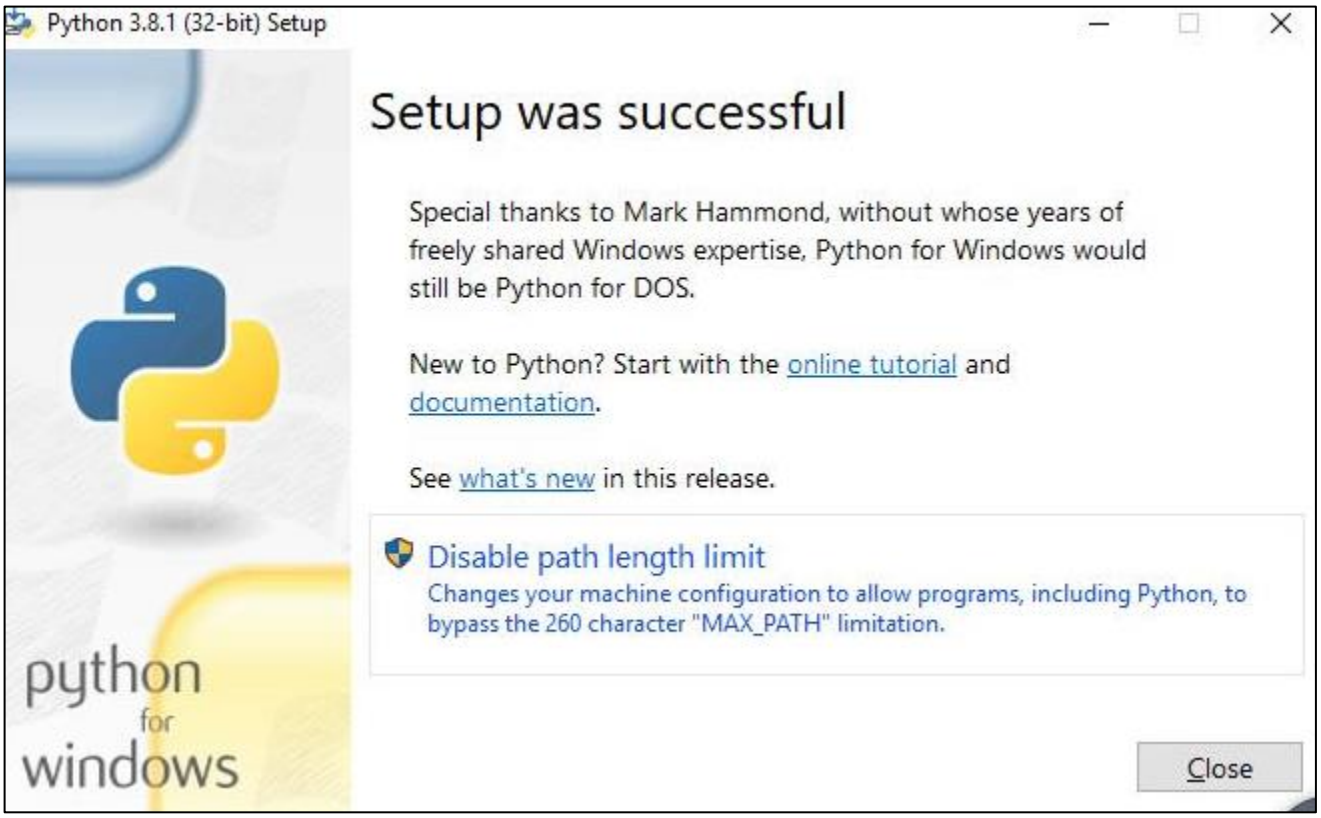
Here you can select the installation path of Python. We install it at D drive. If you are a novice, you can select the default path.



Wait for it to finish installing.

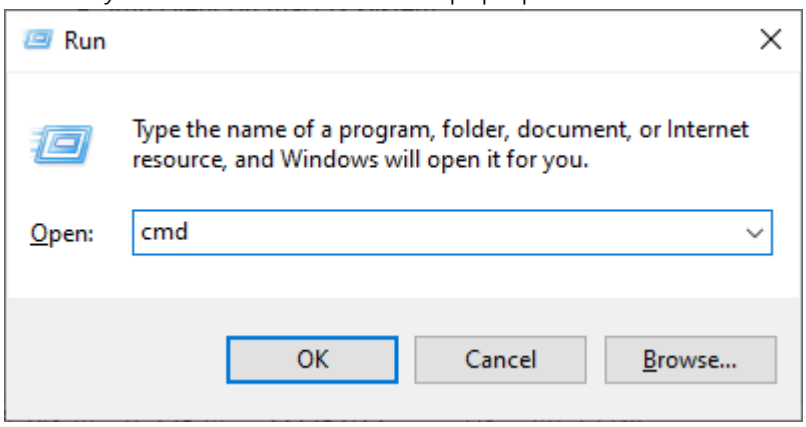


Now the installation is finished.



