Table of Contents

1. Introduction to eArm 3
1.1 Product List 3
1.2 Parameters of the eArm: 6
1.3 Introduction to ESP32 Control Board: 7
2. Assemble the eArm 8
Before You Begin: Important Notes 8
Step 1 Assemble the Robot Base 9
Step 2 Assemble Servo A on Robot Base 10
Step 3 Assemble the Turntable 11
Step 4 Assemble the Lower arm 12
Step 5 Fix the lower arm 13
Step 6 Assemble the Middle Arm 14
Step 7 Assemble the Upper Arm 15
Step 8 Assemble the Servo B on lower arm 16
Step 9 Assemble the Servo C on Middle Arm 17
Step 10 Assemble the Servo D on Upper Arm 18
Step 11 Assemble the Left Clip Plate of The Clamp 19
Step 12 Assemble the Right Clip Plate of The Clamp 19
Step 13 Install the Left Clip Plate of The Clamp 20
Step 14 Install the Right Clip Plate of the Clamp 21
Step 15 Assemble the Clamp and Upper Arm 22
Step 16 Assemble the linkage between Servo B and Servo C 23
Step 17 Assemble the Mounting Structure for Servo A 24
Step 18 Assemble the Mounting Structure for Servo D 25
Step 19 Assemble the Joystick 26
Step 20 Install the Esp32 Board 27
Step 21 Connect the Joystick 28
Step 22 Connect the Servos to the ESP32 Board 29
Step 23 Initialize the Servo Angle (important!) 30
Step 24 Fix the Robot Arm to Servo A of the Base 31
Step 25 Assemble the Middle Arm 32
Step 26 Fix the Clip Plates to Servo D 33
Step 27 Assembly completed 34
3. Using the Robot Arm 35
3.1 Robotic Arm Safety and Operating Guidelines 35
3.1 Degrees of Freedom 36
3.2 Control the eArm with a Joystick 37
3.3 Control the eArm with Web App 38
4. **Programming Learning** 40
4.1 Before You Begin: Important Notes 40
4.3 Install Thonny IDE 40

4.4 Install CH340 USB driver 43
4.5 Flashing MicroPython Firmware using Thonny IDE 46
4.6 Thonny IDE Overview 50
4.7 Running Your First Code - Run current code in Thonny 52
4.8 Uploading Code to the ESP32 Board - Run code in ESP32 54
Method 1: Save/Upload Directly as main.py 54
Method 2: main.py calls other files 56
4.9 Code Examples 61
4.9.1 Play Songs 61
4.9.2 To Drive a Servo 62
4.9.3 Read the Value of the Joystick 64
4.9.4 Joystick Control eArm 65
4.9.5 Read the Value of the Web APP 66
4.9.6 Web App-Controlled eArm Robot 69
5. **Factory Reset Function** 72
5.1 Firmware 72
5.2 Burning Tool 73
5.3 How to Burn Firmware 74
6. **Common Issues & Solutions (Windows)** 75
6.1 How to Install Thonny IDE on MacOS? 75
6.2 Robotic arm doesn't work 78
6.3 Port Connection & Device Recognition Errors 78
6.4 MicroPython Firmware & Interpreter Configuration Errors 80
6.5 Runtime Errors (Code Execution on ESP32) 82
6.6 File System & Device Display Abnormalities 85
6.7 Serial Port Occupation & Connection Drop Issues 86
7. Contact Us 87

1

Introduction to eArm

The eArm robot is an ESP32-powered robotic arm, it offers a hands-on way to explore robotics, coding, and electronics in an engaging platform.

Features:

① **Quick Assembly & Ready-to-Use**

▸ The robotic arm is designed for effortless assembly—you can build it and start experimenting within 1–2 hours.

② **Pre-Programmed for Instant Operation**

▸ The control board comes pre-flashed with firmware, eliminating the need for initial code uploads. Simply power it on and begin using it right away.

③ **Integrated Power Management**

▸ Features an onboard battery holder with a power indicator. The robot can be charged directly via USB, ensuring uninterrupted operation.

④ **Precision Control with Joystick**

▸ The included ergonomic joystick delivers stable control, offering intuitive and responsive manipulation of the robotic arm.

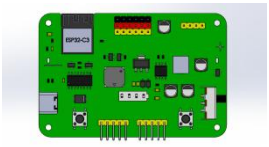
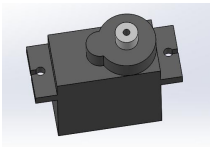
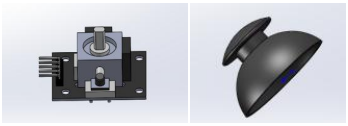
⑤ **Web-Based Control**

▸ Supports cross-platform control via a dedicated Web APP, accessible from any device with a browser—no additional installations required.

⑥ **Expandable Learning with Arduino**

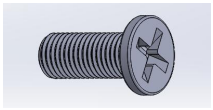
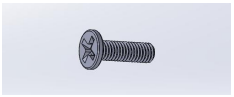
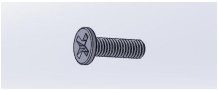
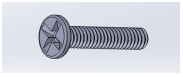
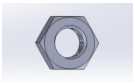
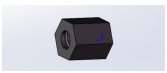
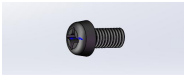
▸ Includes step-by-step Arduino programming tutorials, helping users advance from basic operations to custom robotics development.

1.1 Product List

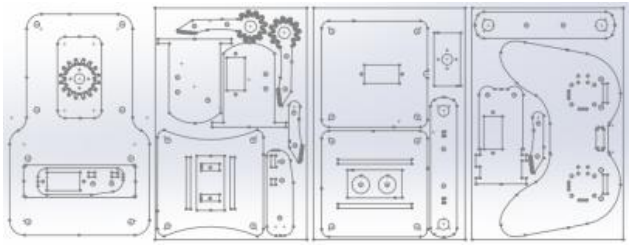
#Part 1 : Electronics Components		
Control Board 1PCS	Servo 4 PCS	5P Dupont Wire 2PCS
		
Joystick 2PCS	USB Cable 1PCS	
		

#Part 2 : Fasteners / Accessories

The following accessories listed in sequential order are all packaged in a small bag with a label and serial number attached. You can quickly locate them by referring to the serial number and name on the bag.

1 M4×10mm Screw 21Pcs	2 M3×14mm Screw 3Pcs	3 M3×10mm Screw 19Pcs
		
4 M2×10mm Screw 9Pcs	5 M1.4×5mm Screw 8Pcs	6 M4 Nut 5Pcs
		
7 M3 Lock Nut 9Pcs	8 M2 Nut 10Pcs	9 M3×7mm Standoff 5Pcs
		
10 M3×28mm Standoff 4Pcs	11 M3×40mm Standoff 2Pcs	12 M4×20mm Standoff 8Pcs
		
13 M3×6mm Screw 9Pcs	14 Flanged Bearing 4Pcs	Turntable 1PCS
		

#Part 3 : Acrylic Sheet



#Part 4 : Tools

		
M3 screwdriver 1PCS	M1.5 screwdriver 1PCS	wrench 1PCS

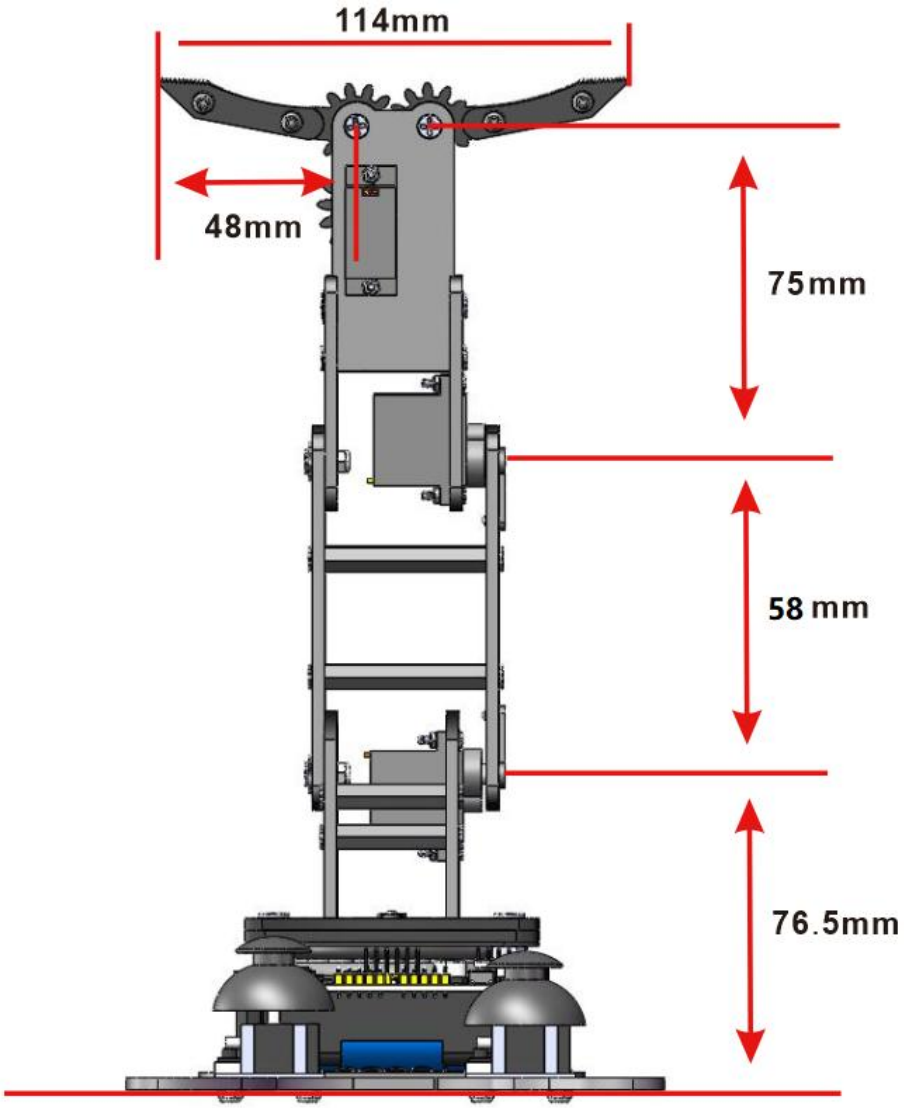
Note：You need to buy a 18650 lithium battery yourself!

Parameters of 18650 lithium battery

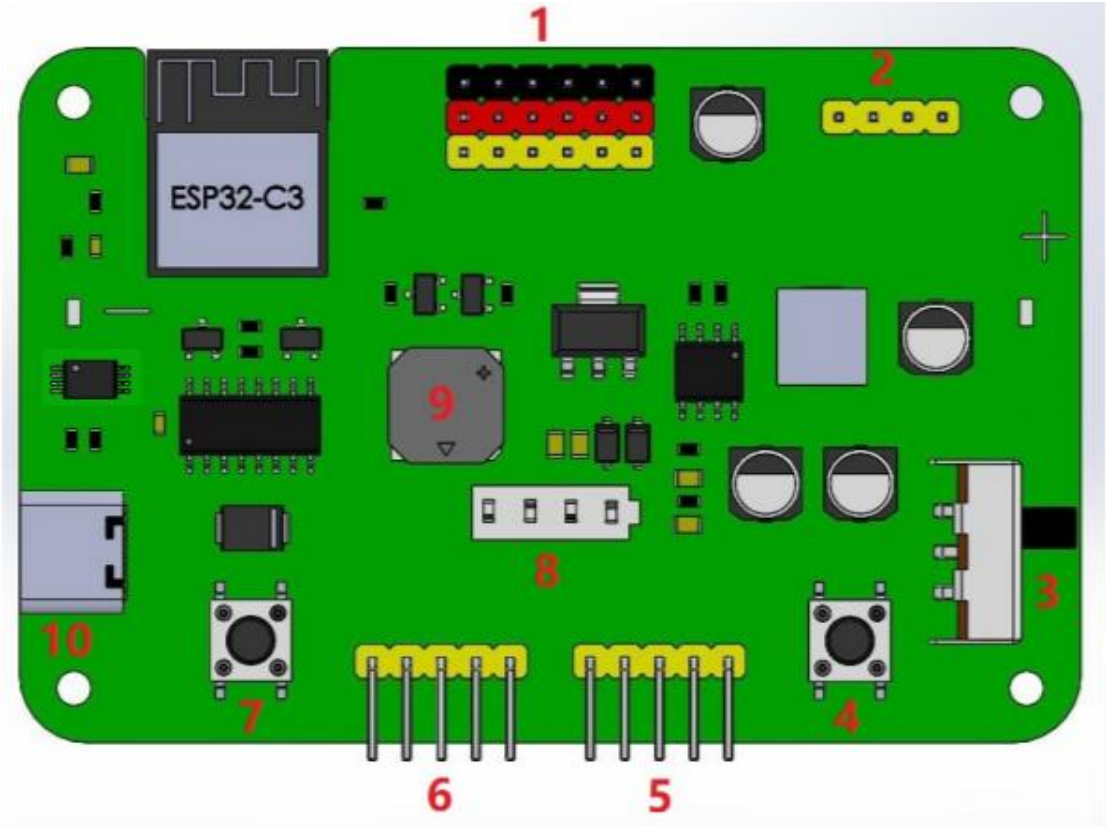
- Model: 18650 lithium battery
- Top type: Both button top and flat top are acceptable
- Capacity: >2000mAh
- Maximum charging voltage: 4.2V
- Nominal voltage: 3.7V
- End-off voltage: 2.75V
- Minimum charging current: >2A
- Minimum discharge current: >4A

1.2 Parameters of the eArm:

Control Board	eArm-ESP32-C3-MINI-1
Programming language	Arduino\MicroPython
Charge/discharge	5V/2A
Servo type	MG90S 180°
Remote control method	Joystick or Web APP
Assembly time required	1-2 hours
Product size	15X9.5X26MM
Packaging size	18.5X12.5X4.3MM
Applicable age	experienced players of any age or beginner aged 12+
Battery required	a 18650 battery(Not included)



1.3 Introduction to ESP32 Control Board:



■ 1: IO port, 5V/2A driving capability.

■ 2: IO port and serial port expansion port.

■ 3: Power switch.

■ 4: Battery level display button(When the power switch is off, pressing it briefly can display the battery level).

■ 5: IO port, 3.3V/500mA driving capability.

■ 6: IO port, 3.3V/500mA driving capability.

■ 7: Reset button.

■ 8: Battery level indicator LED.

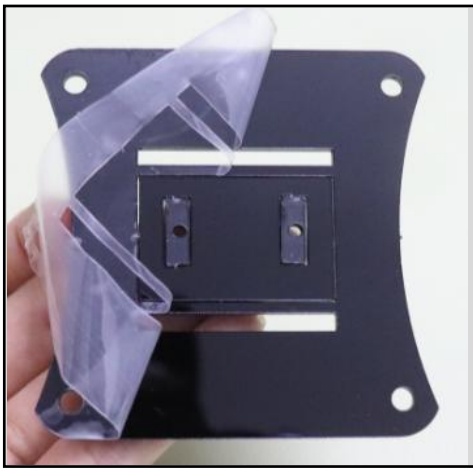
■ 9: Buzzer.

■ 10: USB Type-c Port for charging or uploading code

2

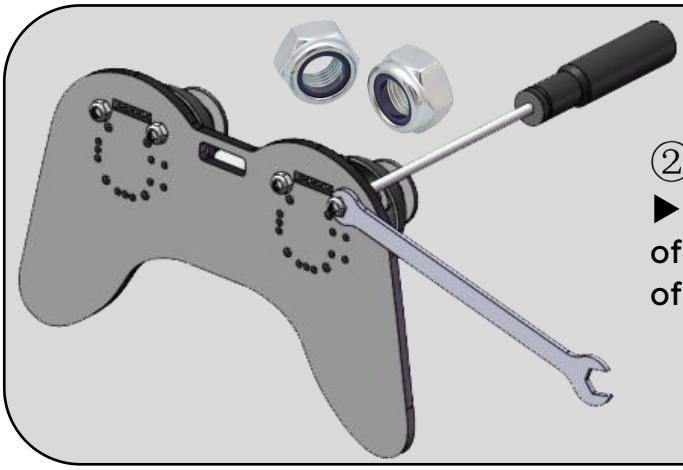
Assemble the eArm

Before You Begin: Important Notes



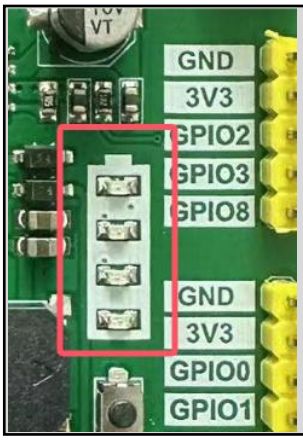
①

▶The surface of the acrylic board is covered with a protective film, please tear it off before installation!



②

▶Installation and removal of Lock nuts require the use of a hex wrench.



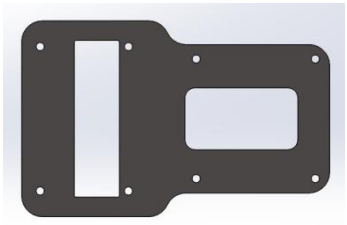
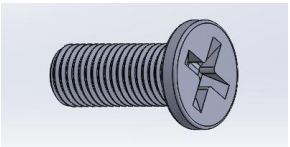
③

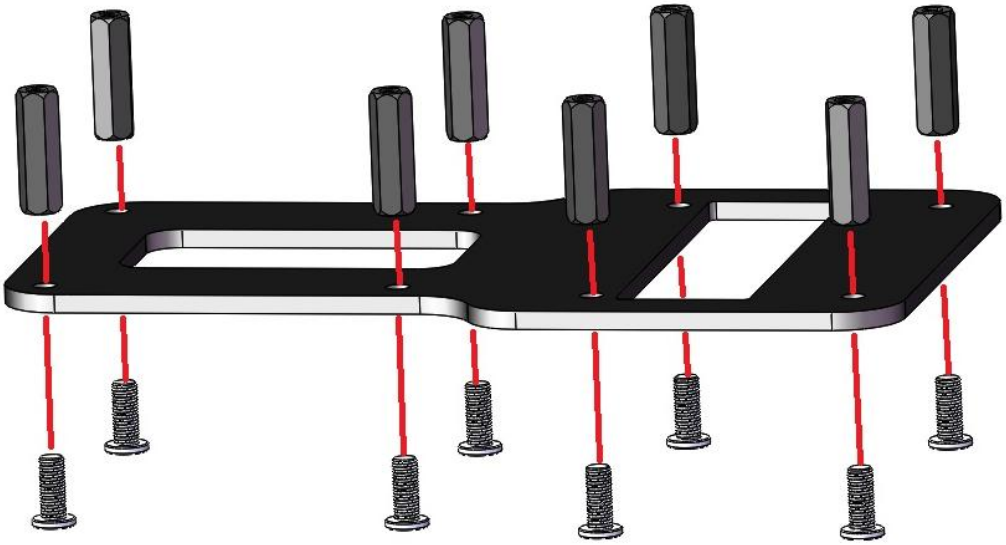
A minimum of three battery bars is required to ensure proper installation and operation of the robot. After installing an 18650 battery into the battery holder, check the power indicator. If it shows less than three bars, please charge the battery either:

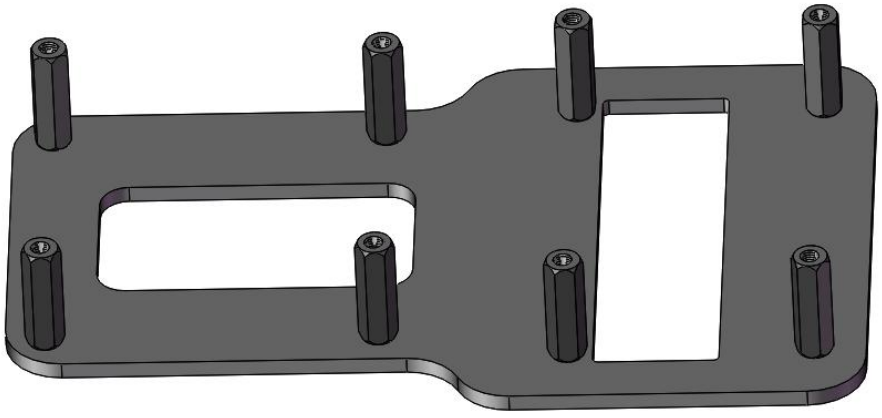
▶Using an 18650 dedicated charger

▶Via the board's charging port with a USB cable.