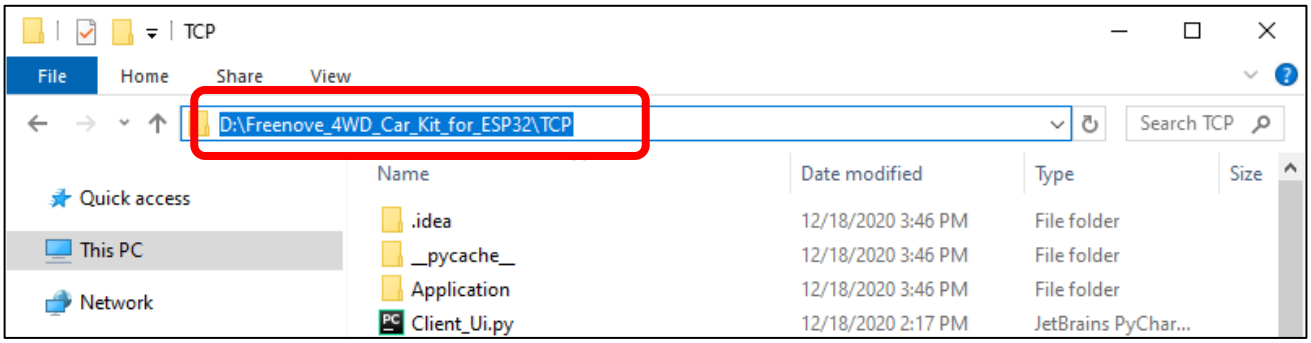
Install PyQt5、opencv、numpy and other libraries.

Press "Ctrl" and "R" on the keyboard at the same time will pop up a window.



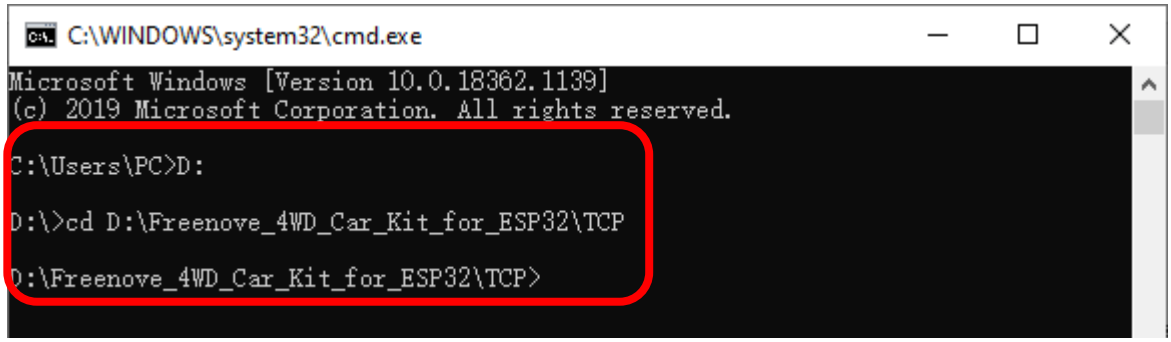
Enter **cmd** in the pop-up window and click "OK".

Use the command to enter the storage location of Freenove_4WD_Car_Kit_for_ESP32. Here we assume it is saved in D drive.

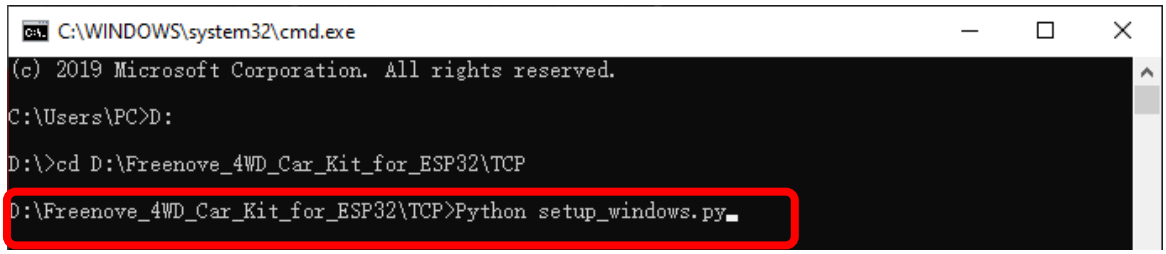


The address depends on where you saved Freenove_4WD_Car_Kit_for_ESP32. You need to modify it based on your own storage location.