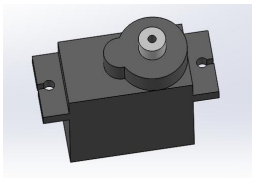
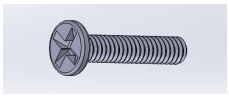
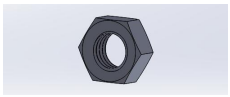
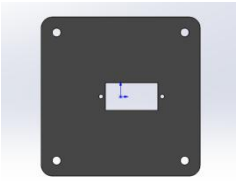
Step 1 Assemble the Robot Base

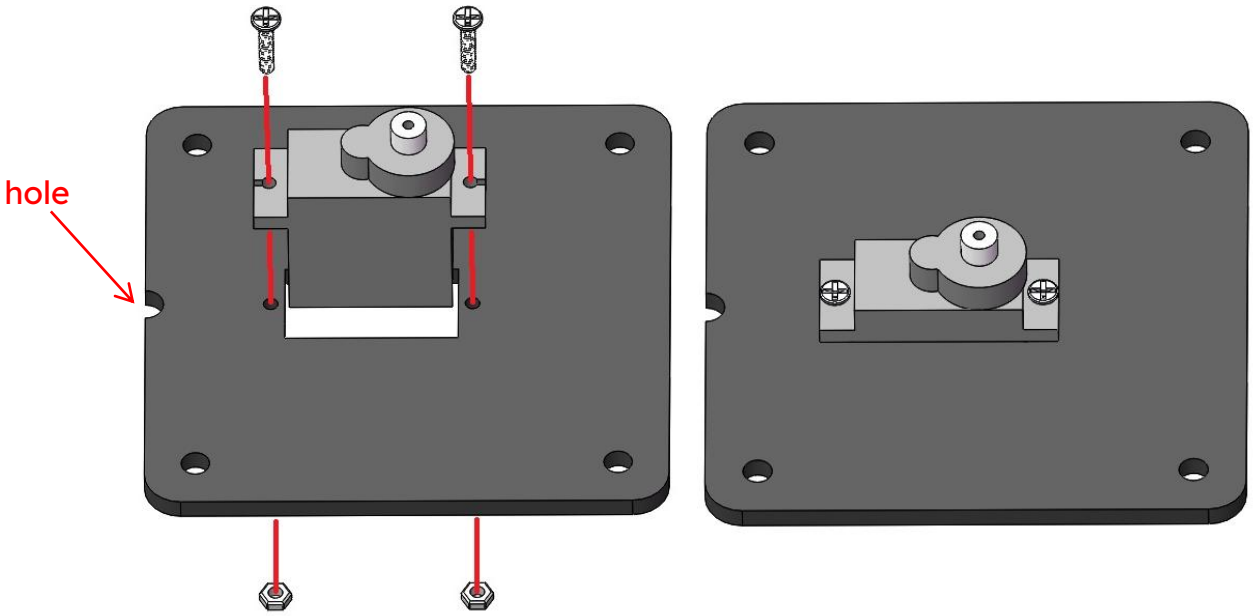
Acrylic Sheet 1PCS	**Bag No.⑫** M4X20MM Standoff 8PCS	**Bag No.①** M4X10MM Screw 8PCS
		



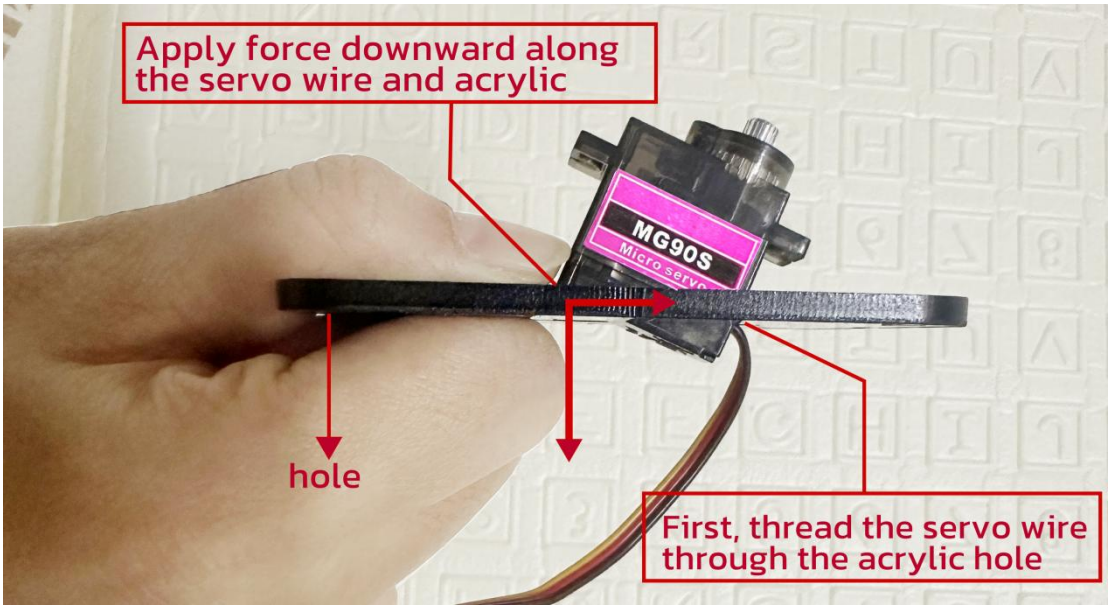


Step 2 Assemble Servo A on Robot Base

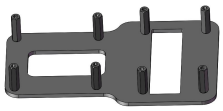
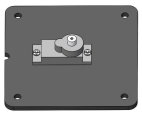
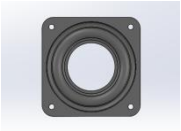
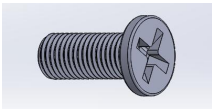
Servo 1 PCS	**Bag No.④** M2x10MM Screw 2PCS	**Bag No.⑧** M2 Nut 2PCS	Acrylic Sheet 1PCS
			

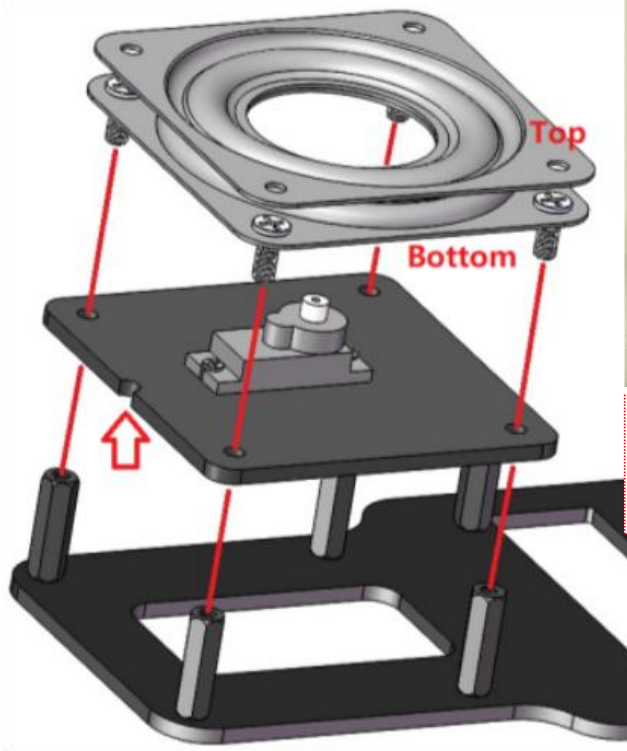


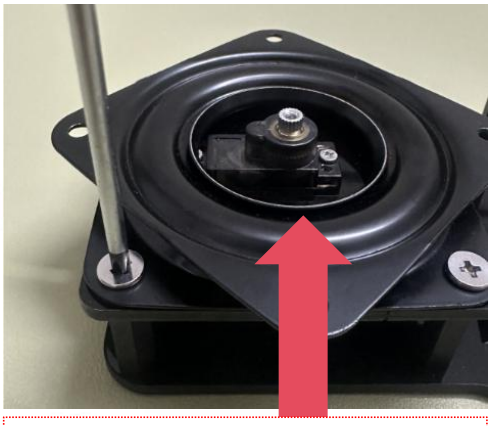
Note:



Step 3 Assemble the Turntable

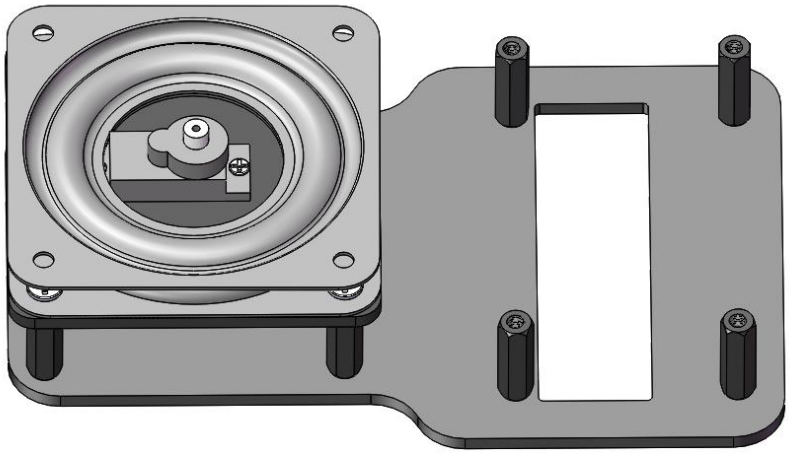
Step 1 Structure	Step 2 Structure	Turntable 1PCS	**Bag No.①** M4X10MM Screw 4PCS
			



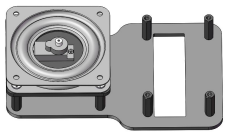


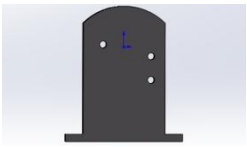
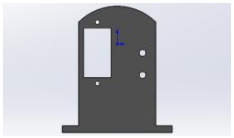
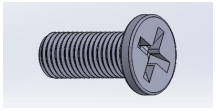
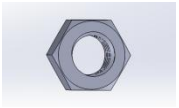
The innermost circular edge of the turntable should face upward

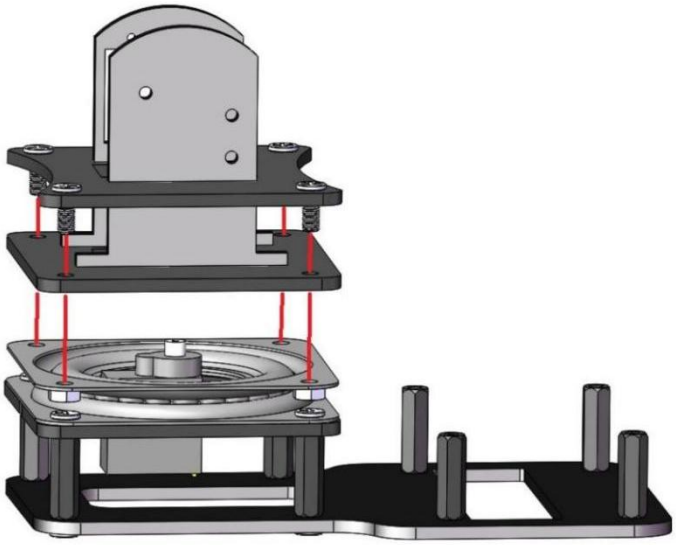
---

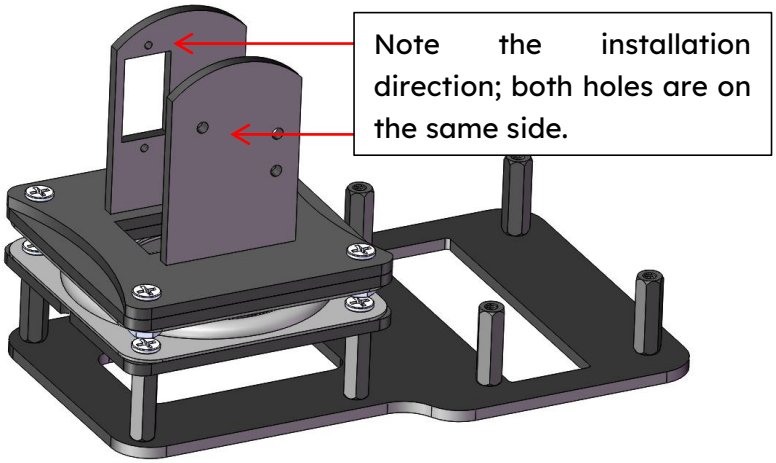


Step 4 Assemble the Lower arm

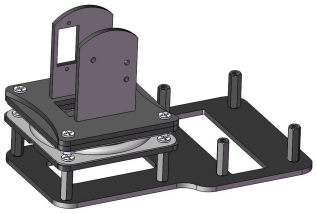
Step 3 Structure	Acrylic Sheet 1PCS	Acrylic Sheet 1PCS
		

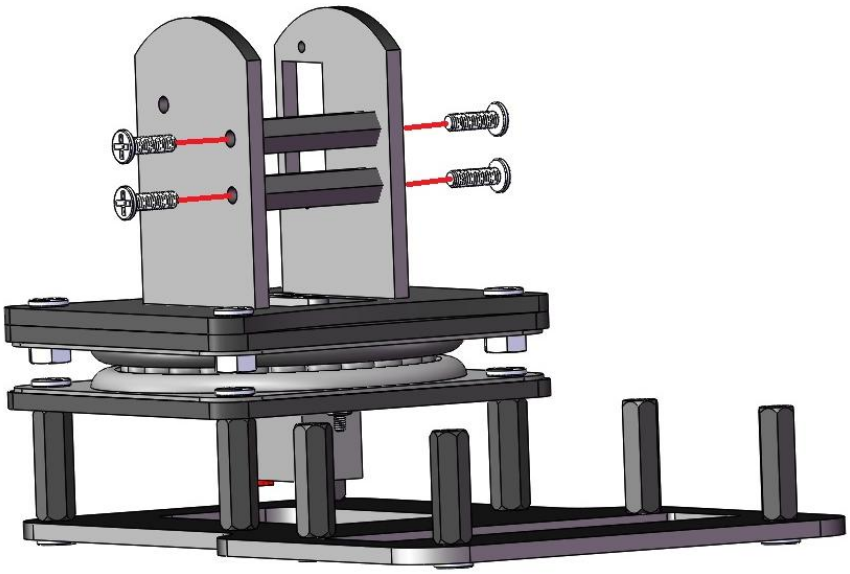
Acrylic Sheet 1PCS	Acrylic Sheet 1PCS	**Bag No.①** M4X10MM Screw 4PCS	**Bag No.⑥** M4 Nut 4PCS
			

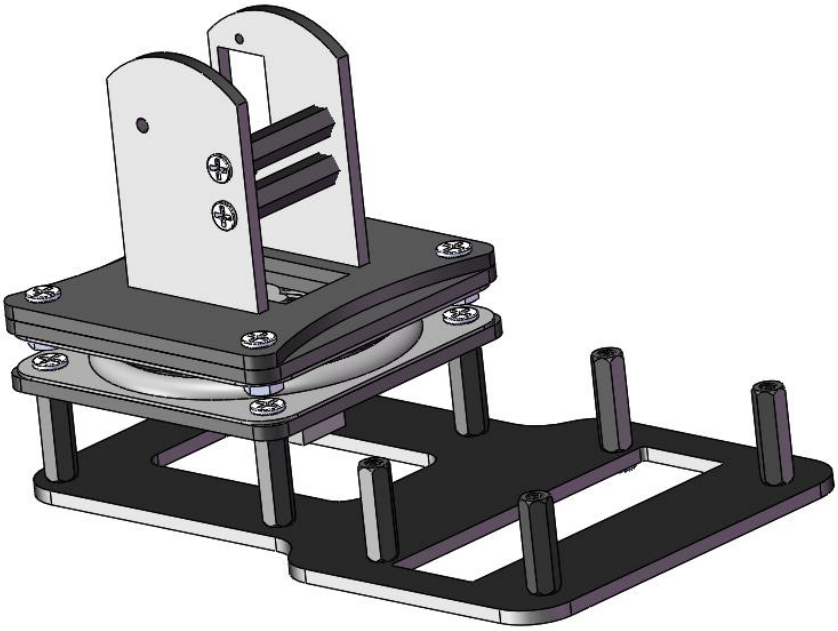




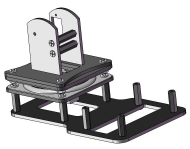
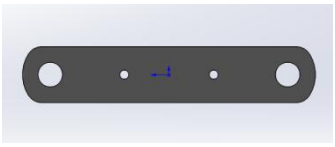
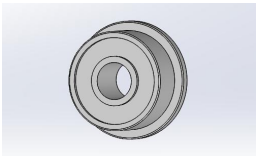
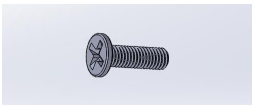
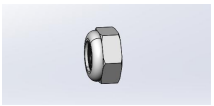
Step 5 Fix the lower arm

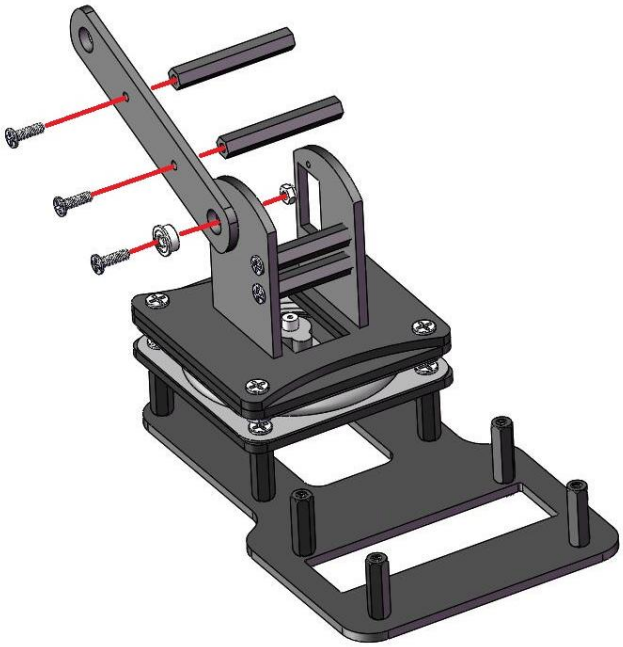
Step 4 Structure	**Bag No.⑩** M3x28MM Standoff 2PCS	**Bag No.③** M3X10MM Screw 4PCS
		



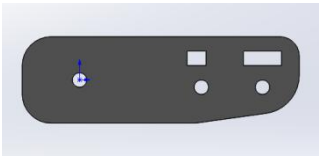
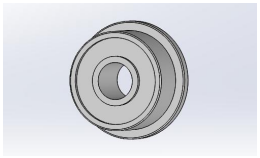
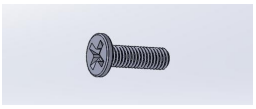


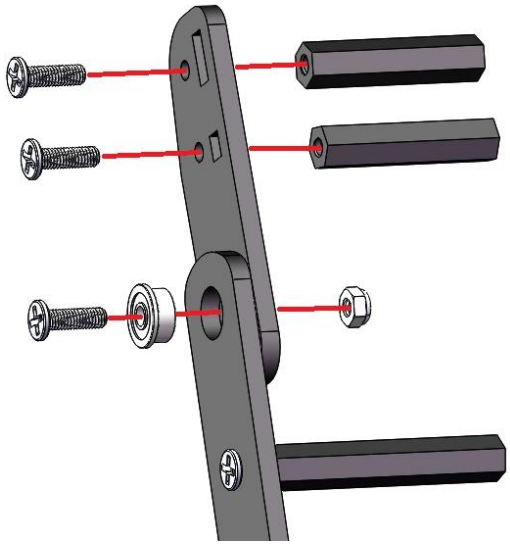
Step 6 Assemble the Middle Arm

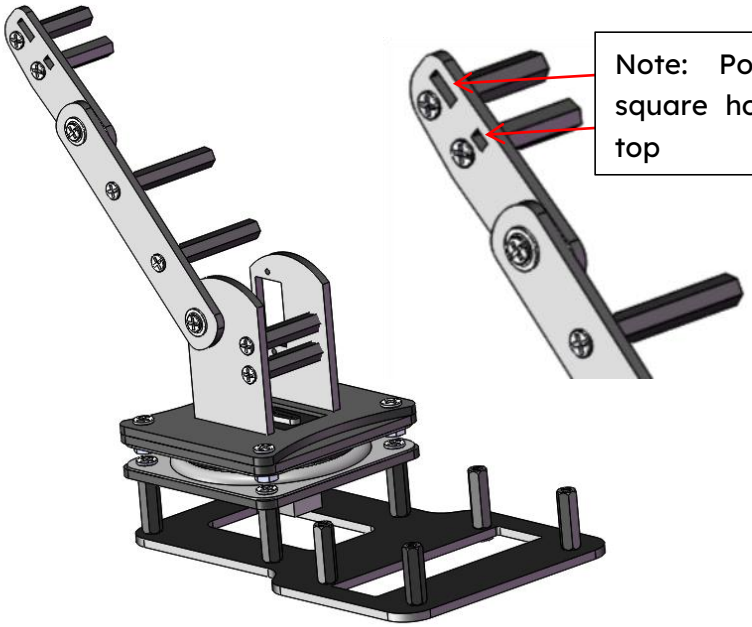
Step 5 Structure	Acrylic Sheet 1PCS	**Bag No.⑭** Flange Bearing 1PCS
		
Bag No.⑪ M3x40MM Standoff 2PCS	**Bag No.③** M3X10MM Screw 3PCS	**Bag No.⑦** M3 Lock Nut 1PCS
		



Step 7 Assemble the Upper Arm

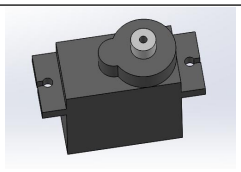
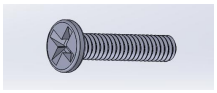
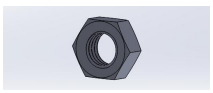
Step 6 Structure	Acrylic Sheet 1PCS	**Bag No.⑭** Flange Bearing 1PCS
		
Bag No.⑩ M3x28MM Standoff 2PCS	**Bag No.③** M3X10MM Screw 3PCS	**Bag No.⑦** M3 Lock Nut 1PCS
		

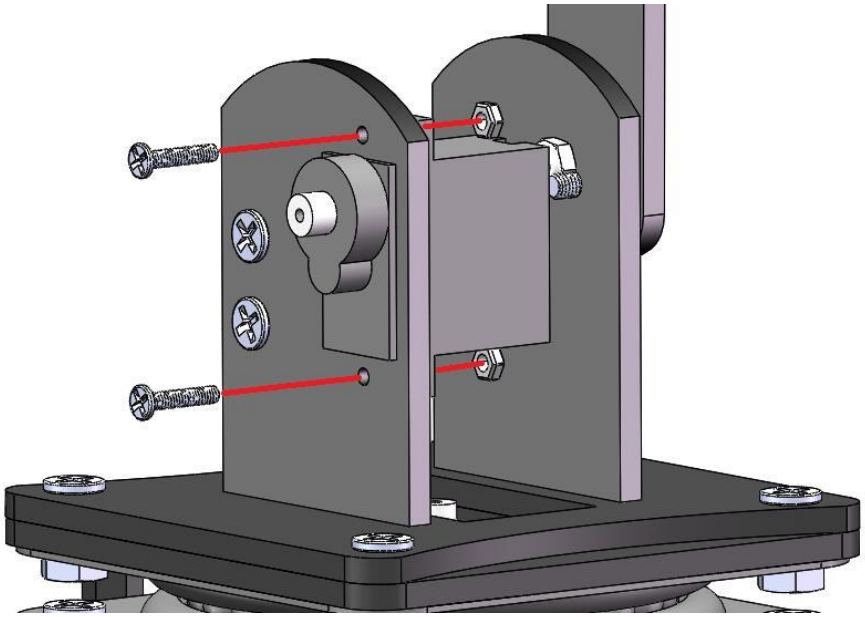


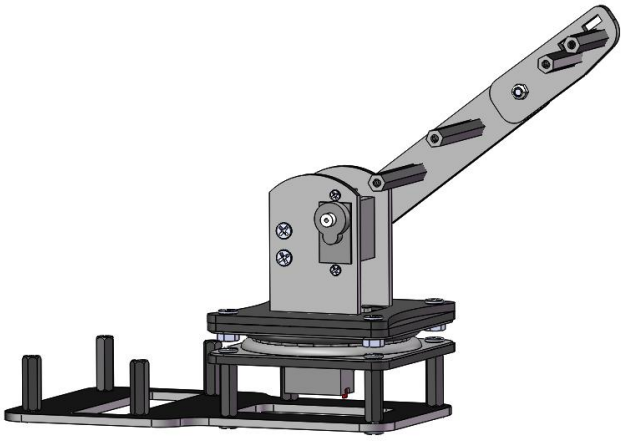


Note: Position the square holes at the top

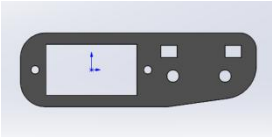
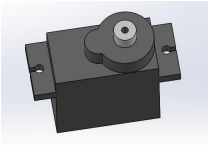
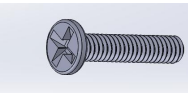
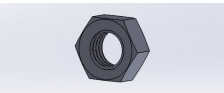
Step 8 Assemble the Servo B on lower arm

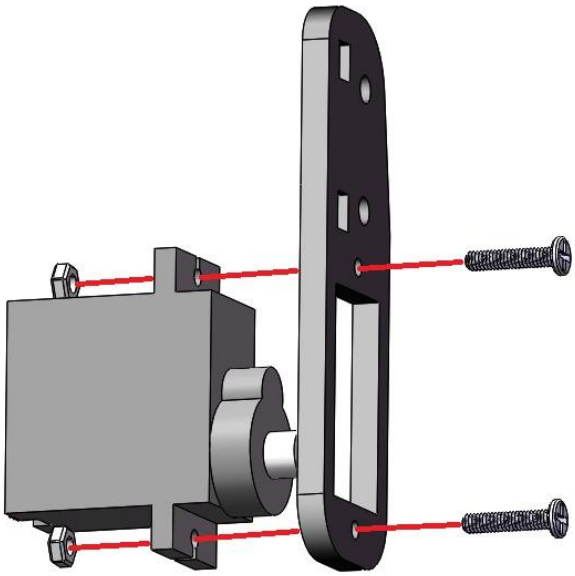
Step 7 Structure	Servo 1PCS	**Bag No.④** M2x10MM Screw 2PCS	**Bag No.⑧** M2 Nut 2PCS
			

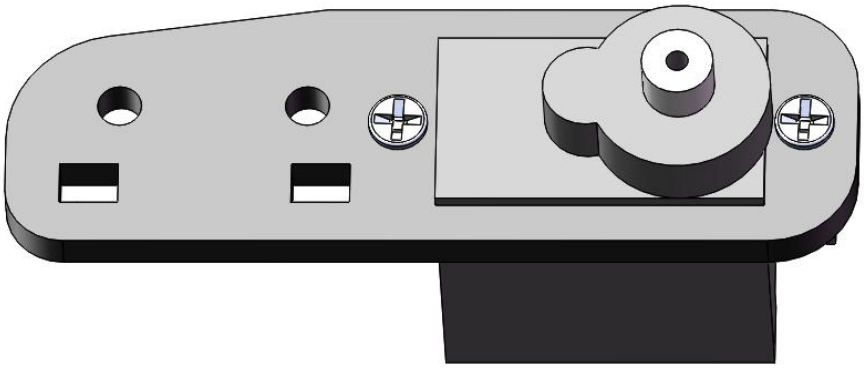




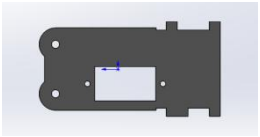
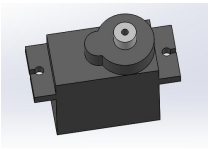
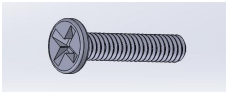
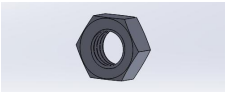
Step 9 Assemble the Servo C on Middle Arm

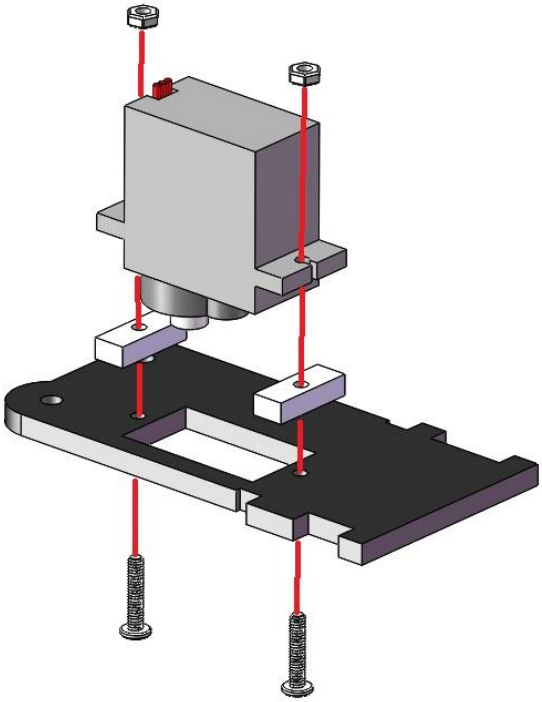
Acrylic Sheet 1PCS	Servo 1PCS	**Bag No.④** M2x10MM Screw 2PCS	**Bag No.⑧** M2 Nut 2PCS
			

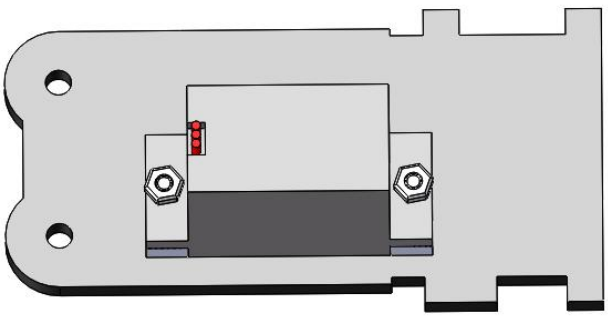




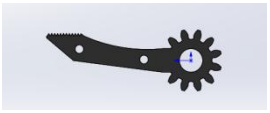
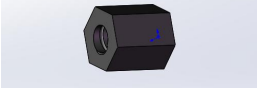
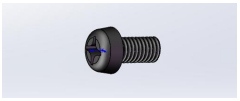
Step 10 Assemble the Servo D on Upper Arm

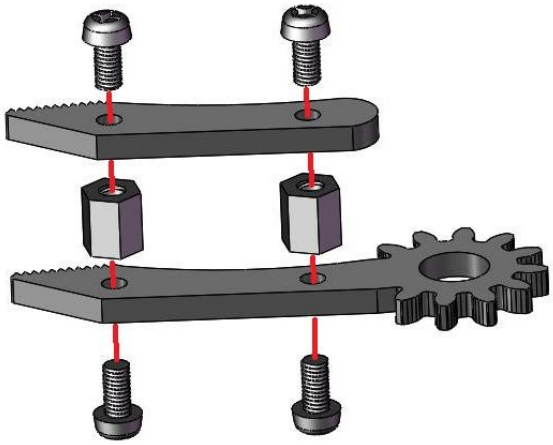
Acrylic Sheet 1PCS		Servo 1PCS
		
Acrylic Sheet 2PCS	**Bag No.④** M2x10MM Screw 2PCS	**Bag No.⑧** M2 Nut 2PCS
		

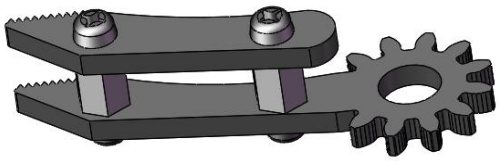




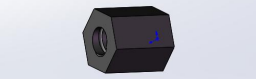
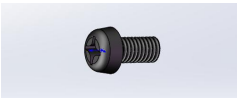
Step 11 Assemble the Left Clip Plate of The Clamp

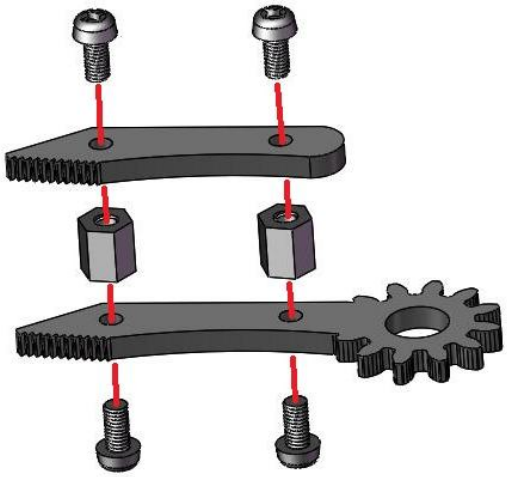
Acrylic Sheet 1PCS	Acrylic Sheet 1PCS	**Bag No.⑨** M3x7MM Standoff 2PCS	**Bag No.⑬** M3x6MM Screw 4PCS
			

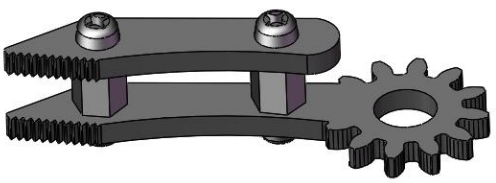




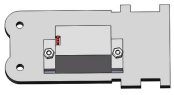
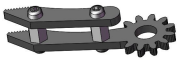
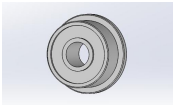
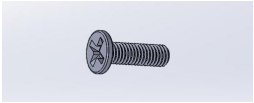
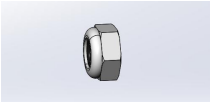
Step 12 Assemble the Right Clip Plate of The Clamp

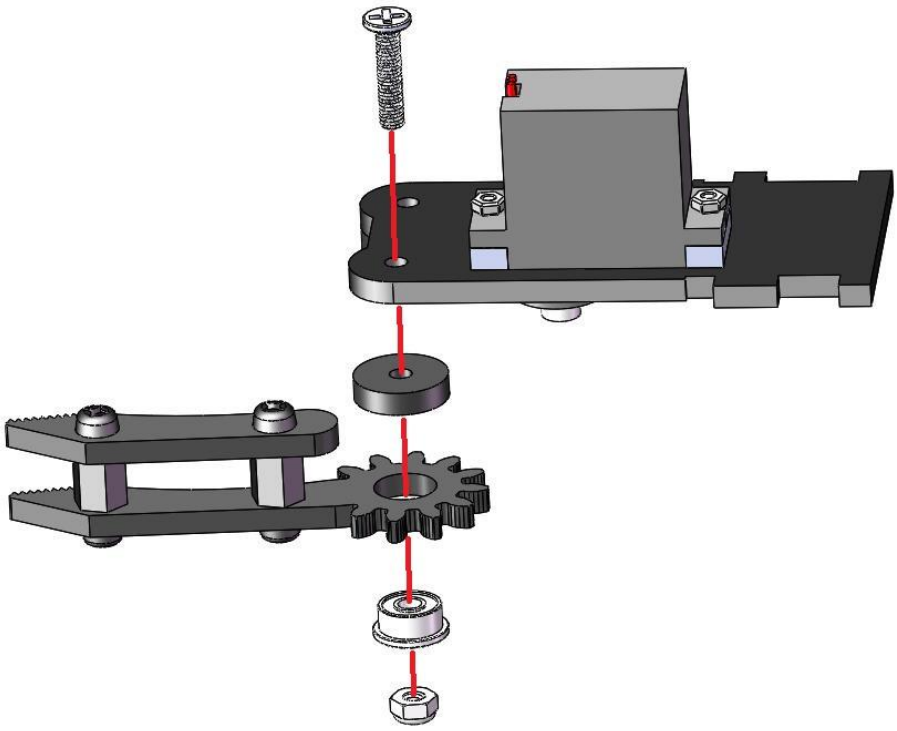
Acrylic Sheet 1PCS	Acrylic Sheet 1PCS	**Bag No.⑨** M3x7MM Nylon Column 2PCS	**Bag No.⑬** M3x6MM Nylon Screw 4PCS
			

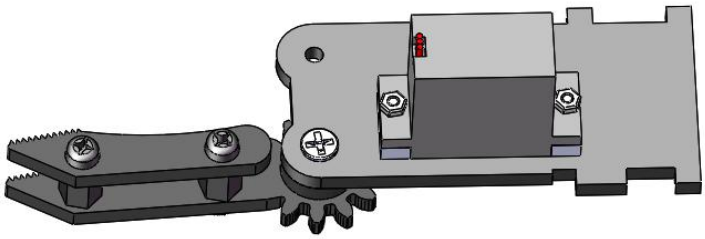




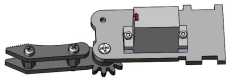
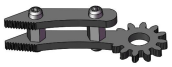
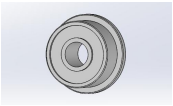
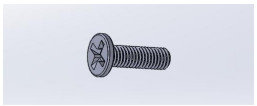
Step 13 Install the Left Clip Plate of The Clamp

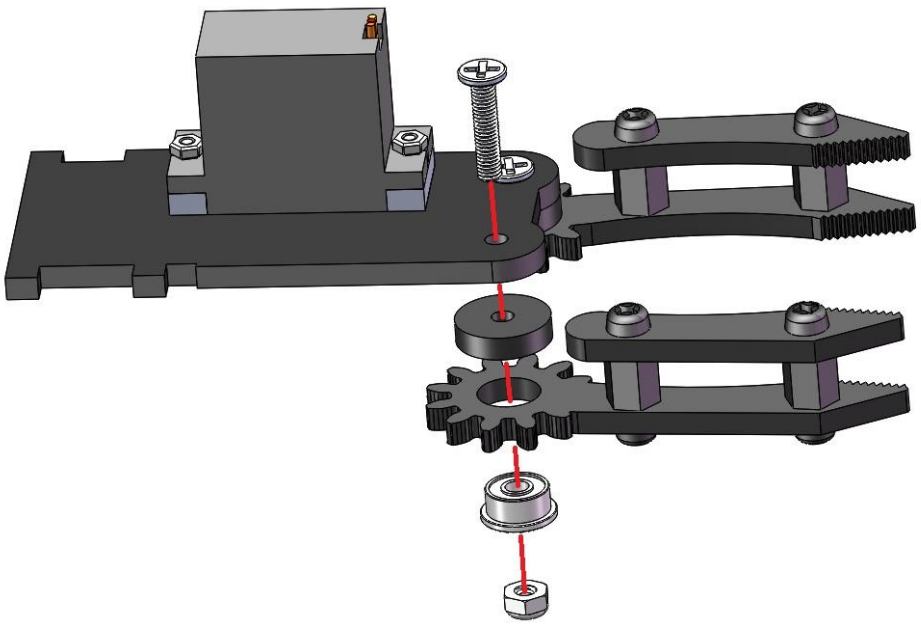
Step 10 Structure	Step 11 Structure	**Bag No.⑭** Flange Bearing 1PCS
		
Acrylic 1PCS	**Bag No.②** M3X14MM Screw 1PCS	**Bag No.⑦** M3 Lock Nut 1PCS
		

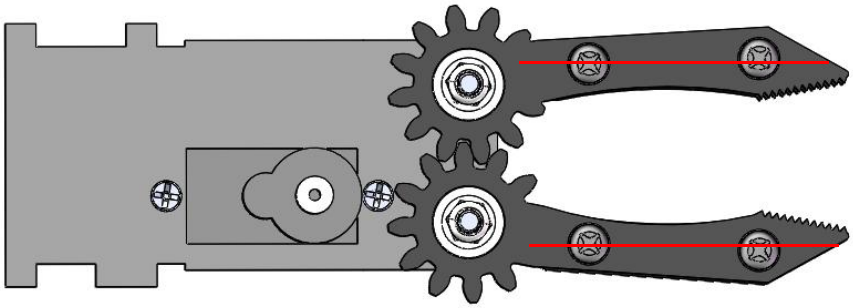




Step 14 Install the Right Clip Plate of the Clamp

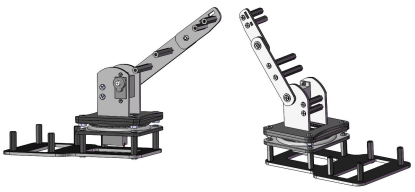
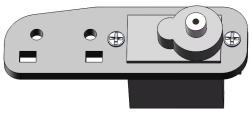
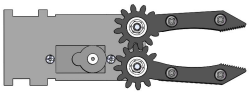
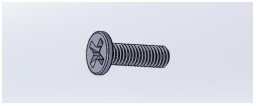
Step 13 Structure	Step 12 Structure	**Bag No.⑭** Flange Bearing 1PCS
		
Acrylic 1PCS	**Bag No.②** M3X14MM Screw 1PCS	**Bag No.⑦** M3 Lock Nut 1PCS
		

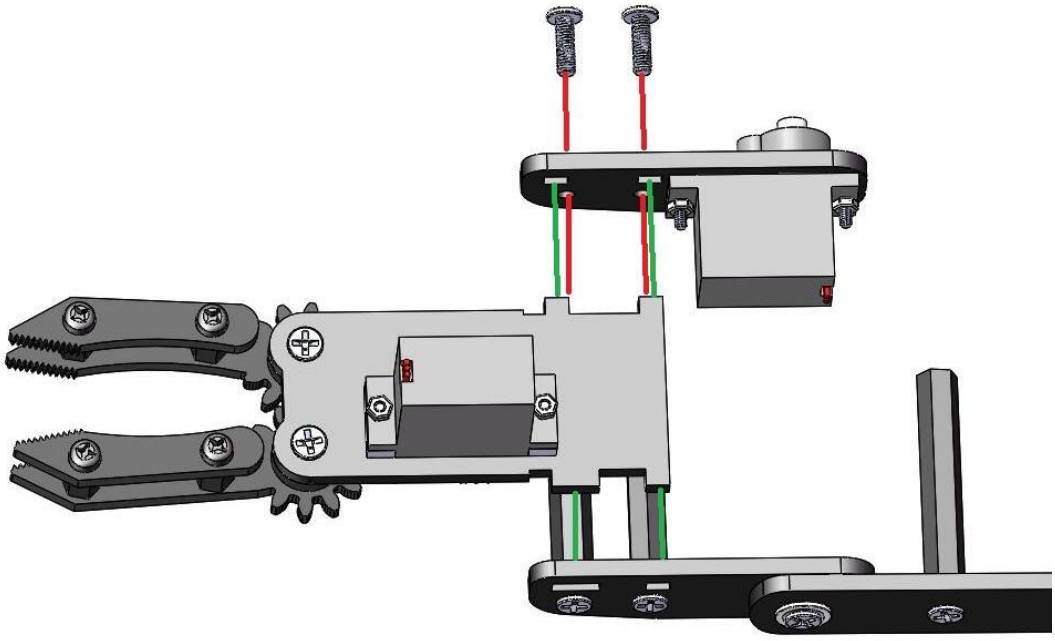


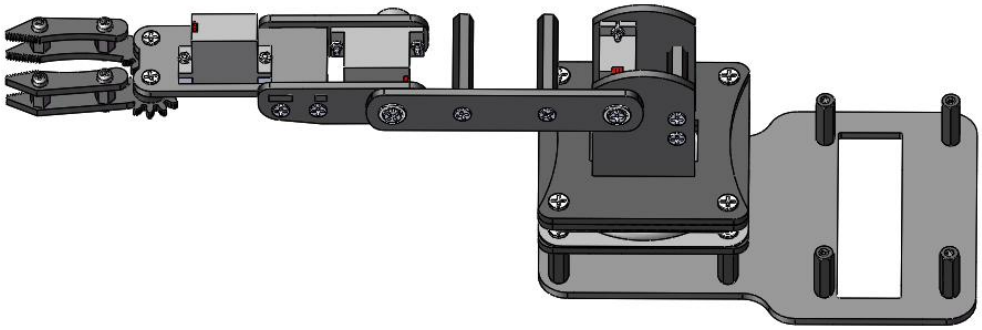


During installation, keep the screw threads horizontal

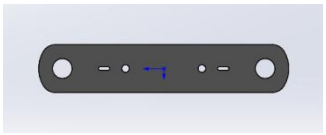
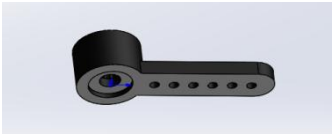
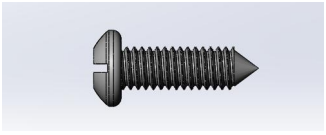
Step 15 Assemble the Clamp and Upper Arm

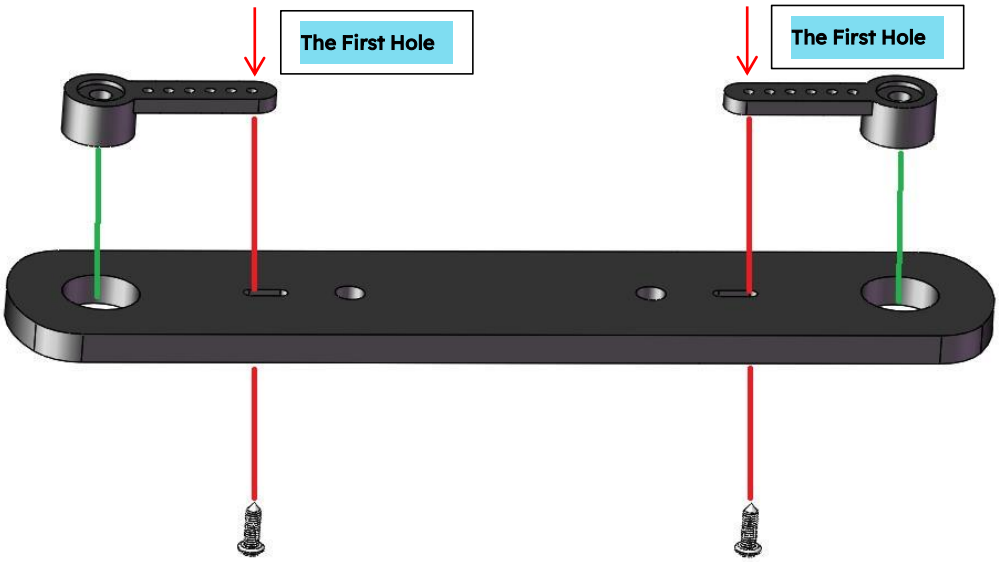
Step 8 Structure	Step 9 Structure
	
Step 14 Structure	**Bag No.③** M3X10MM Screw 2PCS
	



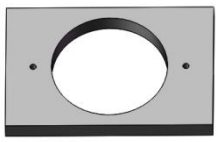
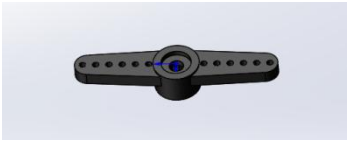
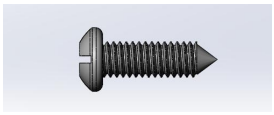


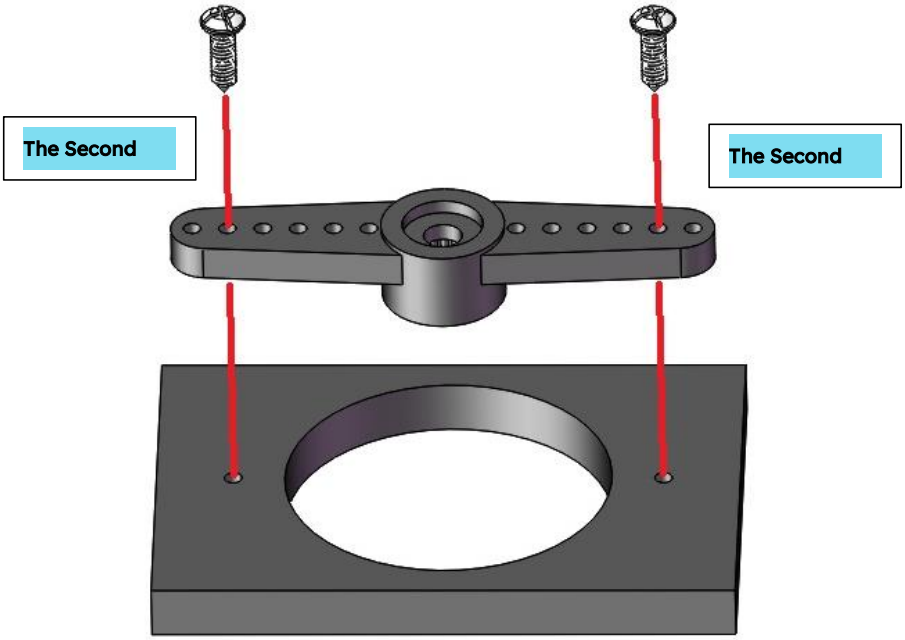
Step 16 Assemble the linkage between Servo B and Servo C