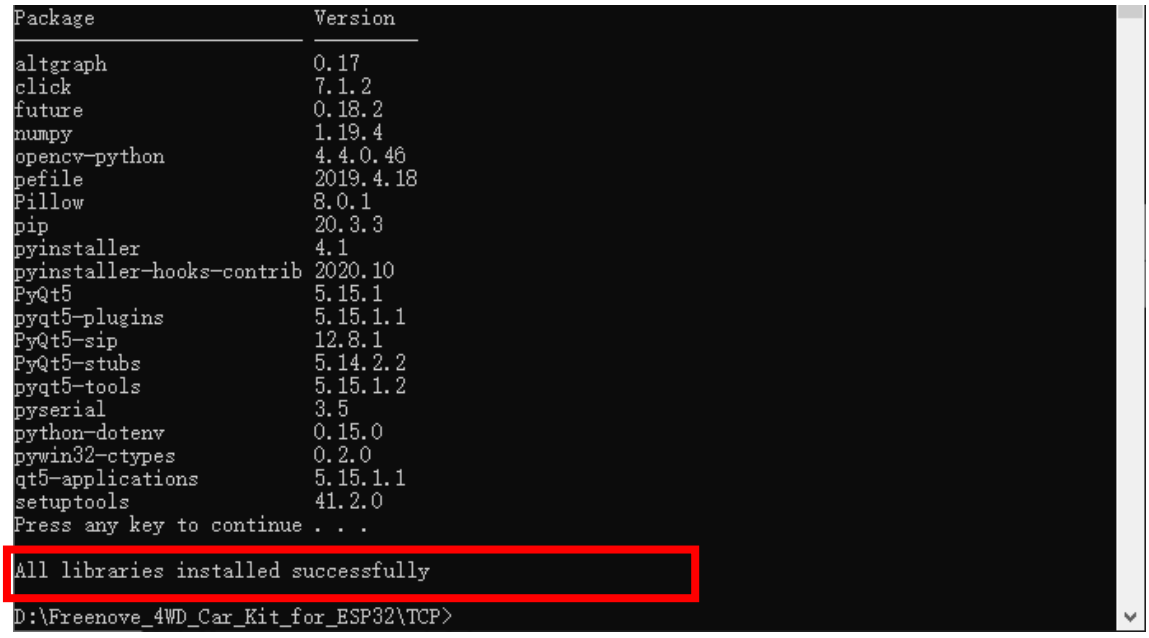
1. Input "D:", press enter to enter D drive, and then input "cd D:\Freenove_4WD_Car_Kit_for_ESP32\TCP", press enter to enter the folder.



2. Input "Python setup_windows.py" and press enter. If it fails, you can input "Python3 setup_windows.py" to try again.



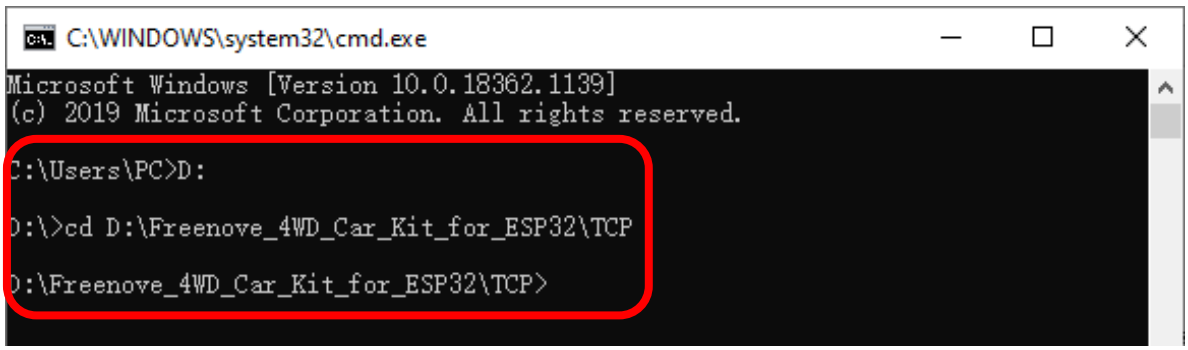
3. Wait for it to finish installing and press any key according to the prompt. When it displays "All libraries installed successfully", it indicates the environment is installed successfully.



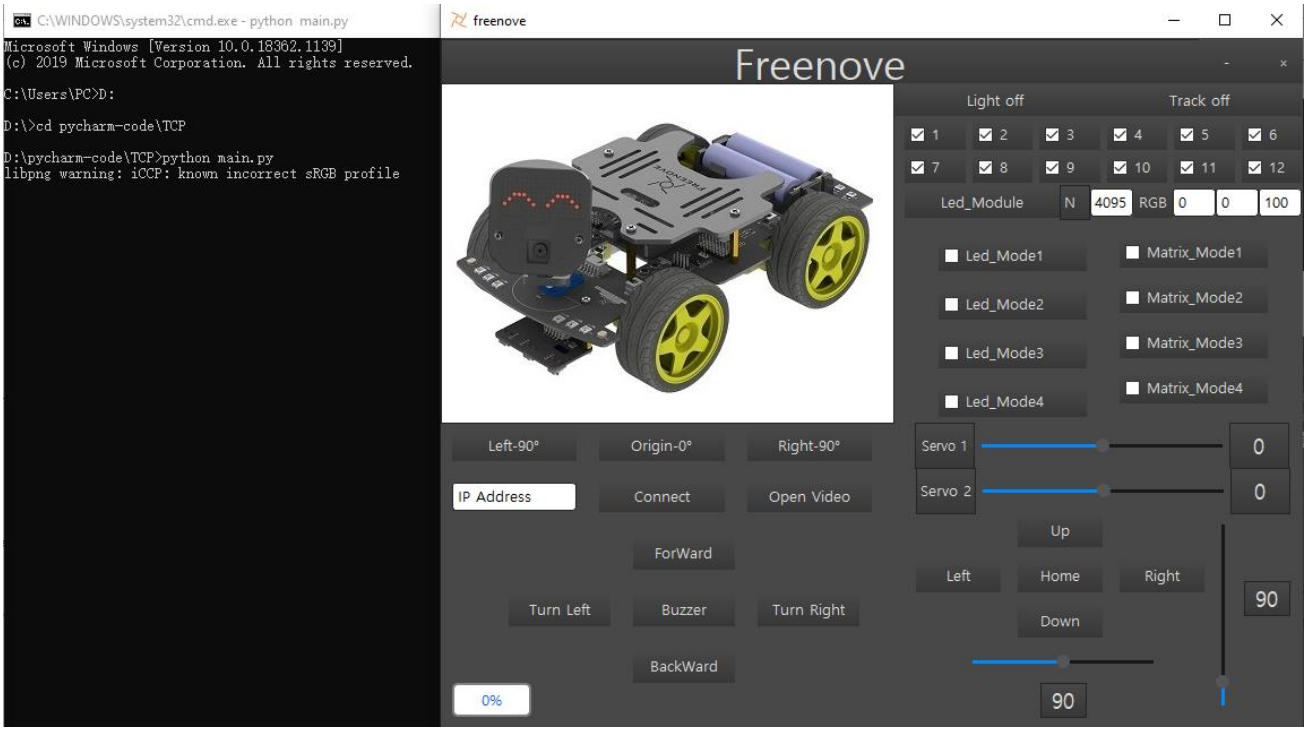
If not all libraries are installed successfully, it will prompt "Some libraries have not been installed. Please run Python3 setup_windows.py again', and then you need to enter and execute the command again. Most installation failures are caused by poor networks. You can Check your network before installation.

Run main.py

1. Input "D:", press enter to enter D drive and then input "cd D:\Freenove_4WD_Car_Kit_for_ESP32\TCP", press enter to enter the folder.



2. Input "Python main.py" and press enter. If it fails, you can input 入"Python3 main.py" to try again.



After the program is run, you can control the car through this window.

If you want to modify the code, you can modify the files in TCP folder.

Control the car with keyboard

The car can be controlled by clicking the client. And it can also be controlled by pressing keys on your keyboard.

The following is the corresponding operation of the buttons and keys.

Button on Client	Key	Action
ForWard	W	Move
BackWard	S	Back off
Turn Left	A	Turn left
Turn Right	D	Turn right
Left	left arrow	Turn camera left
Right	right arrow	Turn camera right
Up	up arrow	Turn camera up
Down	down arrow	Turn camera down
Home	Home	Turn camera back Home
Connect/ Disconnect	C	On/off Connection
Open Video/ Close Video	V	On/off Video
Buzzer	Space	On/off Buzzer
Led_Mode 1,2,3,4	L	Switch RGB Led Mode
Matrix_Mode 1,2,3,4	K	Switch Matrix Mode
Matrix_Static Mode	J	Switch Matrix Static Mode
Light Car Mode	R	On/off Light Car Mode
Track Car Mode	T	On/off Track Car Mode

The function of SliderBar is below:

SliderBar	Function
Servo 1,2	SliderBar Servo 1, 2 are used to slightly adjust the angle. If the servo is not fully centered during installation, you can slightly tune it via the SliderBar.

Other control information:

Control	Function
IP address Edit box	Enter IP address of ESP32.
Power box	Show power level.
N,R,G,B Edit box	Control the color of LED selected.
Button “Light On/Off”	Turn on the light seeking function of the car. If WiFi is disconnected, the car will stop moving.
Button “Track On/Off”	Turn on the tracking function of the car. If WiFi is disconnected, the car will stop moving.
Left-90°	Rotate the received video 90° to the left to display.
Origin-0°	Restores the received video to its original display.
Right-90°	Rotate the received video 90° to the right to display.

Run client on macOS system

Here we take MacOS 10.13as an example. To run client on MacOS, you need to install some software and libraries. MacOS 10.13 comes with python2 but not python3. However, the projects in this program need to be run with python3, so you need to install it first.