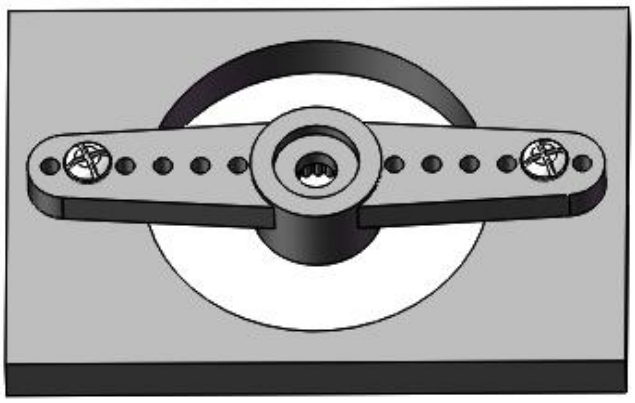
Acrylic Sheet 1PCS	Servo Arm 2PCS	**Bag No.⑤** M1.4X5 Lock Screw 2PCS
		



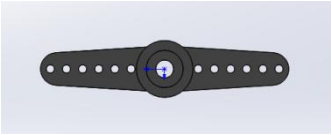
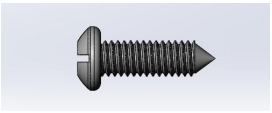
Step 17 Assemble the Mounting Structure for Servo A

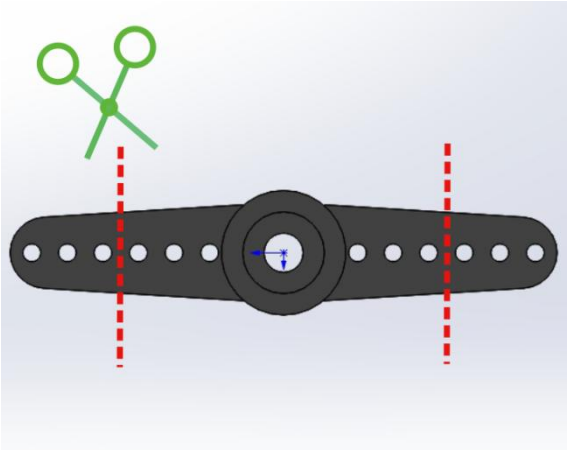
Acrylic Sheet 1PCS	Servo Arm 1PCS	**Bag No.⑤** M1.4X5 Lock Screw 2PCS
		



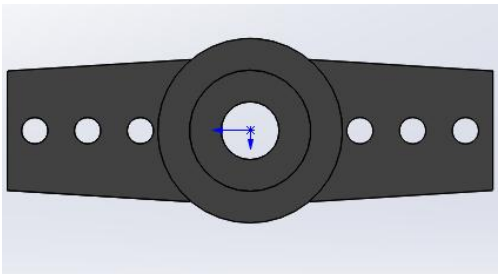


Step 18 Assemble the Mounting Structure for Servo D

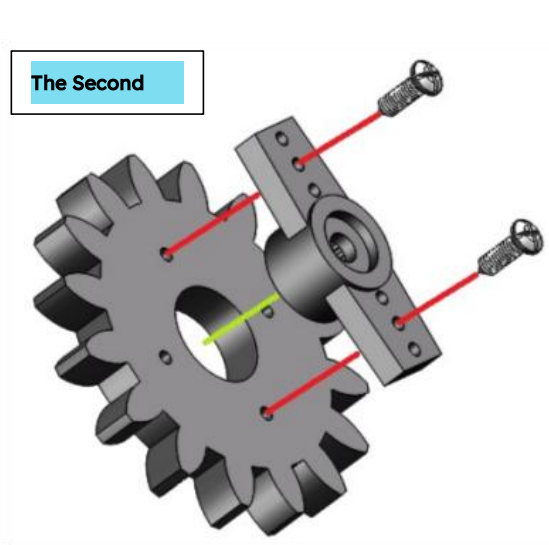
Acrylic Sheet 1PCS	Servo Arm 1PCS	**Bag No.⑤** M1.4X5 Lock Screw 2PCS
		



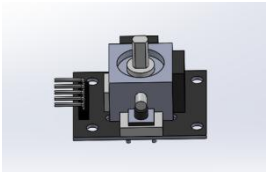
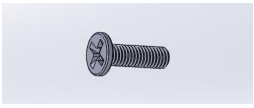
Pre-cut before factory shipment.

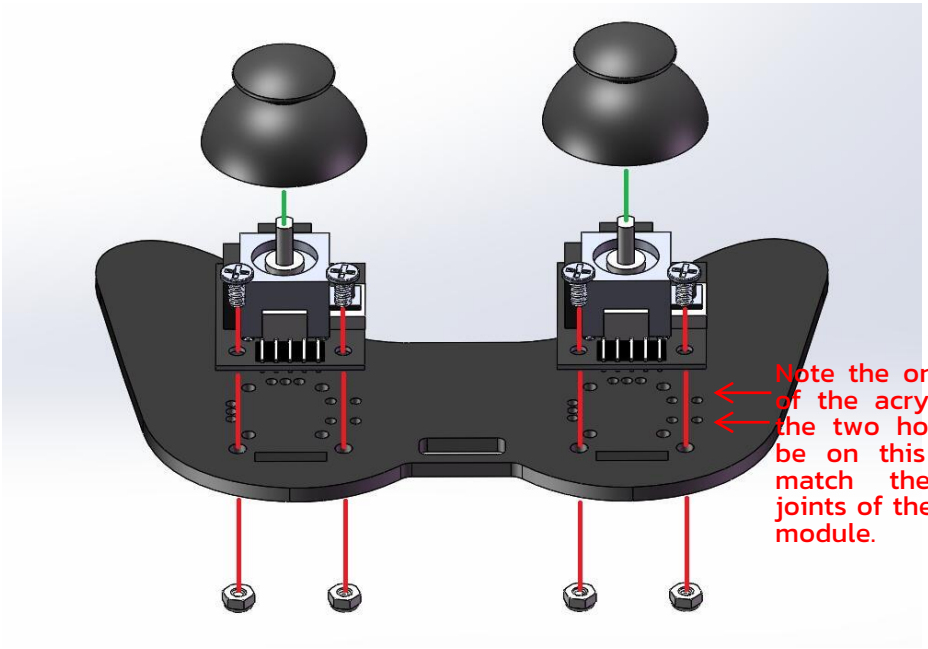


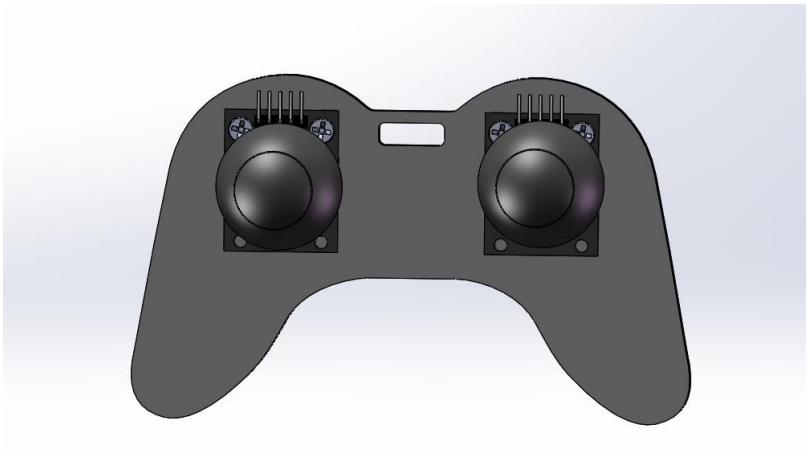
The Second



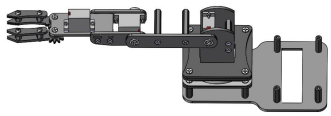
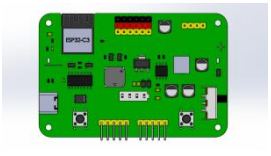
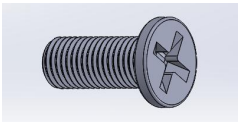
Step 19 Assemble the Joystick

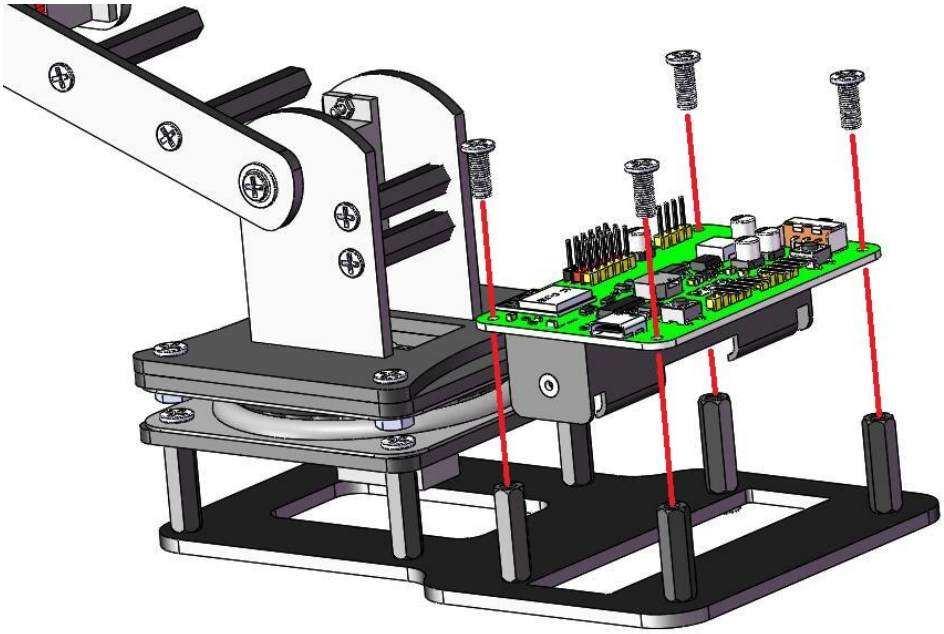
Acrylic Sheet 1PCS	Joystick Module 2PCS	
		
Joystick Hat 2PCS	**Bag No.③** M3X10MM Screw 4PCS	**Bag No.⑦** M3 Lock Nut 4PCS
		

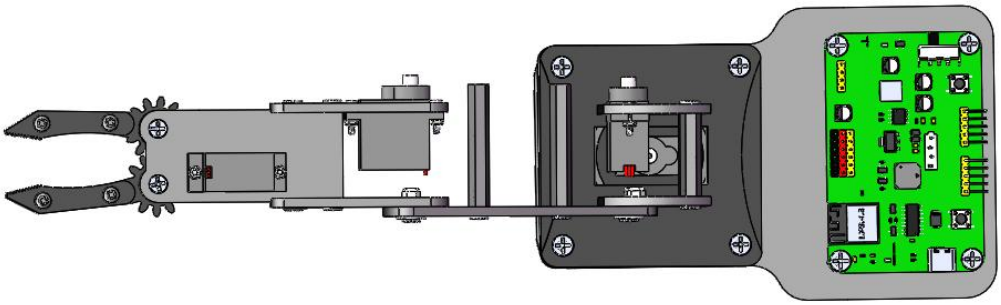




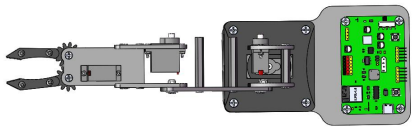
Step 20 Install the Esp32 Board

Step 15 Structure	ESP32 Control Board 1PCS	**Bag No.①** M4X10MM Screw 4PCS
		

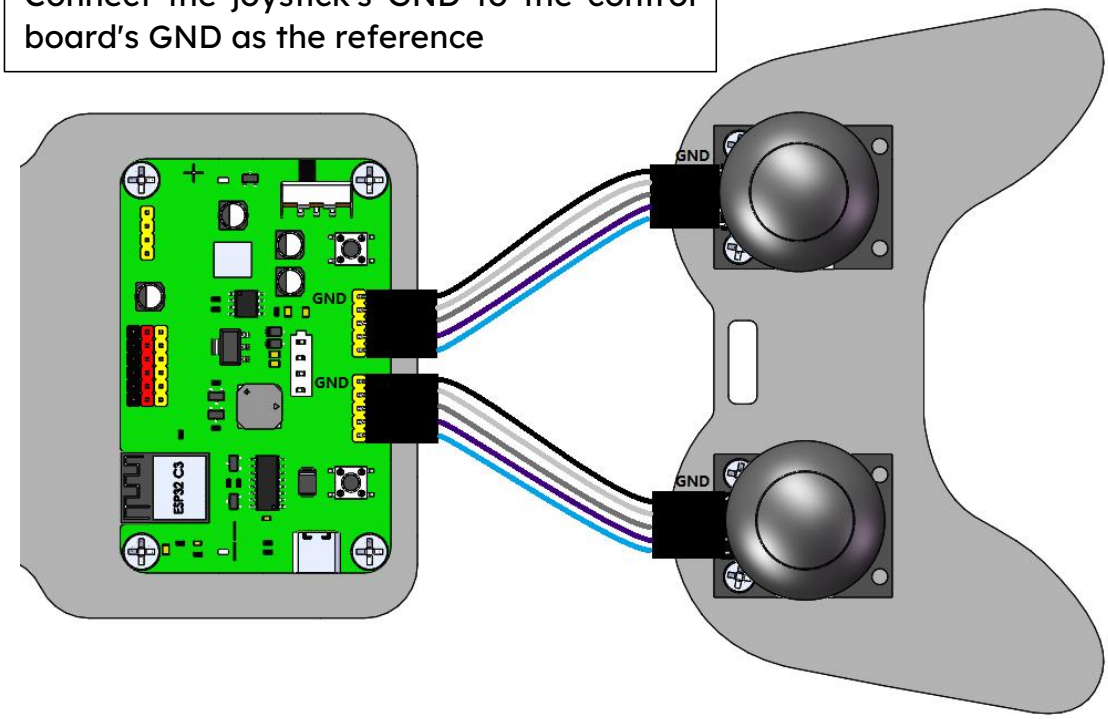


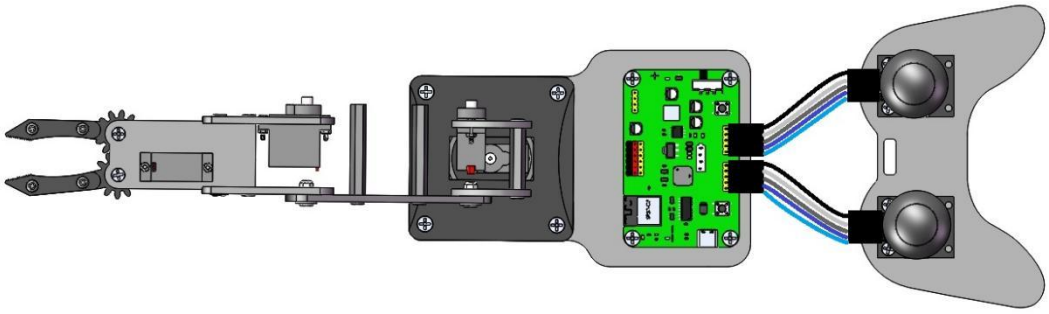


Step 21 Connect the Joystick

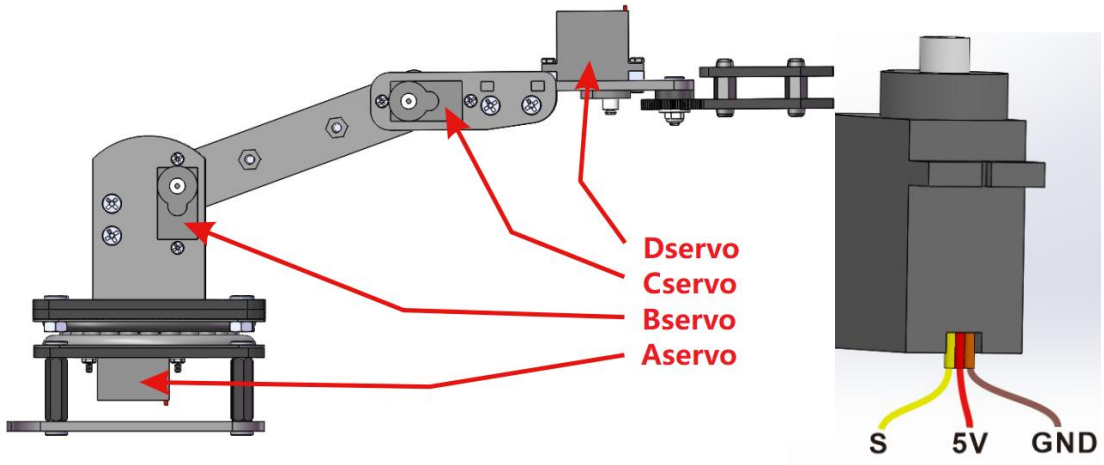
Part in Step 19	Part in Step 20
	
5P Dupont Wire 2PCS	
	

Connect the joystick's GND to the control board's GND as the reference

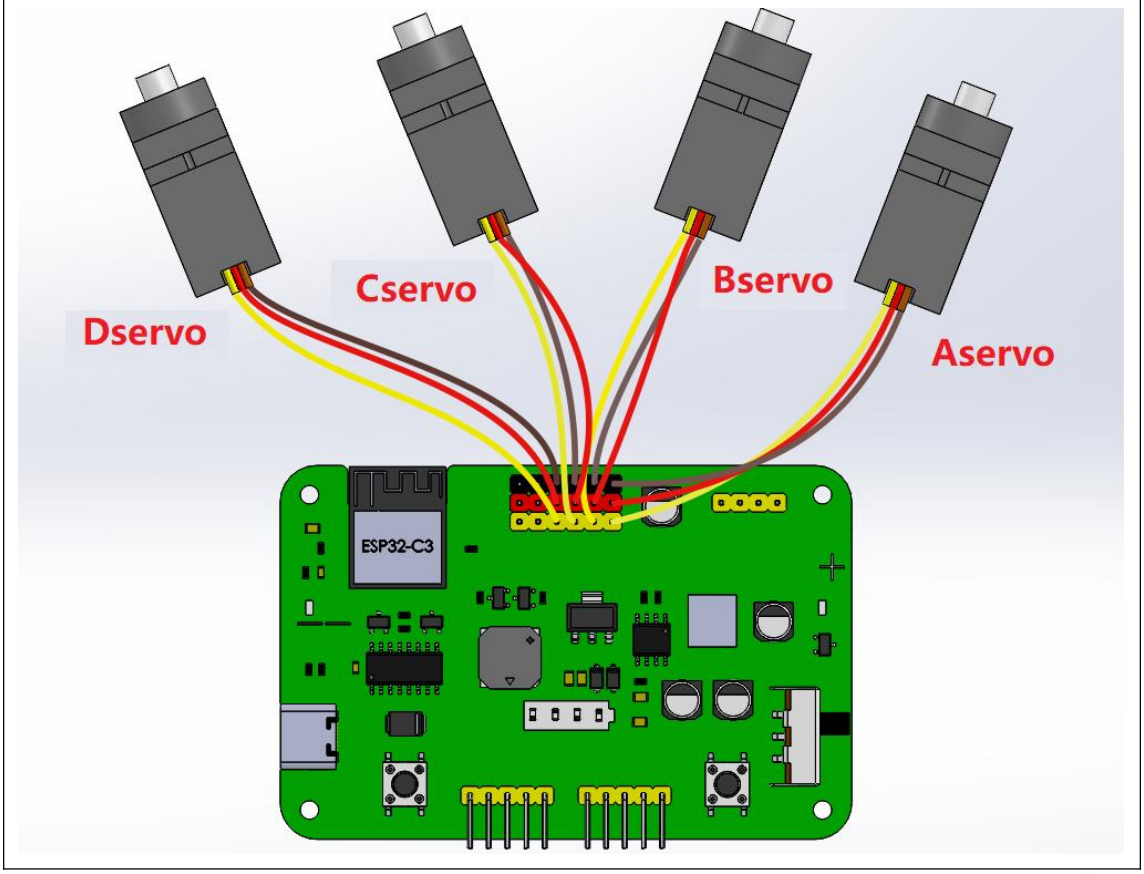


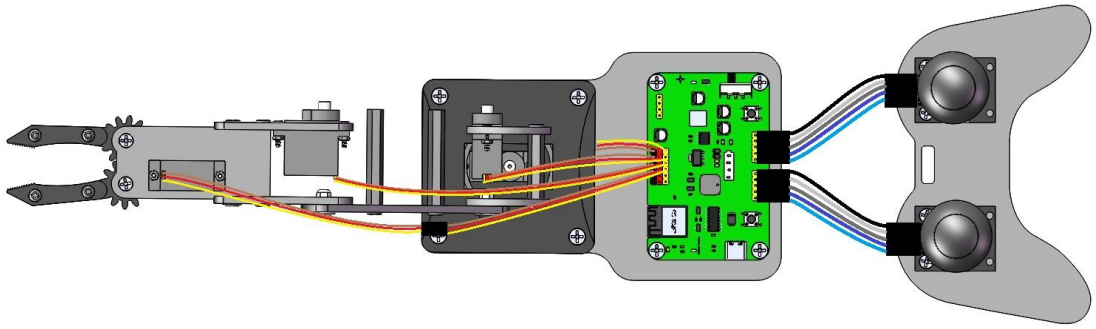


Step 22 Connect the Servos to the ESP32 Board



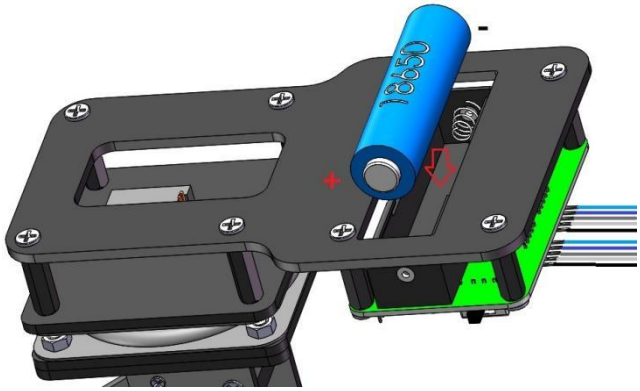
Dservo=GPIO7	Cservo=GPIO6	Bservo=GPIO5	Aservo=GPIO4



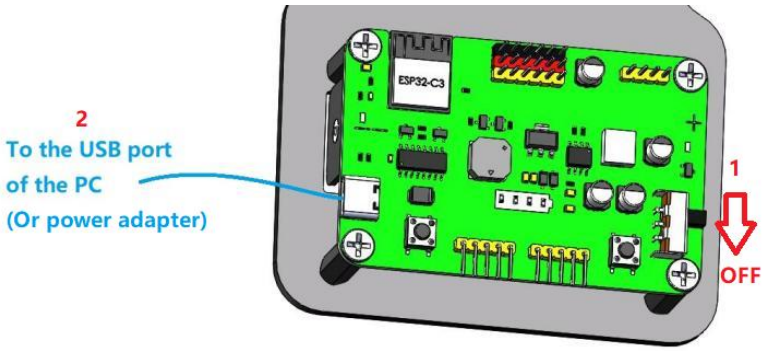


Step 23 Initialize the Servo Angle (important!)

1) Install a 18650 lithium battery (need to be purchased by yourself)

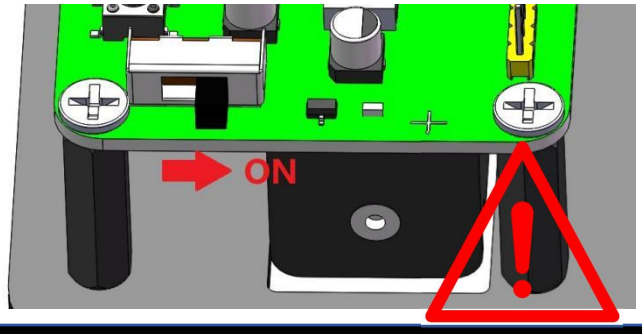


2) When installing the battery for the first time, please use the USB cable to charge the battery to activate the battery!



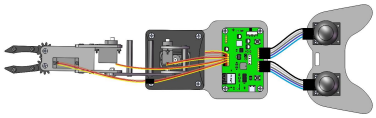
3) Very important: Turn on the power switch ! ! !

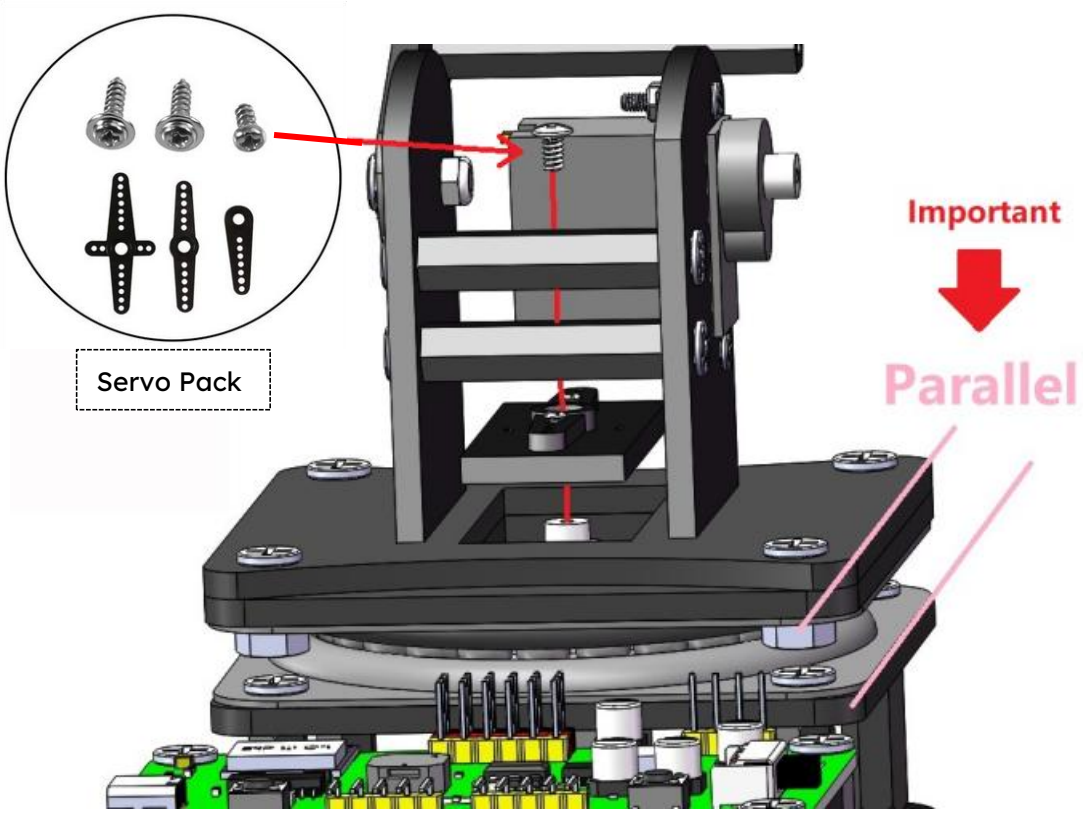
The firmware has been successfully programmed into the control board. When powered on, the board will automatically position all servo motors at the 90-degree default position.

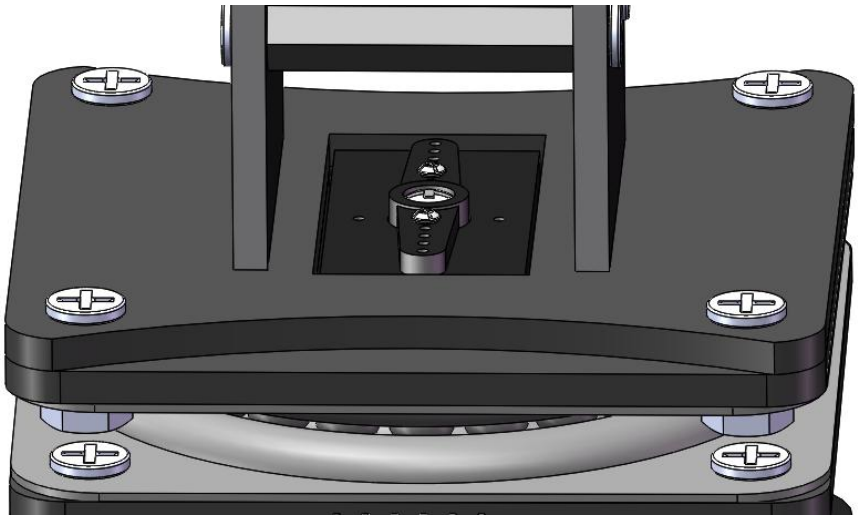


- Make sure the battery is fully charged!
- When you turn on the power switch, You will hear the servo motor shaft rotating.
- In the following installation steps, please keep the power on to prevent the servo angle from being changed!

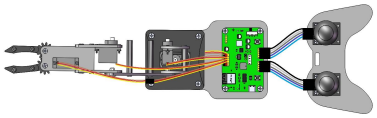
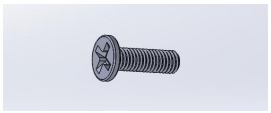
Step 24 Fix the Robot Arm to Servo A of the Base

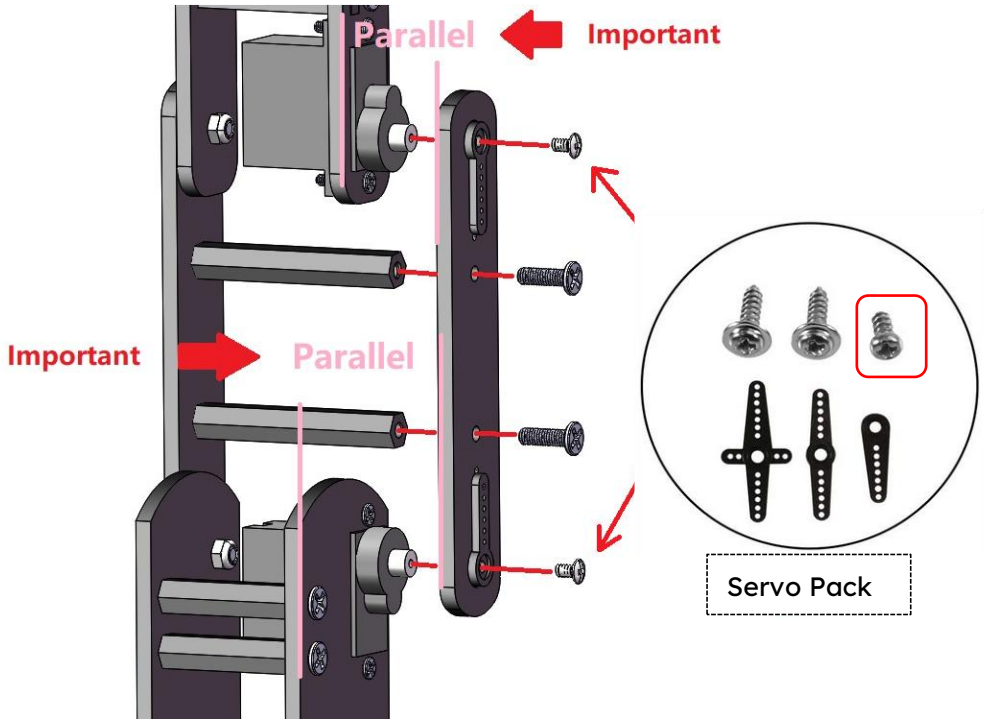
Step 23 Structure	Step 17 Structure	M2.5X4mm Screw 1PCS
		

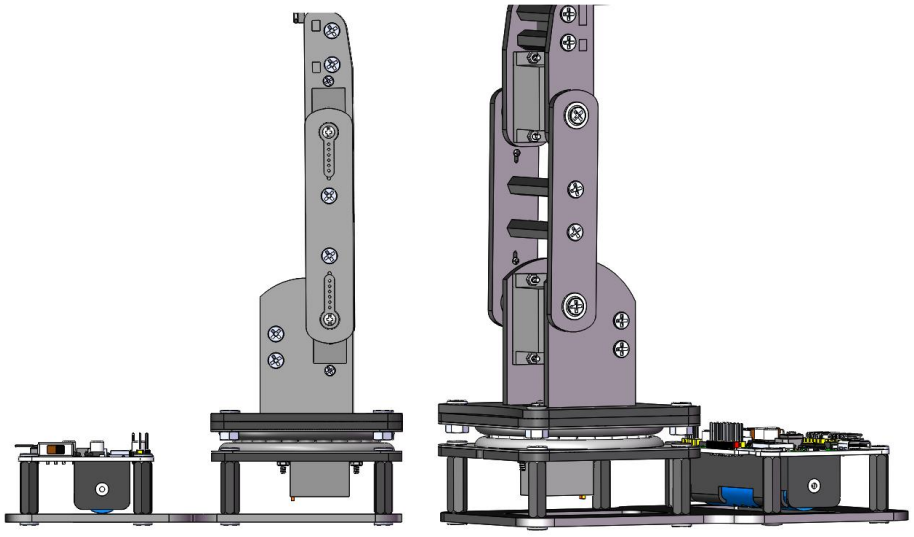




Step 25 Assemble the Middle Arm

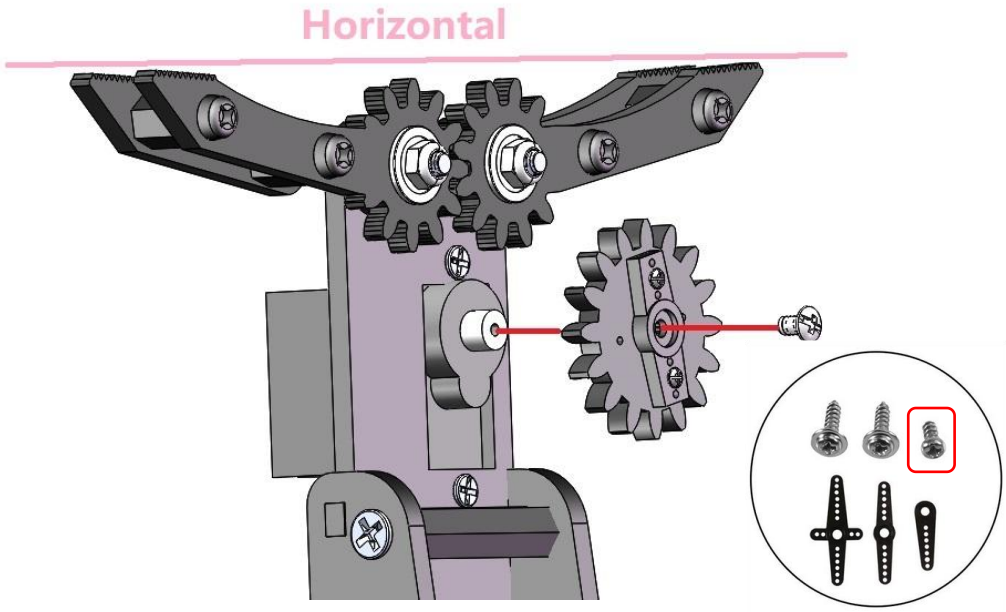
Step 24 Structure	Step 16 Structure
	
Bag No.③ M3X10MM Screw 2PCS	M2.5X4mm Screw 2PCS
	

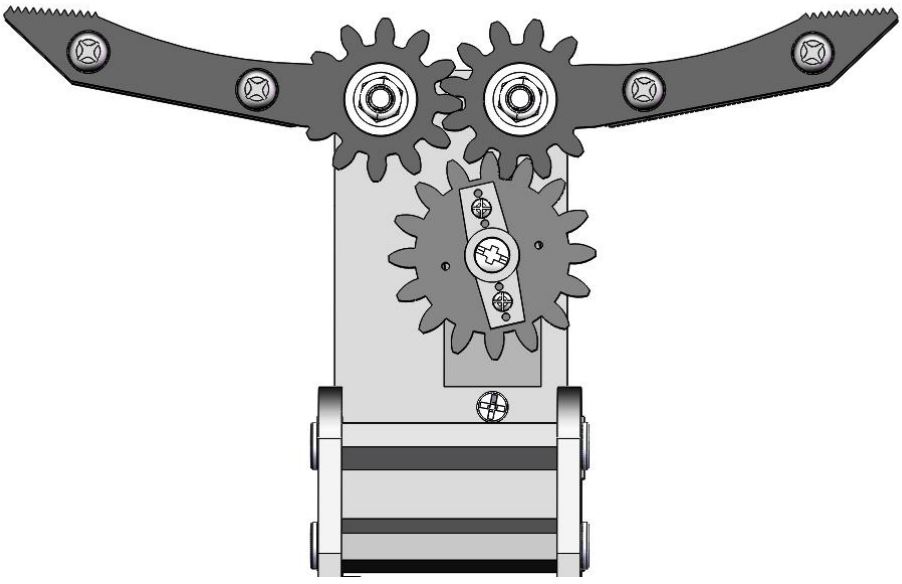




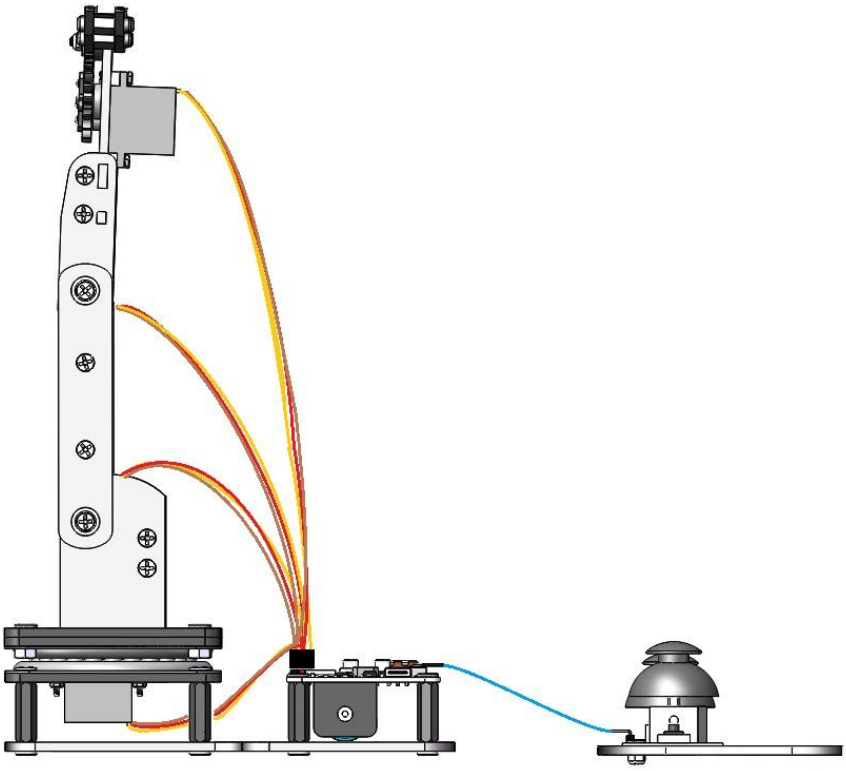
Step 26 Fix the Clip Plates to Servo D

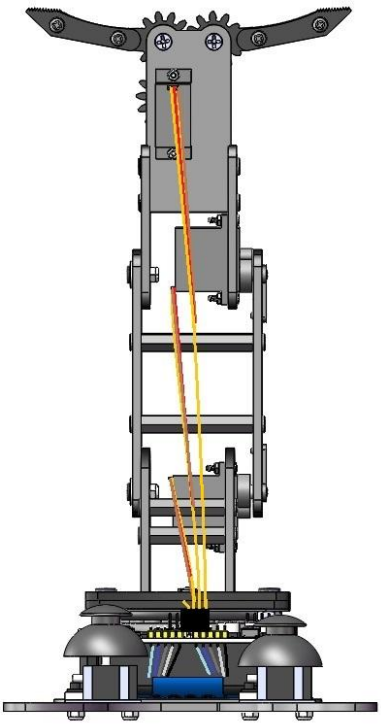
Step 25 Structure	Step 18 Structure	M2.5X4mm Screw 1PCS
		





Step 27 Assembly completed





3

Using the Robot Arm

3.1 Robotic Arm Safety and Operating Guidelines

To ensure optimal performance and maximize the service life of your ESP32 Robotic Arm, please read and follow these guidelines carefully.

1. Primary Operating Precautions

Do Not Overload: This arm is designed for lightweight tasks. The maximum recommended load for the gripper is 40 grams.

Maintain a Clear Workspace: Ensure the area around the robotic arm is free of obstacles to allow unobstructed movement of all joints.

Avoid Forced Movement: Never manually force or twist any joint while the arm is powered on, as this may damage internal gears and servo motors.

2. Critical Gripper Maintenance (Servo Protection)

Continuously gripping an object under load places significant stress on the servo motor (particularly the D-axis driving the gripper), causing it to generate heat. Therefore, we strongly recommend limiting continuous load-bearing gripping to no more than 40 seconds. After completing a gripping operation, release the object immediately by opening the gripper to allow the motor to cool down and rest. Exceeding this duration is the primary cause of motor overheating and burnout.

Auto-Protection Function: Our latest optimized firmware/code includes a protective feature. If the gripper servo does not receive a new command within 40 seconds, it will automatically disengage (cease PWM signal transmission) to prevent overheating.

Best Practice: Actively control the gripper. When not manipulating objects, open the gripper or send commands to maintain it in a relaxed state.

3. Routine Maintenance

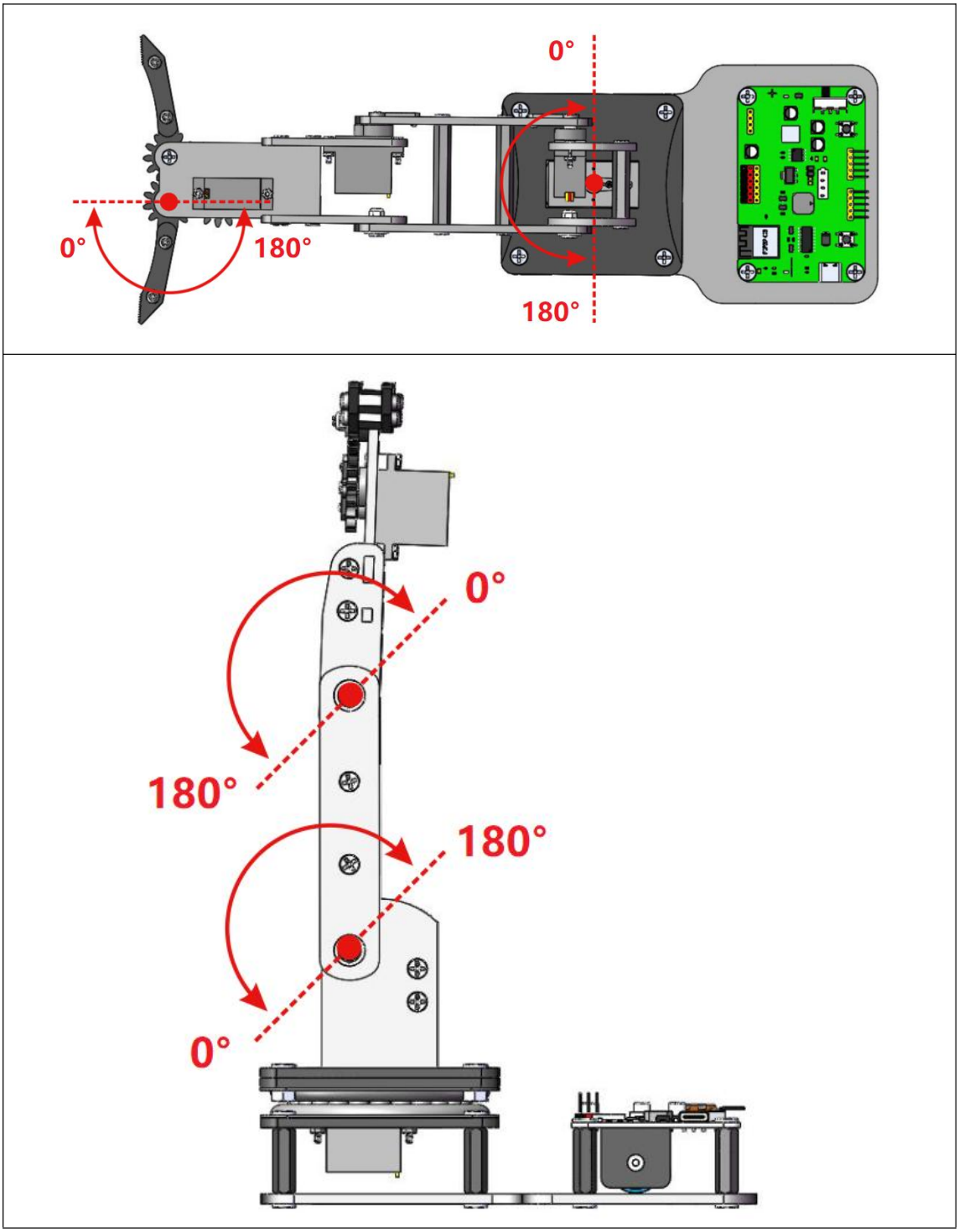
Power off the unit when not in use.

Periodically check and tighten all screws and connections.

Operate the robotic arm on a stable, flat surface.

Following these instructions will help ensure a reliable and durable robotic experience.