Install python3

Download installation package, link: https://www.python.org/downloads/

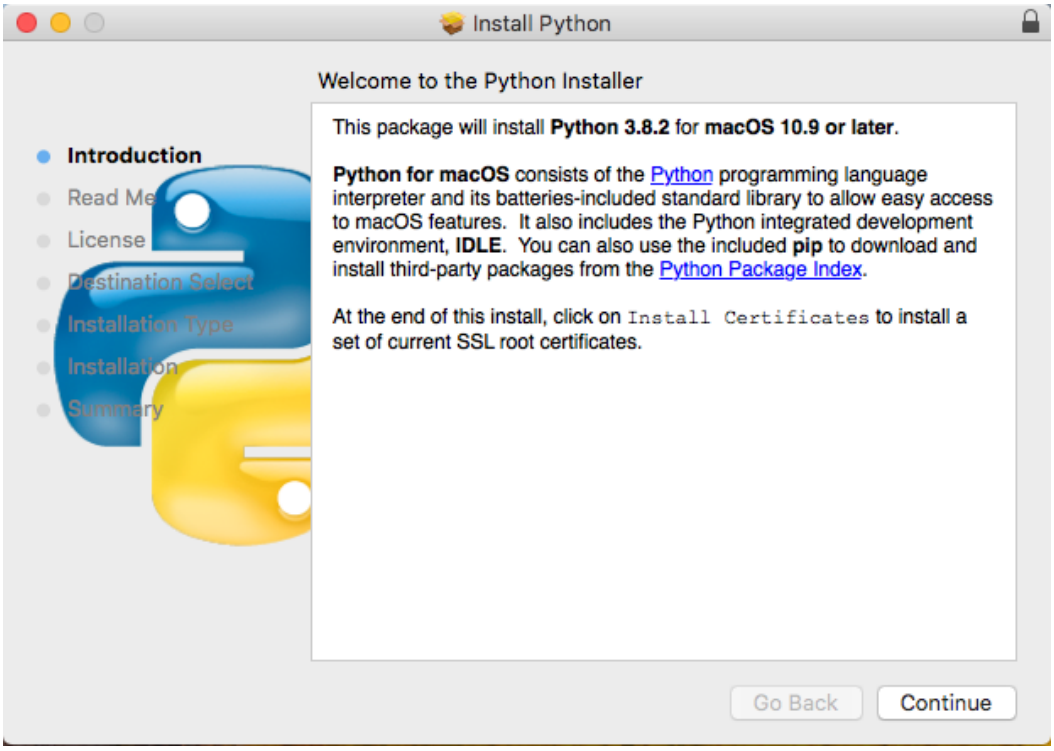
Click Python 3.8.2.

Release version	Release date	
Python 3.8.2	Feb. 24, 2020	Download
Python 3.8.1	Dec. 18, 2019	Download
Python 3.7.6	Dec. 18, 2019	Download
Python 3.6.10	Dec. 18, 2019	Download
Python 3.5.9	Nov. 2, 2019	Download
Python 3.5.8	Oct. 29, 2019	Download

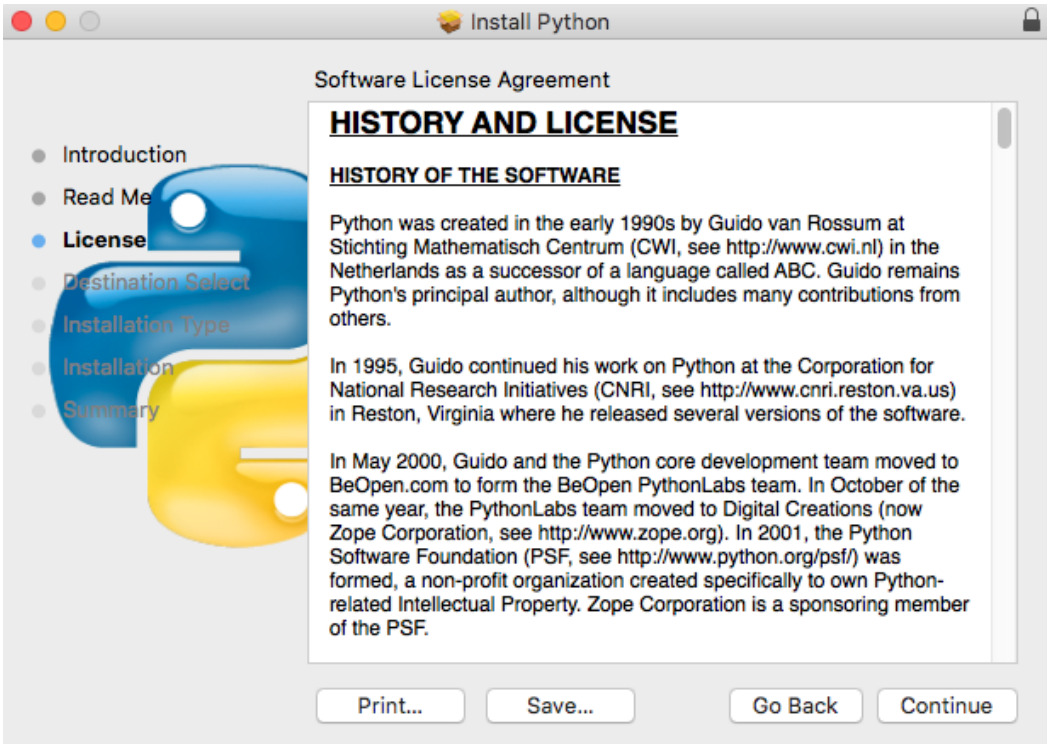
At the bottom of the page, click macOS 64-bit installer and download installation package.

Version	Operating System	Description
Gzipped source tarball	Source release	
XZ compressed source tarball	Source release	
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later
Windows help file	Windows	
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64
Windows x86 embeddable zip file	Windows	
Windows x86 executable installer	Windows	
Windows x86 web-based installer	Windows	

Click Continue.