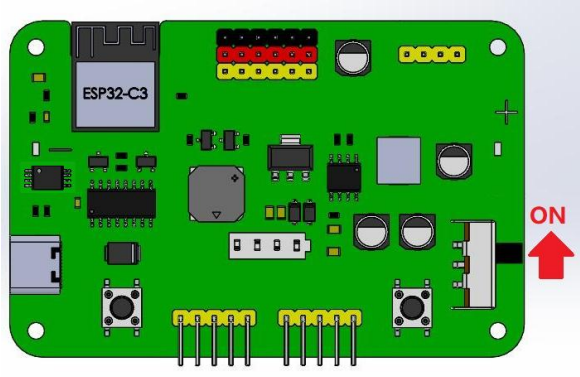
3.1 Degrees of Freedom

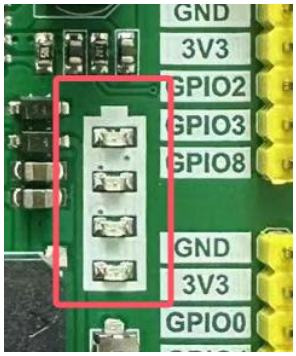


3.2 Control the eArm with a Joystick

Turn the eArm's power switch to the **ON** position.

- 1: Make sure the 18650 lithium battery is installed.
- 2: The battery indicator shows 3 bars or more.

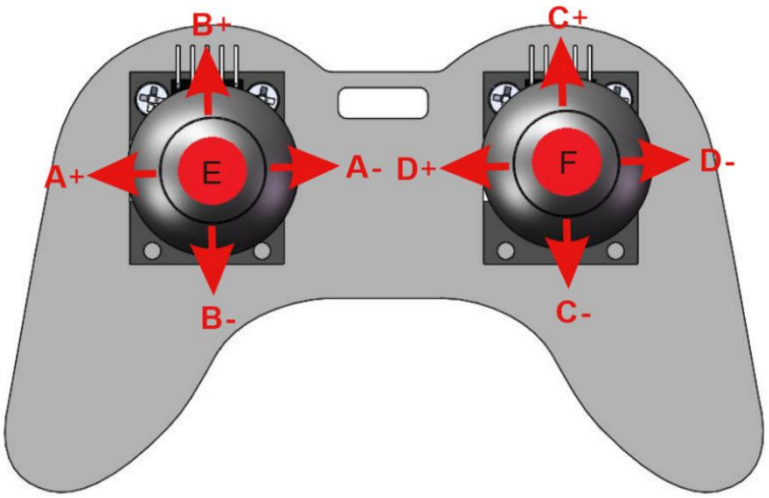




After powering on the robot, the default control mode is **Joystick Mode**. Available modes include:

- ☑ Joystick Mode (default)
- ☑ Web App Mode

To switch between control modes, press the **"F"** button on the right joystick. When you press "**F**", the buzzer will beep once, indicating the switch to **Web App Mode**. In this mode, joystick controls will be **disabled**.



A+: Rotate arm left (Servo A)
A-: Rotate arm right (Servo A)
B-: Raise rear arm (Servo B)
B+: Lower rear arm (Servo B)
C-: Raise the forearm (Servo C)
C+: Lower the forearm (Servo C)
D+: Open the gripper (Servo D)
D-: Close the gripper (Servo D)

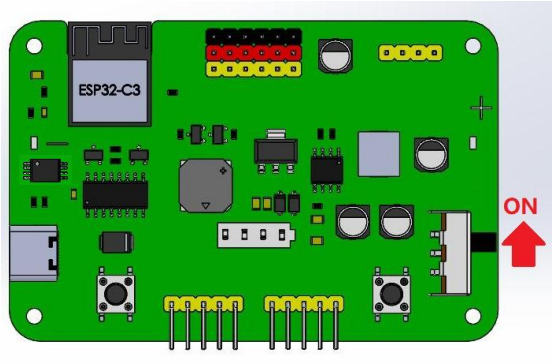
F: Toggle between Joystick and Web App control modes
E: Enable/disable the buzzer

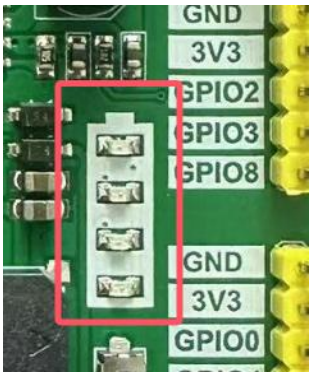
3.3 Control the eArm with Web App

1) Turn the eArm's power switch to the ON position.

■: Make sure the 18650 lithium battery is installed.

■: The battery indicator shows 3 bars or more.





2) Turn on the phone's Wi-Fi and connect to the network named 'eArm'

■: Once connected to the Wi-Fi, you may see a 'Network connection failed' message; just disregard it.

■: Turn off mobile data in your phone settings

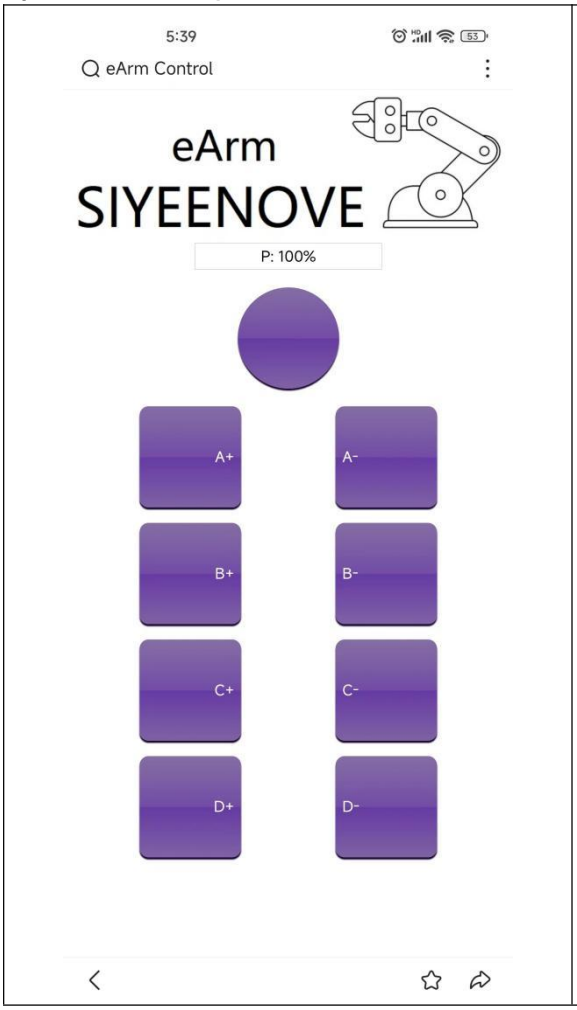
■: This ensures stable connection to the Web App

3) Open the web browser on your phone

4) Type 192.168.4.1 into the address bar

5) Press Enter to connect

After successful connection, the control interface will appear in your browser - you can now operate the eArm!



Button	Mechanical Movement
A+	Robot arm turns left
A-	Robot arm turns right
B-	Rear arm raises
B+	Rear arm lowers
C-	Front arm raises
C+	Front arm lowers
D+	Gripper opens
D-	Gripper closes

4

Programming Learning

4.1 Before You Begin: Important Notes

The ESP32-C3 control board comes preloaded with firmware that enables robotic arm operation as demonstrated in the previous section. This functionality will remain available until you Flashing MicroPython Firmware using Thonny IDE.
Note: flashing your boards with MicroPython is reversible. This means that after you flash the ESP32/ESP8266 with MicroPython, you can still use Arduino IDE in the future. You just need to upload a code using the Arduino IDE to your board.

We provide python example code in the tutorial package you download. Please remember the file path where it is stored. Remember where you saved it (the file path).

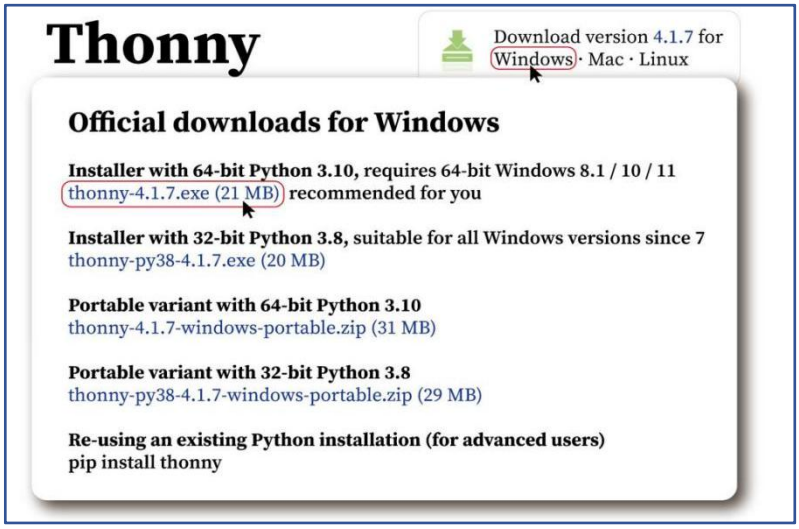
4.3 Install Thonny IDE

Official website of Thonny: https://thonny.org

Thonny supports three mainstream operating systems: Windows, MacOS, and Linux. Follow the section for the operating system you're using.

Operating System	Installation Methods
Windows	Ⅰ) Installing Thonny IDE – Windows PC
Mac OS	Ⅱ) Installing Thonny IDE – Mac OS

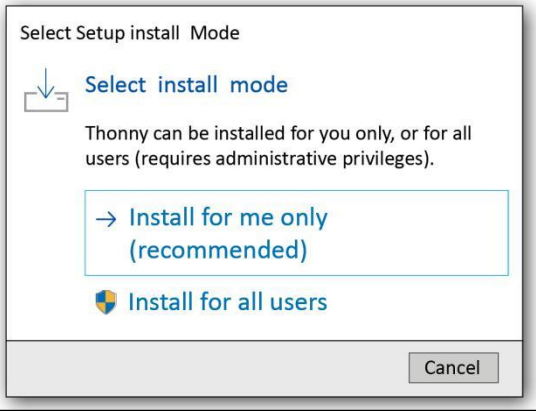
I) Installing Thonny IDE – Windows PC



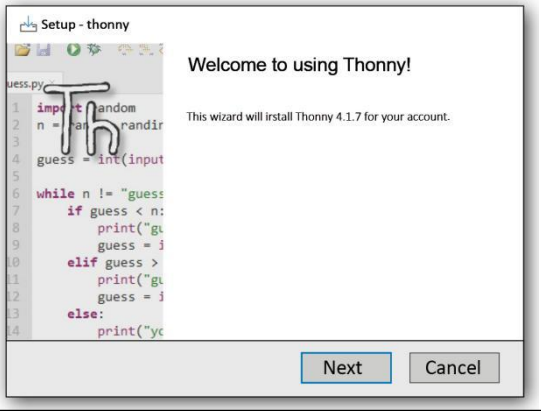
1. Download the installer thonny-4.1.7.exe, double-click to run it.

2. Keep click "Next" to complete the installation.

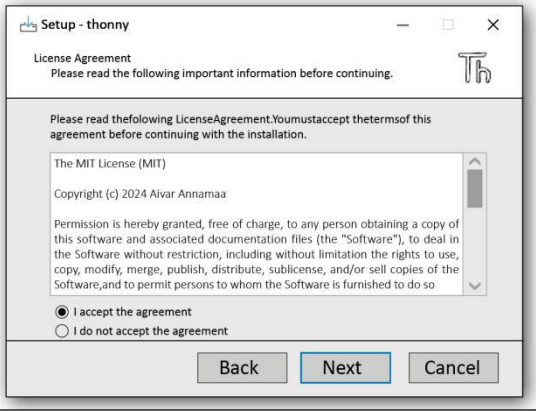
1. Click Install for me only



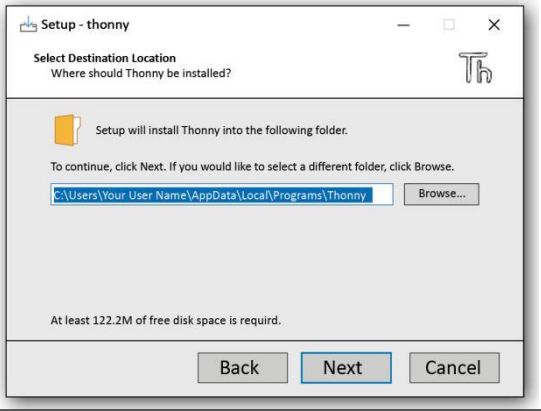
2. Click Next



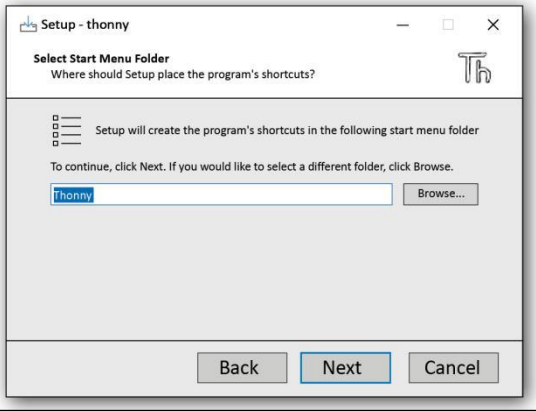
3. Click Next



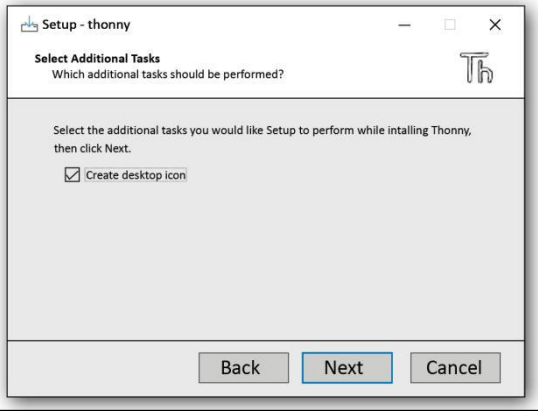
4. Click Next or click "Browse" to modify the default installation path.



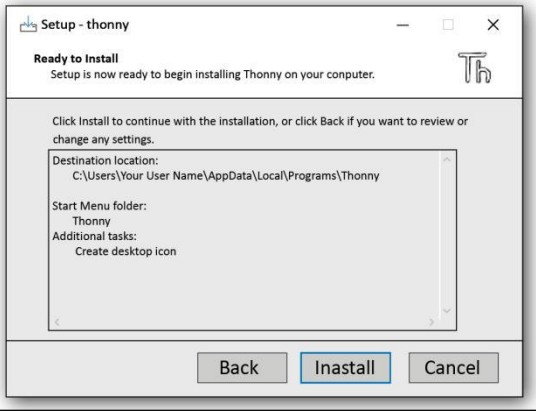
5. Click Next



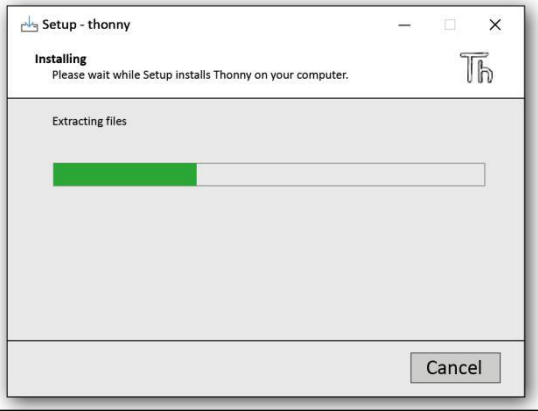
6. Check Create desktop icon, click Next



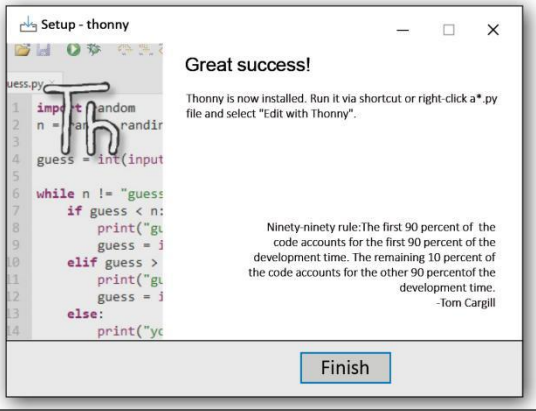
7. Click Install



8. Wait a few seconds for the installation to complete.



9. Finish



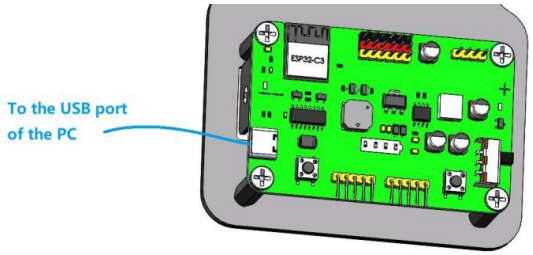
3. After installation, find the desktop application icon to run it.

4.4 Install CH340 USB driver

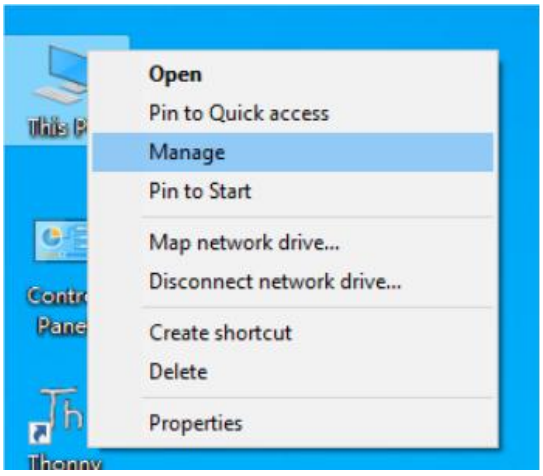
Operating System	Installation Methods
Windows	Ⅰ) **Install the Driver on Windows system**
Mac OS	Ⅱ) Install the Driver on Linux system
Linux	Ⅲ) Install the Driver on MAC system

Ⅰ) Install the Driver on Windows system

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

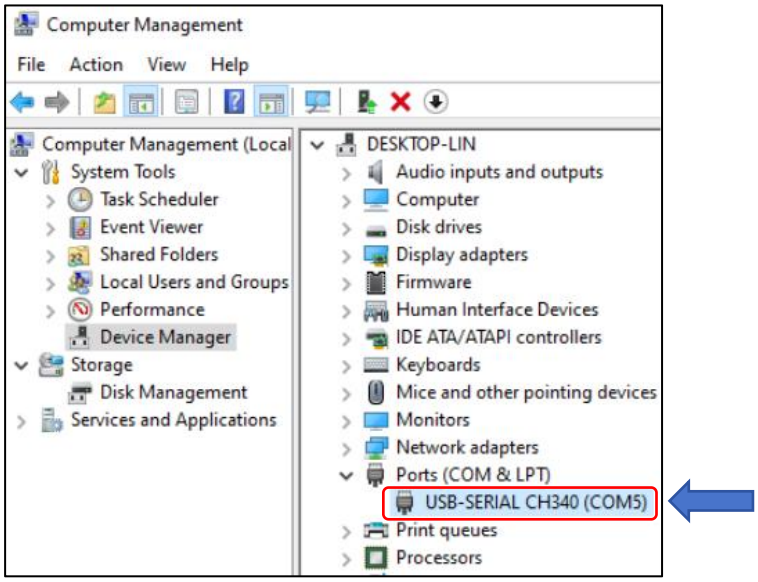


2. Turn to the main interface of your computer, select “This PC” and right-click to select “Manage”



3.Check whether CH340 has been installed

If the driver is properly installed, when the ESP32 board is connected to the computer via a USB cable, the device "USB-Serial CH340 (COMX)" can be found under the path "Device Manager" → Ports (COM & LPT), as shown in the figure below.

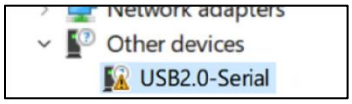


Note: Driver display symbol is “USB-SERIAL CH340 (COMx)”, “x” can be any number, it depends on what number your computer assigns to the ESP32 device.

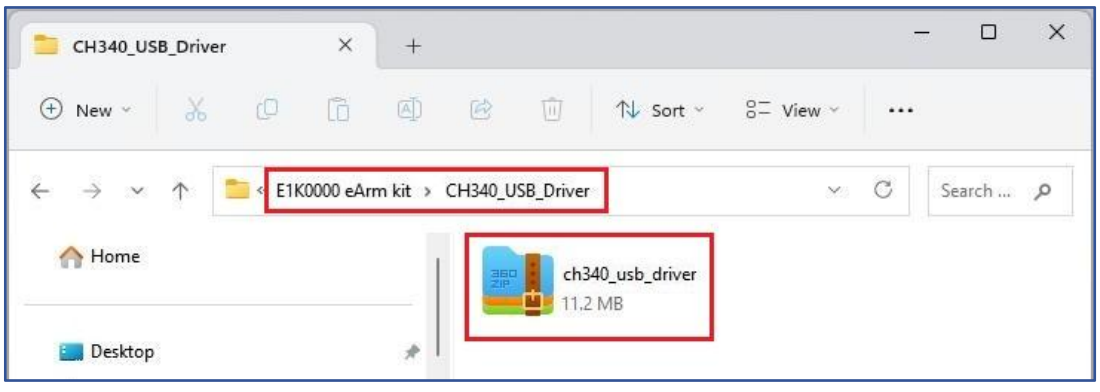
If the driver is already installed, skip the driver installation step and proceed to the next chapter.

4.Install CH340 driver manually

If the CH340 driver is not installed, the Windows Device Manager typically shows a "USB2.0-Serial" or "Unknown Device" with a yellow exclamation mark under "Ports (COM & LPT)" or "Other Devices". This indicates that the computer has detected the hardware but cannot use it properly. As a result, a COM port number will not be assigned, preventing serial communication.

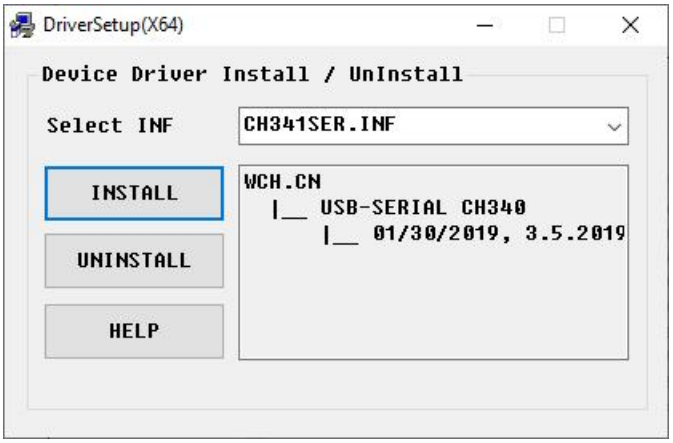


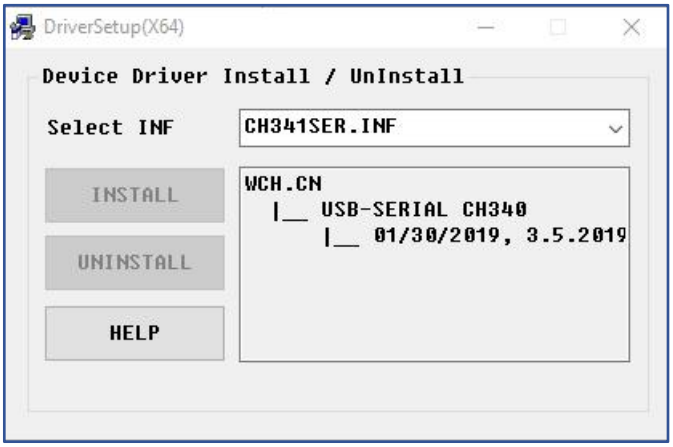
Driver file is provided in the tutorial package:



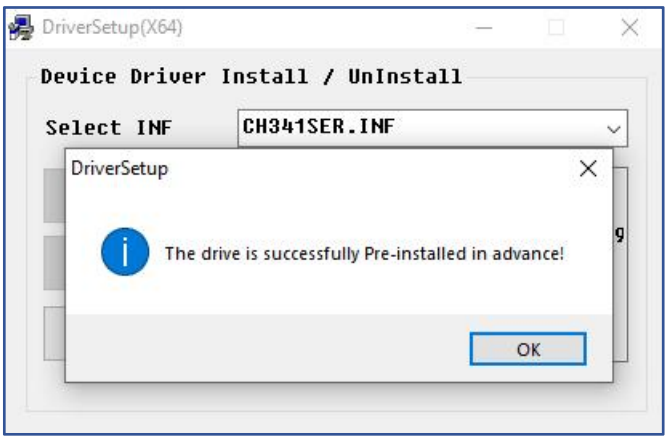
Unzip the file we provided and double-click **"SETUP"** to run the installer:

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





Install successfully. Close all interfaces.

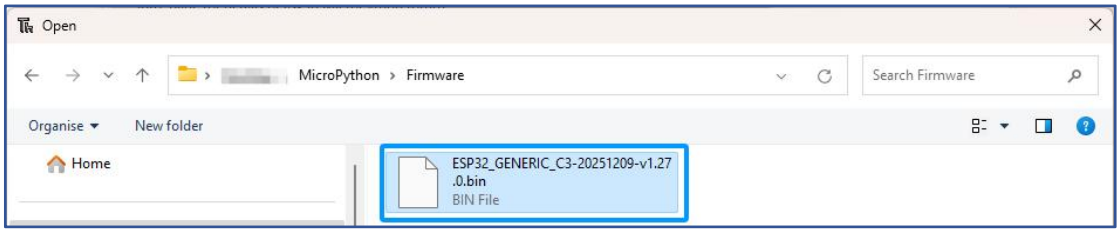


4.5 Flashing MicroPython Firmware using Thonny IDE

To upload code to an ESP32 from Thonny IDE, you must first ensure the ESP32 is flashed with MicroPython firmware and Thonny is configured to use the MicroPython interpreter.

Firmware used in this tutorial is ESP32_GENERIC_C3-20251209-v1.27.0.bin

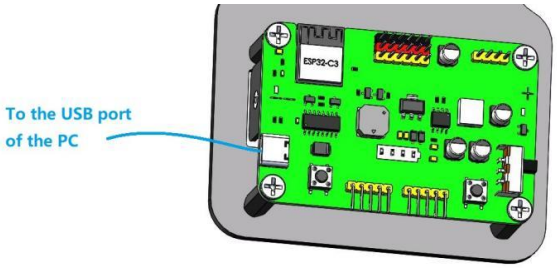
This file is also provided in our tutorial folder : "MicroPython > Firmware".



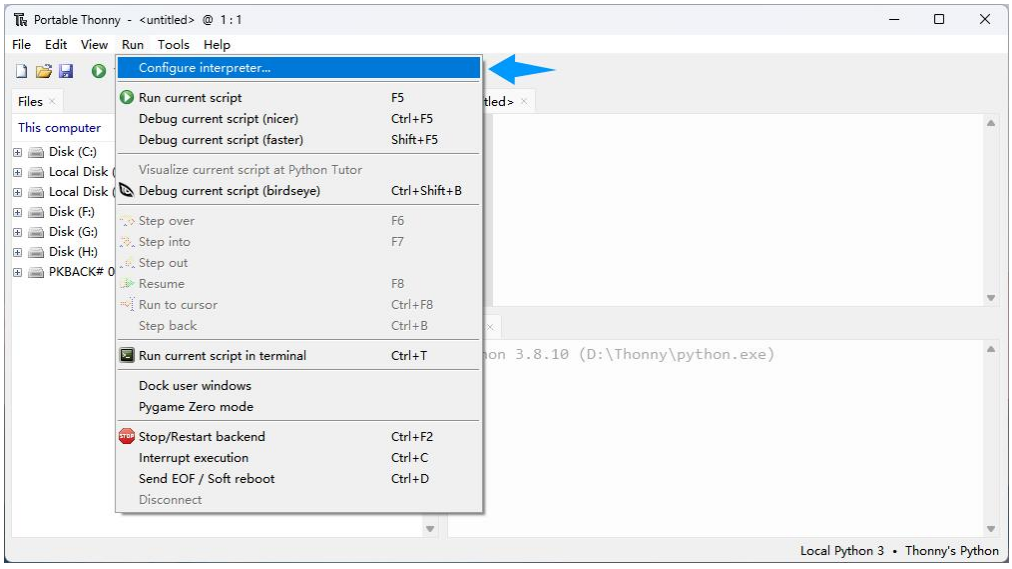
Or download it from the official website
List of microPython firmware for ESP32-C3:
https://micropython.org/download/ESP32_GENERIC_C3/

Follow the next steps:

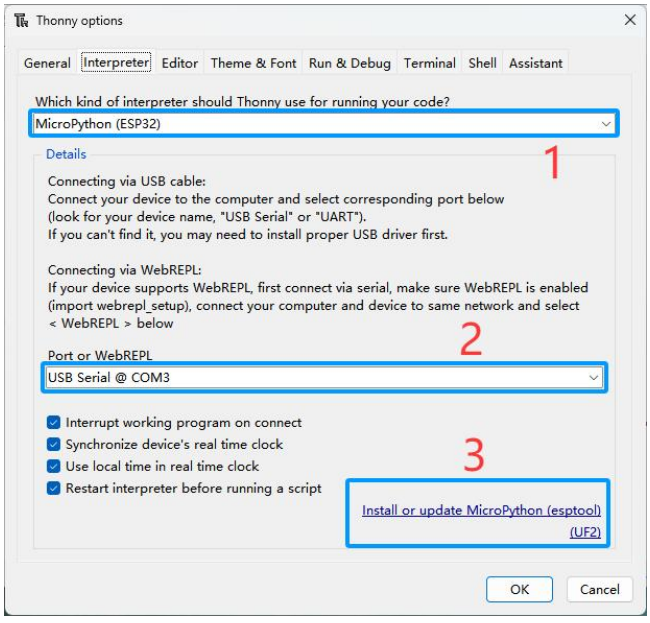
1) Connect your ESP32 board to your computer.



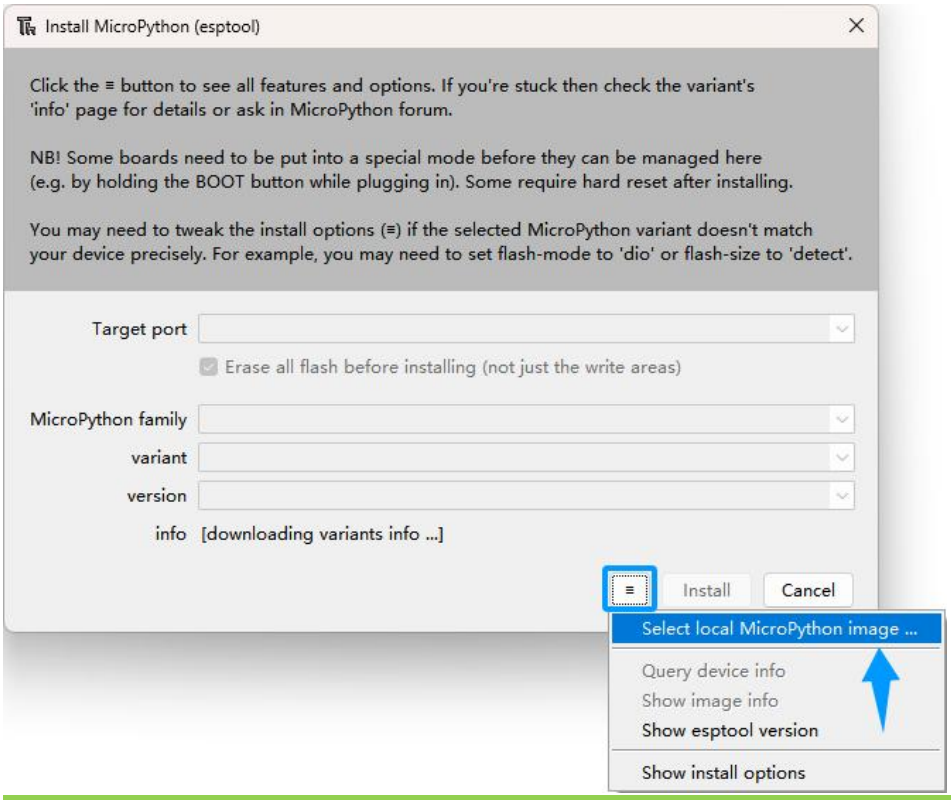
2) Open Thonny IDE. Click “Run” and select “Configure interpreter...”



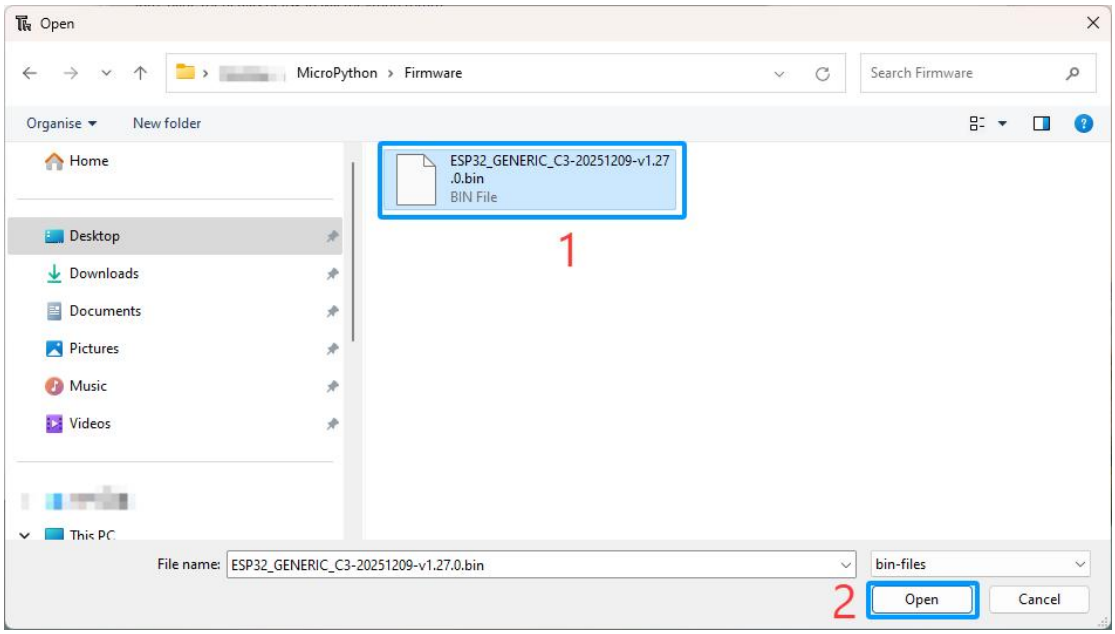
3) Select “Micropython (ESP32)”, select “USB Serial @ COMX(Depending on the number assigned by the Device Manager”, and then click the long button under “Install or update MicroPython(esptool)(UF2)”.



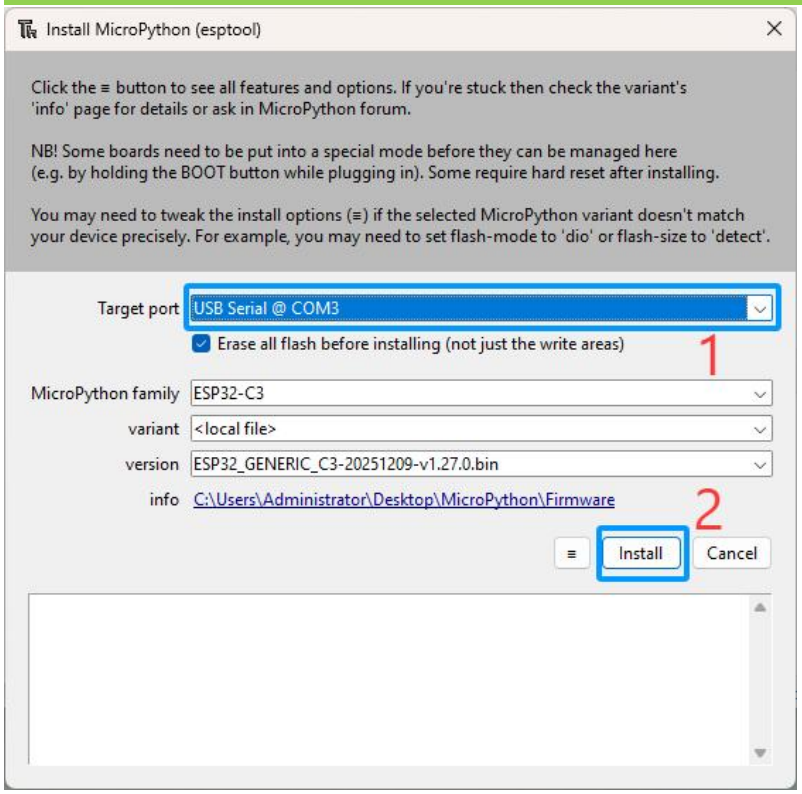
4) The following dialog box pops up. Select “Select local MicroPython image...”.



5) Select the microPython firmware “ESP32_GENERIC_C3-20251209-v1.27.0.bin” provided in the tutorial package. Click "Open".



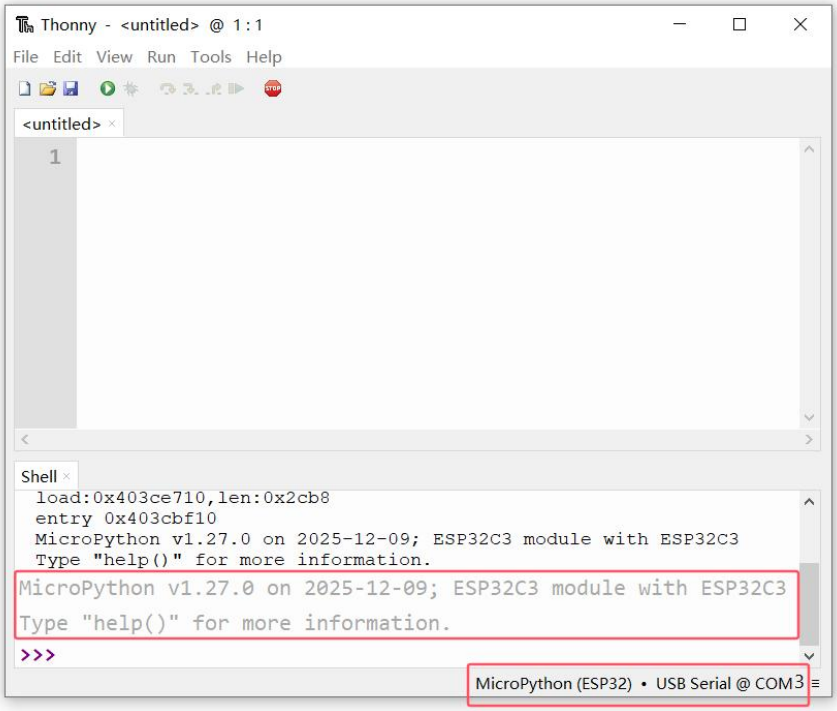
6) Select “USB Serial @ (COM3)”, and then click “Install”.



Wait for the installation to be done, and then click "Close".

7) Testing the Installation

When the installation is completed, you can close the window. You should get the following message on the Shell (see the picture below), and at the bottom right corner, it should have the Interpreter it’ s using and the COM port.



The REPL (Shell)

You should also see the **>>> symbol**
This symbol represents the MicroPython REPL (Read-Eval-Print-Loop) prompt. This symbol means you ' re interacting with MicroPython (the board with MicroPython firmware installed) in interactive mode in real time. This simply means that you can enter and execute commands directly.

Type **print('Hello')** in the Shell after the prompt >>> and press Enter
It should print Hello on the Shell right after you enter the command.

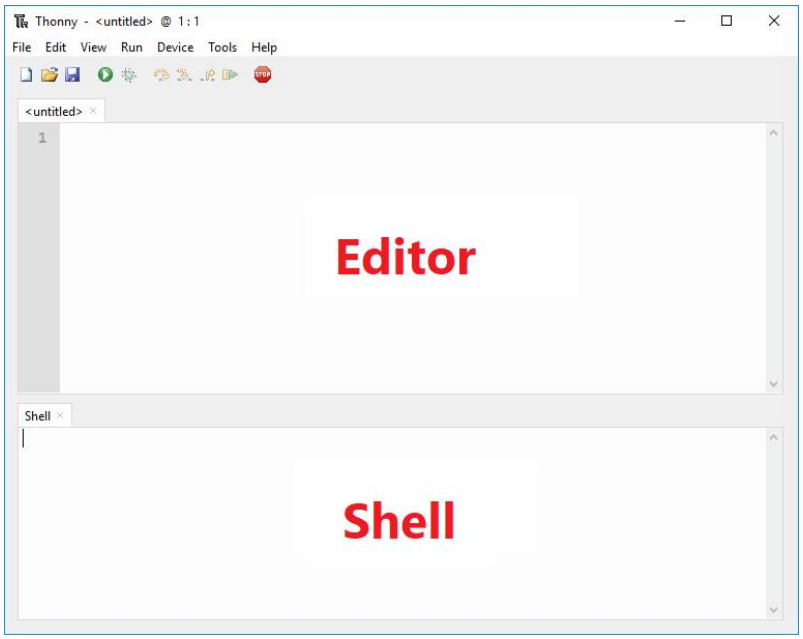
```
>>> print('Hello')
  Hello
```

If everything is working as expected, it means you have successfully installed Thonny IDE, as well as MicroPython firmware on your ESP32 board.

4.6 Thonny IDE Overview

Open Thonny IDE. There are two different sections:

the Editor and the MicroPython Shell/Terminal:



The Editor section is where you write your code and edit your .py files. You can open more than one file, and the Editor will open a new tab for each file.

In the MicroPython Shell, you can type commands to be executed immediately by your ESP32 board without the need to upload new files. The terminal also provides information about the state of an executing program, shows errors related to the uploading process, syntax errors, prints messages, etc…

The Icons

In the Thonny IDE's top section, there are some menu bars that are frequently used during programming:

 : Create a new python file.

 : The open folder icon allows you to open an existing file from your computer. This might be useful if you come back to a program that you worked on previously.

 : The floppy disk icon allows you to save your code. To prevent loss of the code you are writing, it is recommended to develop the habit of frequently clicking this menu.

 : Run the python code of the current page (being viewed) of the **Editor**.

: Terminate the running code.

The following features are currently not supported by the MicroPython firmware for ESP32 C3 and can be understood briefly:

 : The bug icon allows you to debug your code.

 : The arrow tells Python to take a big step, meaning jumping to the next line or block of code.

 : The arrow tells Python to take a small step, meaning diving deep into each component of an expression.

 : The arrow tells Python to exit out of the debugger.

 : Exit the debug mode.

4.7 Running Your First Code - Run current code in Thonny

1.Ensure that you have configured the development environment.

At this point, you should have:

- ✓ Installed a sufficiently charged 18650 battery and turn on the power switch.
- ✓ Connected the ESP32 board to the computer.
- ✓ Ch340 driver installed on your computer
- ✓ Thonny IDE installed on your computer
- ✓ ESP32 board flashed with MicroPython firmware

To get you familiar with the process of writing a file and executing code on your ESP32 board, we'll upload a new script that simply controls the built-in buzzer of the ESP32 to produce sound.

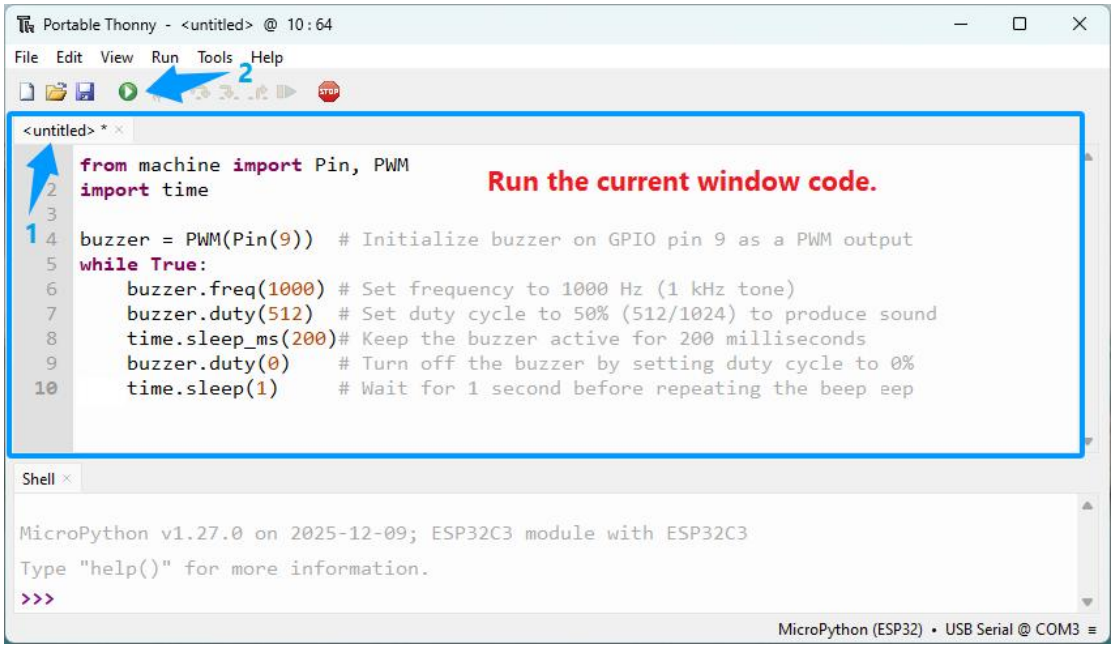
2. Running the Code

When you open Thonny IDE for the first time, the Editor shows an untitled file. Copy the following script to that file:

```
from machine import Pin, PWM
import time

buzzer = PWM(Pin(9))  # Initialize buzzer on GPIO pin 9 as a PWM output
while True:
    buzzer.freq(1000) # Set frequency to 1000 Hz (1 kHz tone)
    buzzer.duty(512)  # Set duty cycle to 50% (512/1024) to produce sound
    time.sleep_ms(200)# Keep the buzzer active for 200 milliseconds
    buzzer.duty(0)    # Turn off the buzzer by setting duty cycle to 0%
    time.sleep(1)     # Wait for 1 second before repeating the beep
```

Copy the code to be Run above into the editor and click the Run icon in Thonny IDE.