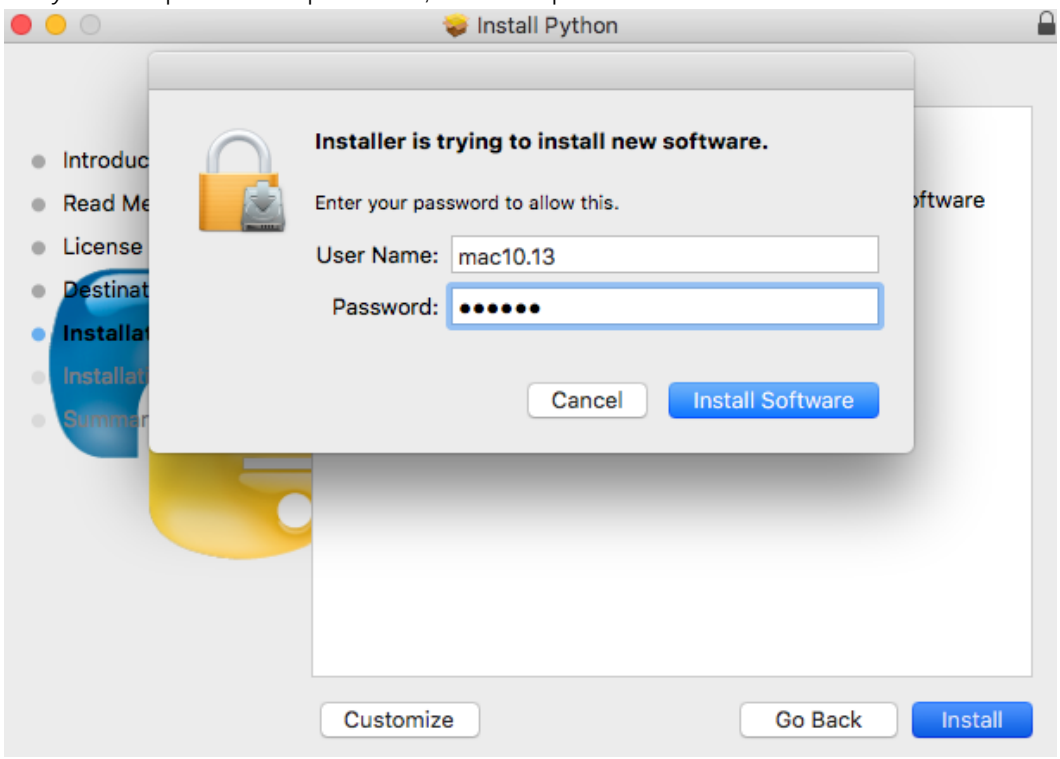


Click Continue

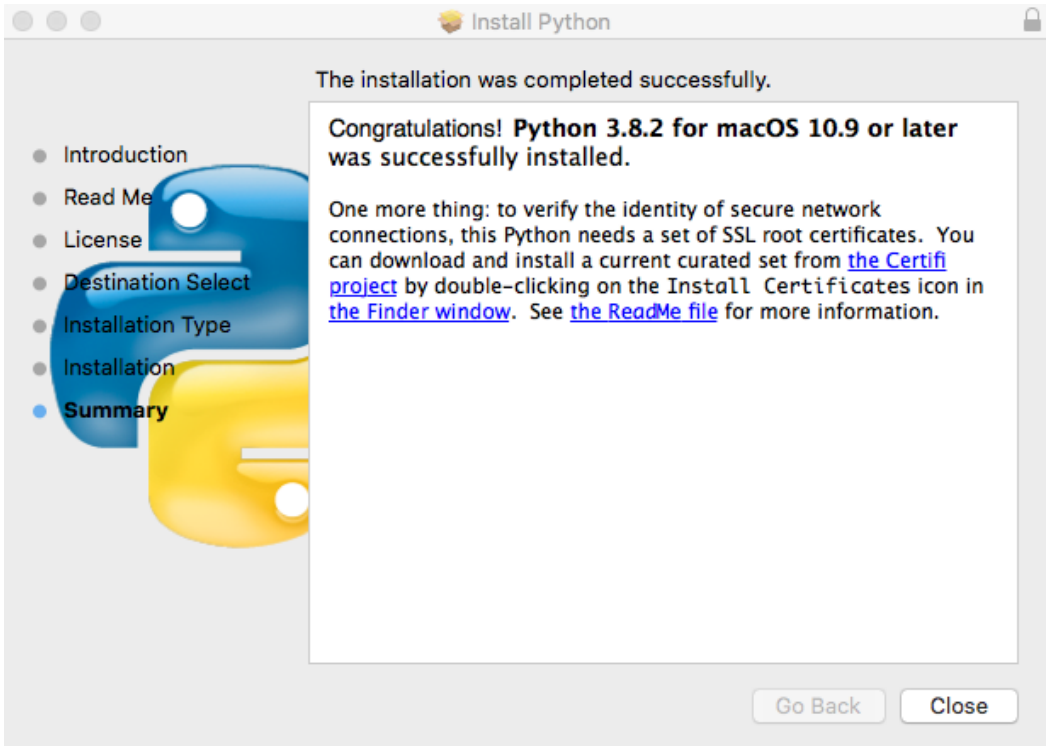


Click Agree.

Click Install. If your computer has a password, enter the password and Install Software.



Now the installation succeeds.



You can find it in Applications.

Install PyQt5、opencv、numpy and other libraries

If you have not yet download the code of ESP32 car to your macOS, you can download it from the following link: https://github.com/Freenove/Freenove_4WD_Car_Kit_for_ESP32.git

After downloading, you can find it in **Downloads** of Macintosh HD.

Open **Terminal**.

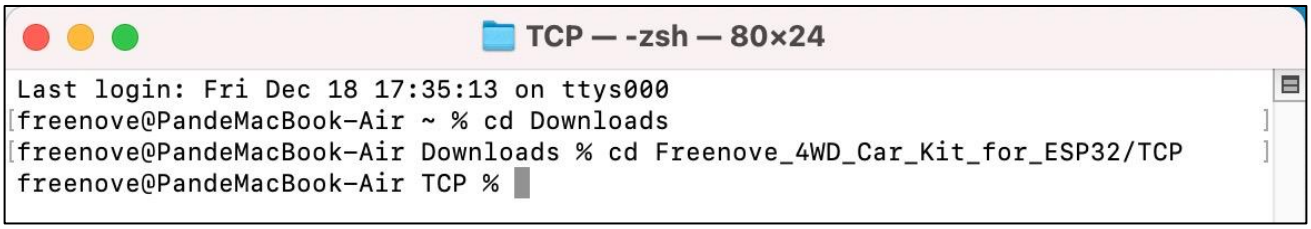
Enter the following commands in Terminal

1. Enter "**Downloads**"(where the code locates). If your code locates in a different path, you need to enter the location of your code.

```
cd Downloads
```

2. Enter the directory where setup_mac.py locates

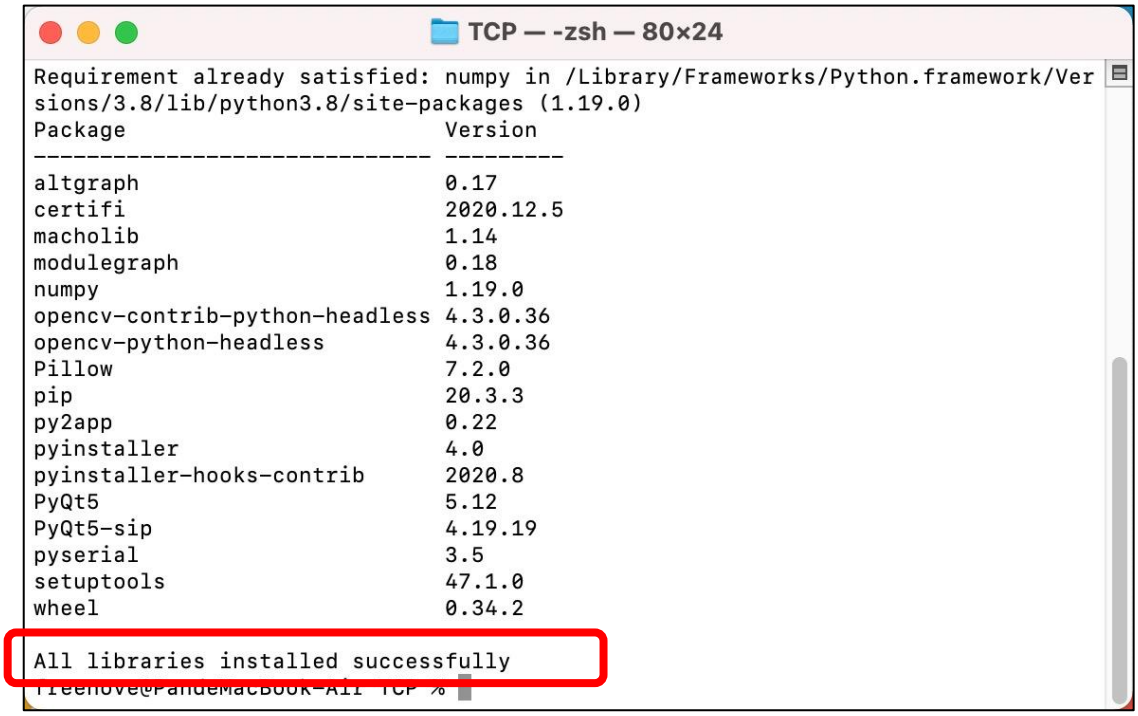
```
cd Freenove_4WD_Car_Kit_for_ESP32/TCP
```



3. Run setup_macos.py:

```
python3 setup_macos.py
```

The installation takes some time. Please wait with patience. When installs successfully, it will print "All libraries installed successfully":



Note: If not all libraries are installed successfully, it will prompt “Some libraries have not been installed”. Please input and run Python3 setup_macos.py again. Most installation failures are caused by poor networks. You can Check your network before installation

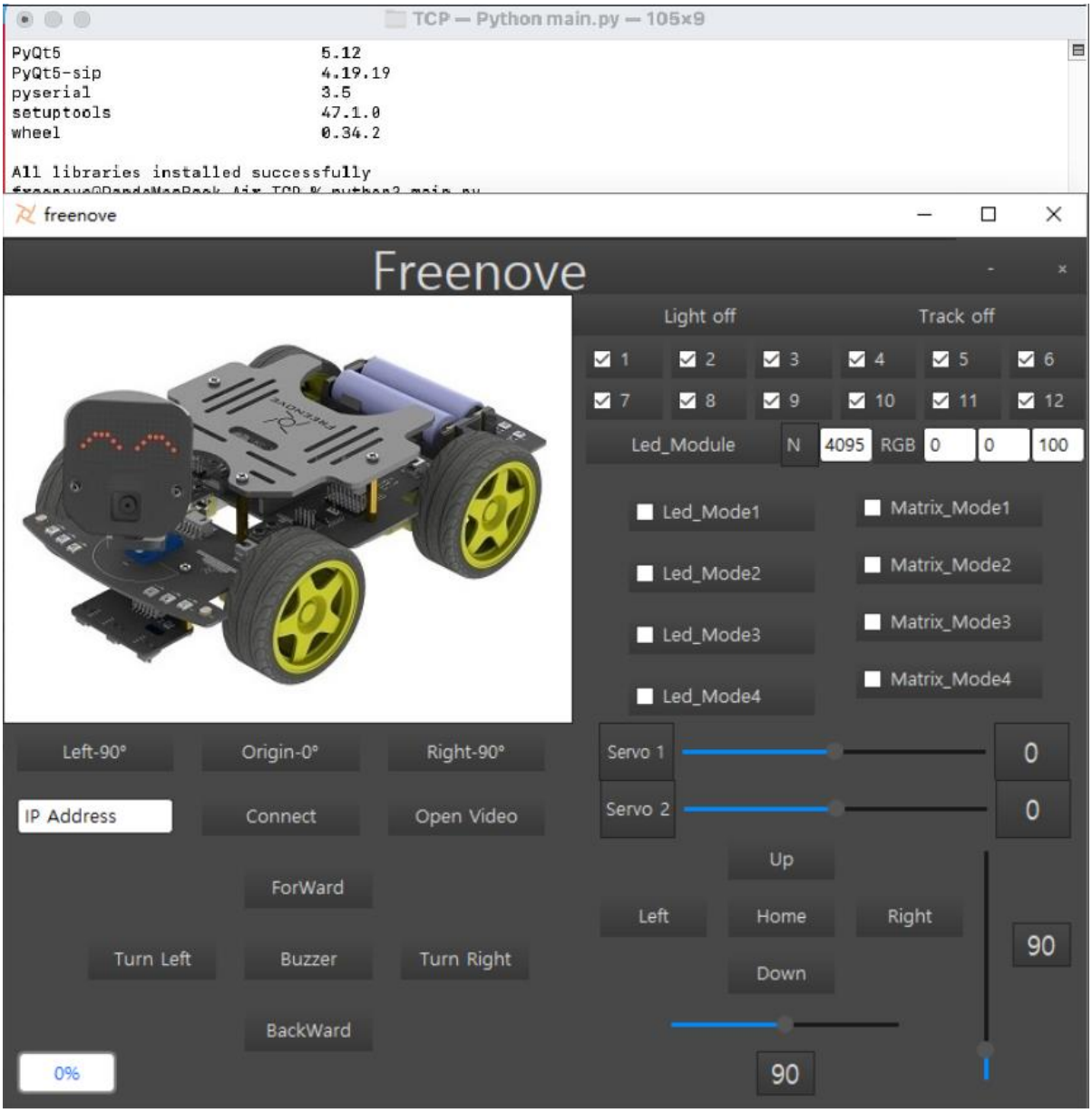
Run main.py

Following the previous steps, after the installation is complete, you are now in the directory where setup_mac.py is located. (Freenove_4WD_Car_Kit_for_ESP32/TCP)

```
All libraries installed successfully
freenove@PandeMacBook-Air TCP %
```

Enter the following command to run main.py.

```
python3 main.py
```



The control way of macOS System client is same with Windows (Control).

What's next?

Thank you again for choosing Freenove products.

THANK YOU for participating in this learning experience!

We have reached the end of this Tutorial. If you find errors, omissions or you have suggestions and/or questions about the Tutorial or component contents of this Kit, please feel free to contact us: support@freenove.com

We will make every effort to make changes and correct errors as soon as feasibly possible and publish a revised version.

If you want to learn more about Arduino, ESP32, Raspberry Pi, Smart Cars, Robotics and other interesting products in science and technology, please continue to visit our website. We will continue to launch fun, cost effective, innovative and exciting products.
http://www.freenove.com/

Thank you again for choosing Freenove products.