Ensure to turn on the power switch of the control board, the buzzer will then emit sound.

To stop the execution of the program, you can hit the STOP button or simply press CTRL+C.

Note:

Just running the file with Thonny doesn't copy it permanently to the ESP32 filesystem. This means that if you unplug it from your computer and apply power to the board, nothing will happen because it doesn't have any MicroPython file saved on its filesystem.

The Thonny IDE Run function is useful to test the code, but if you want to upload it permanently to your board, you need to create and save a file to the board's filesystem. We'll cover this next.

4.8 Uploading Code to the ESP32 Board - Run code in ESP32

Method 1: Save/Upload Directly as main.py

1) After running the buzzer code, stop the execution by clicking the Stop icon. Then, save your file by either clicking the **Save** icon or selecting ***File > Save as...***

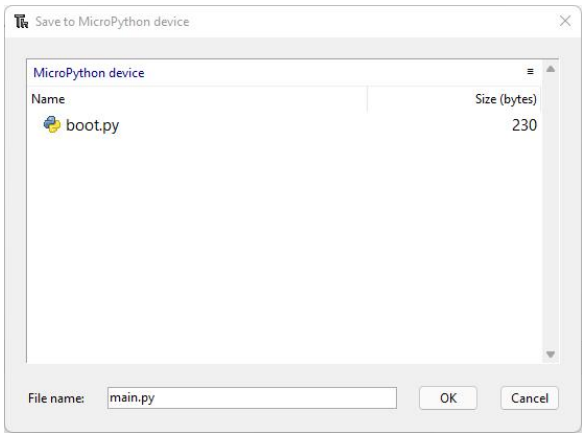
2) Then, select MicroPython device.

Note:

Q：The Thonny file panel doesn't shows "MicroPython Device" ?

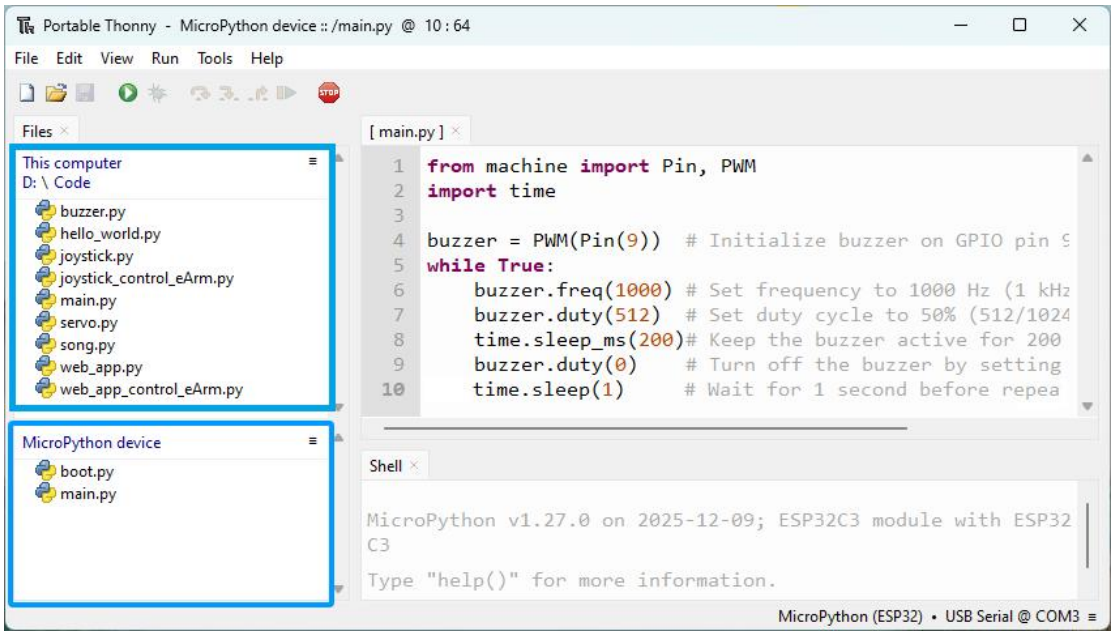
A： Every time the USB port of the computer is connected to the control board, this phenomenon occurs because Thonny does not have the function of automatically detecting the connection of ESP32 devices. Click the **STOP** button or reselect "**MicroPython (ESP32) ▪USB Serial @ COMx**" at the lower right corner of thony to solve this problem.

3) Name the file **main.py**, otherwise, it will not run automatically on the ESP32.



Notice that the window that pops up shows the files currently saved on the board's filesystem. There should be a file called **boot.py.** That file is created by default when you burn MicroPython firmware.

4) You will see two file panels: the a panel labeled "This computer" (your local files) and the panel labeled "MicroPython device" (the connected ESP32's internal storage).



5) Press the on-board REST button so that the board restarts and starts running the code.

6) Your buzzer on the esp32 board should be sound every econds.

After saving the file, you can remove and apply power again to the board. Notice that the buzzer on the esp32 board will automatically emit a sounds when you apply power to it. This means the main.py file was successfully uploaded to the **MicroPython device** and that it is running that file automatically.

Important: when you name a file main.py, the ESP32 will run that file automatically on boot (when it starts). If you call it a different name, it will still be saved on the board filesystem, but it will not run automatically on boot.

Can You Ignore boot.py?

Yes, you can safely ignore the boot.py file. But there is no need to delete it. For almost all beginner and most intermediate ESP32 MicroPython projects (e.g., LED blinking, sensor data reading, basic WiFi connection, simple serial communication). The ESP32's MicroPython runtime operates fully normally, and all your project logic can be directly written in main.py (including simple initialization steps like pin or sensor setup). You only need to use boot.py when you need to separate system-level initialization from main application logic for cleaner code organization.

In short: boot.py prepares the "system environment", main.py runs the "actual project", and REPL is the "debug tool" — the three work together to form the complete running logic of ESP32 MicroPython, with boot.py being an optional but powerful supplement for system-level configuration.

Method 2: main.py calls other files

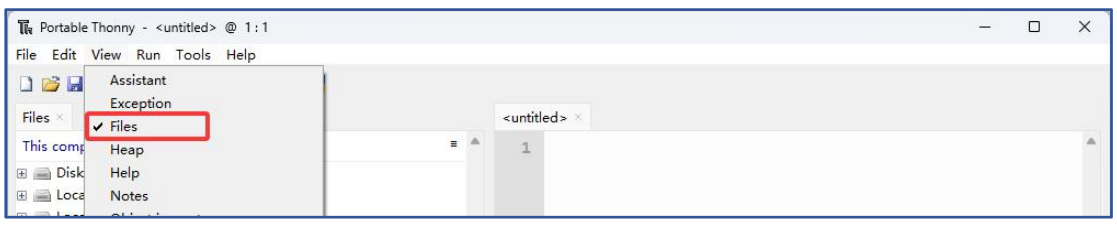
Comparison of the Two Methods

Method	Operation Process	Advantages	Disadvantages	Applicable Scenarios
① Save as main.py (direct upload)	1. Open the tutorial code	All the code is written in the main.py file, which is suitable for simple single-file projects.	1. High coupling, inconvenient for debugging	Single-file experiments, one-time testing
	2. Click ***Save as*** → Select device		2. Requires re-uploading entire code for modifications	
	3. Name it main.py		3. Not modular	
② main.py calls other files	1. Upload functional code (e.g., buzzer.py) to the device	1. Modular, easy to maintain	Requires slightly more file management steps	Multi-module projects, long-term development
	2. Create a new main.py file and write the code to call other files in main.py.	2. Allows independent debugging of submodules		
	3. Upload main.py to the ESP32 device.	3. Supports code reuse		

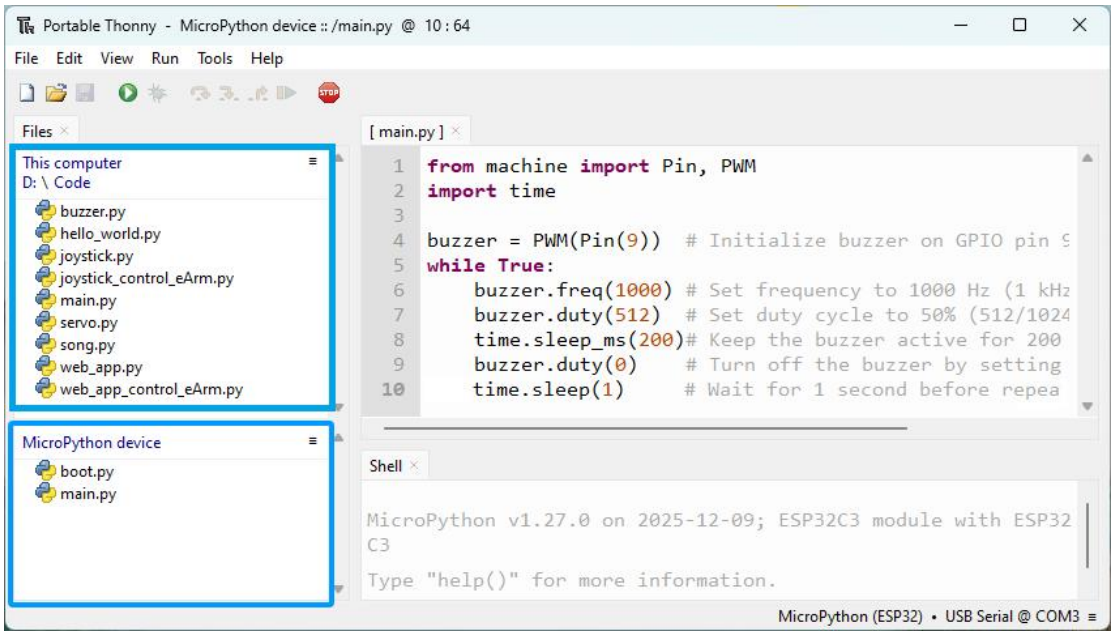
It is recommended to choose the **second** method as it is more professional, flexible and in line with the recommended practices of embedded development.

We provide the **xxx.py** files and **main.py** with different functions for your use in the tutorial package. The code in main.py will call any the **xxx.py** files under "**MicroPython device**" except main.py and boot.py.

1) Click "View" in the menu bar and select "Files" from the dropdown.



2) You will see two file panels: the a panel labeled "This computer" (your local files) and the panel labeled "MicroPython device" (the connected ESP32's internal storage).

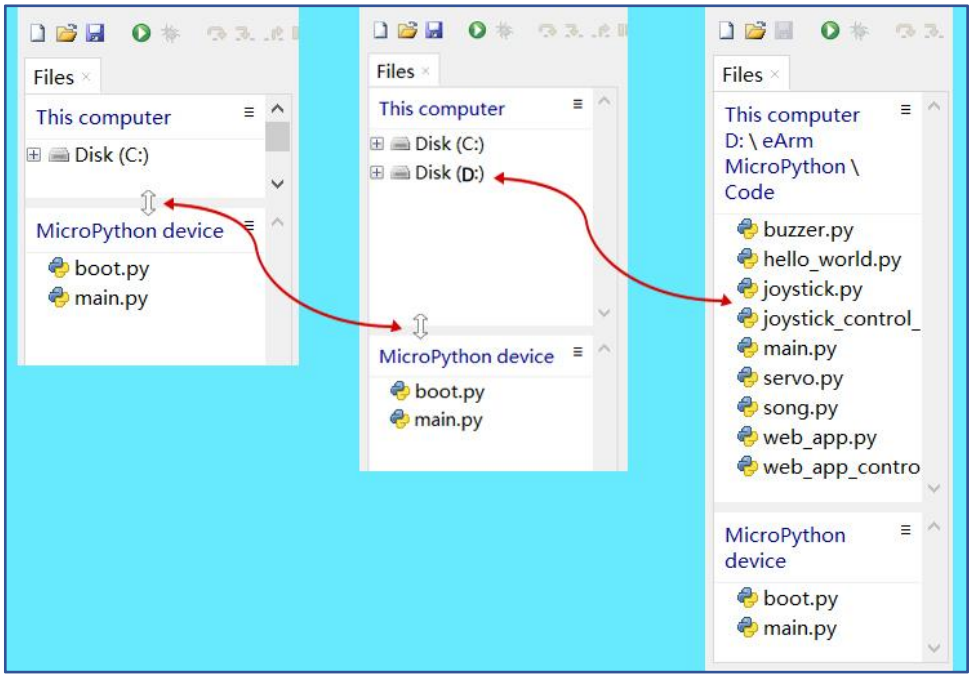


Note:

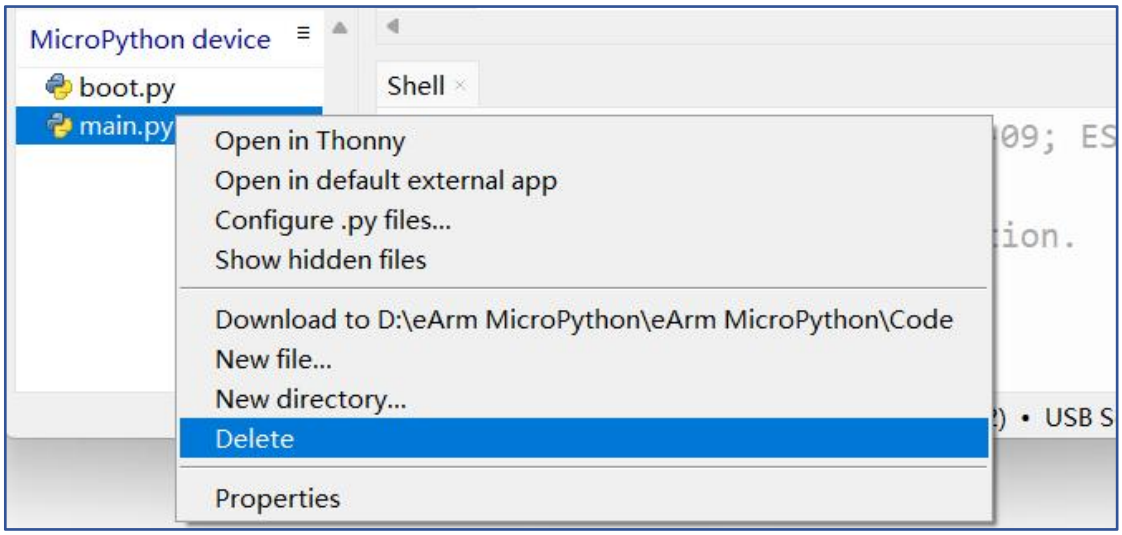
Q：The Thonny file panel doesn't shows "MicroPython Device" ?

A：Every time the USB port of the computer is connected to the control board, this phenomenon occurs because Thonny does not have the function of automatically detecting the connection of ESP32 devices. Click the **STOP** button or reselect "**MicroPython (ESP32) •USB Serial @ COMx**" at the lower right corner of thony to solve this problem.

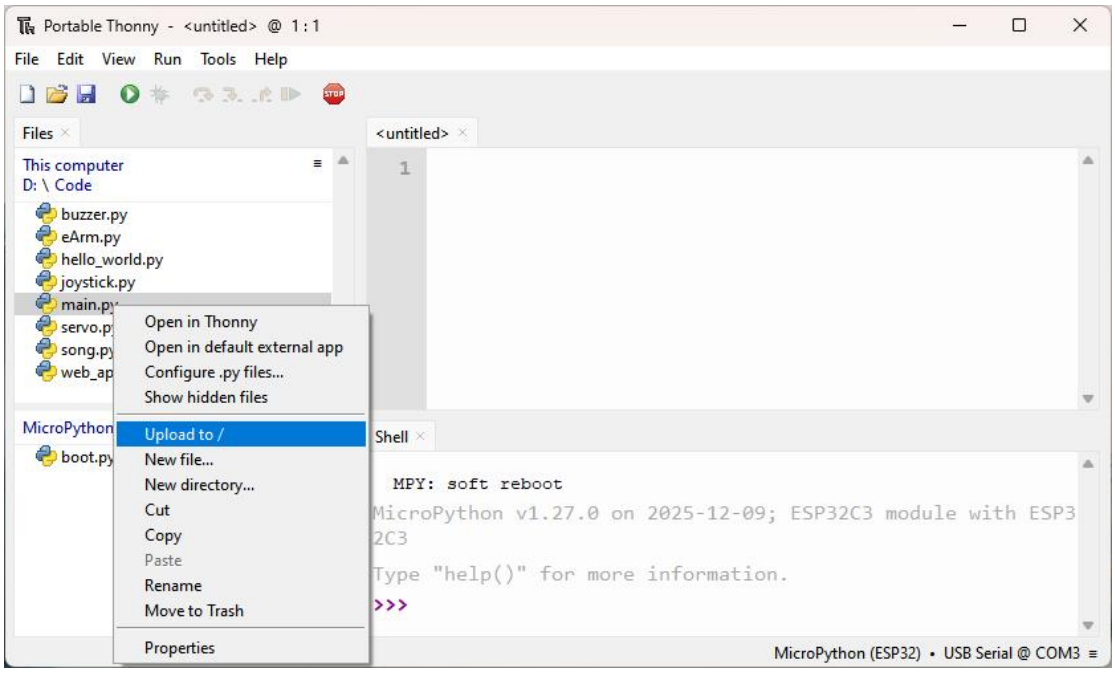
3) Scroll down the navigation bar to expand the folder directory view. Navigate to the directory where the downloaded tutorial package code is stored.



4) Right-click “main.py” previously uploaded to make the buzzer sound in “MicroPython device”, Delete it from ESP32-C3’s root directory.

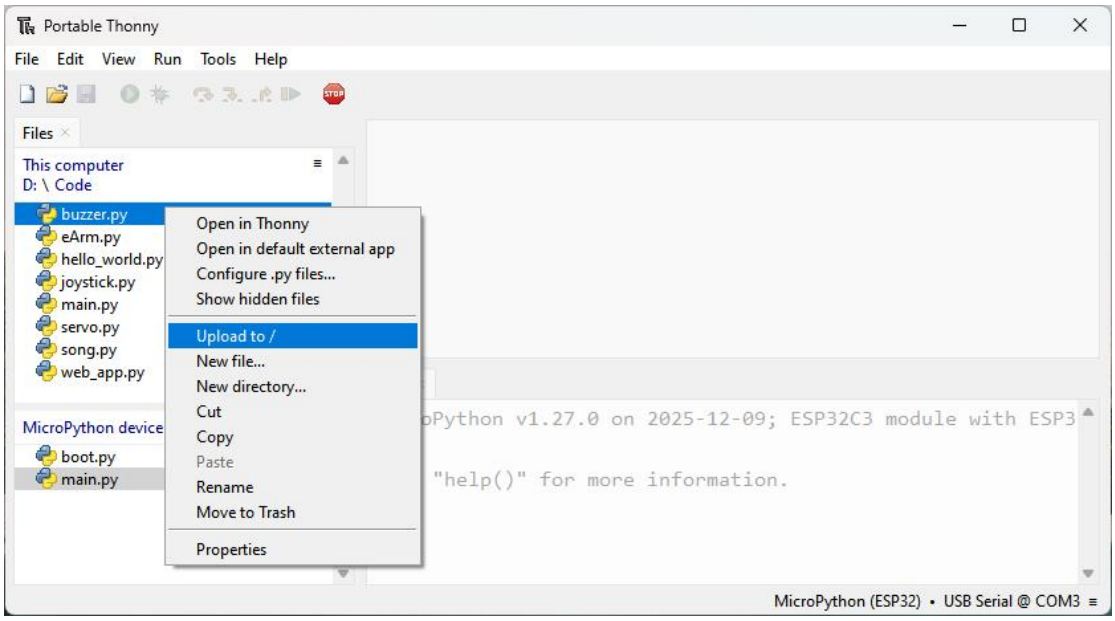


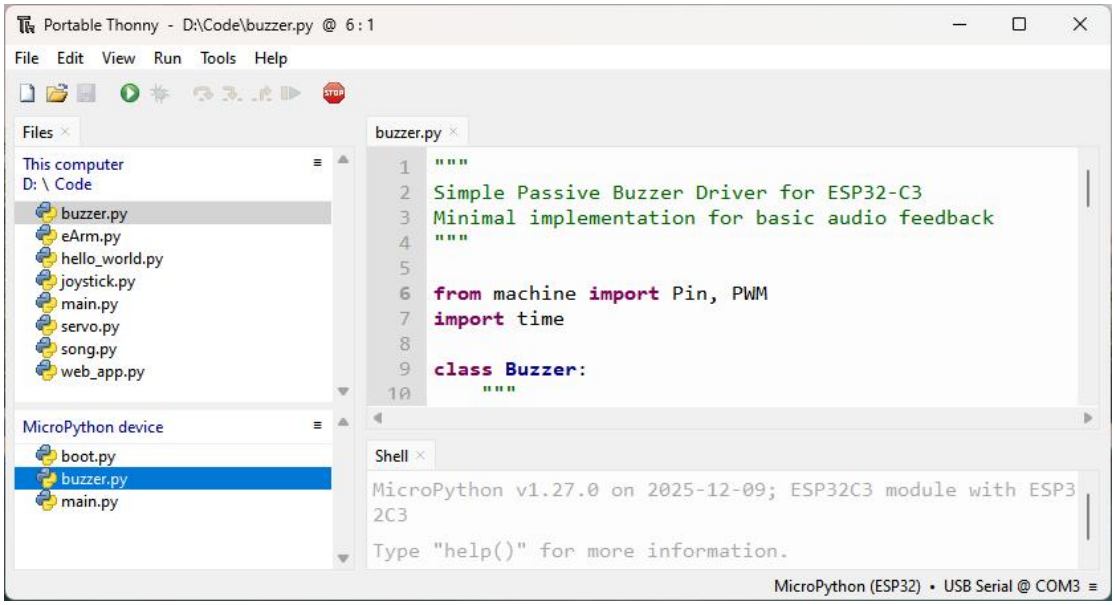
5) Right-click “main.py” in your local files, and click "Upload to/"



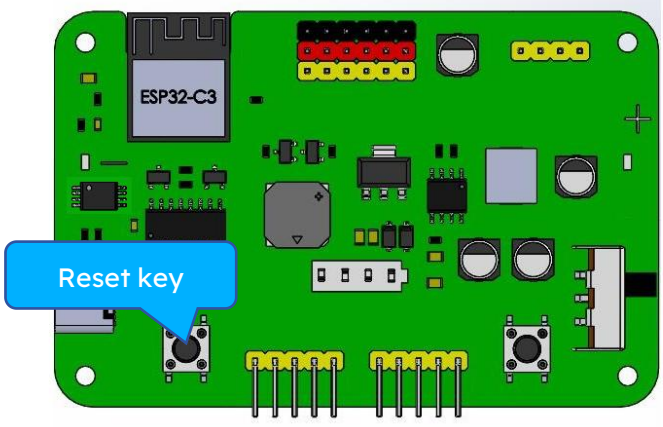
Important: You must stop running online before uploading files to "MicroPython device"! (Click the "**STOP**" menu in Thonny)

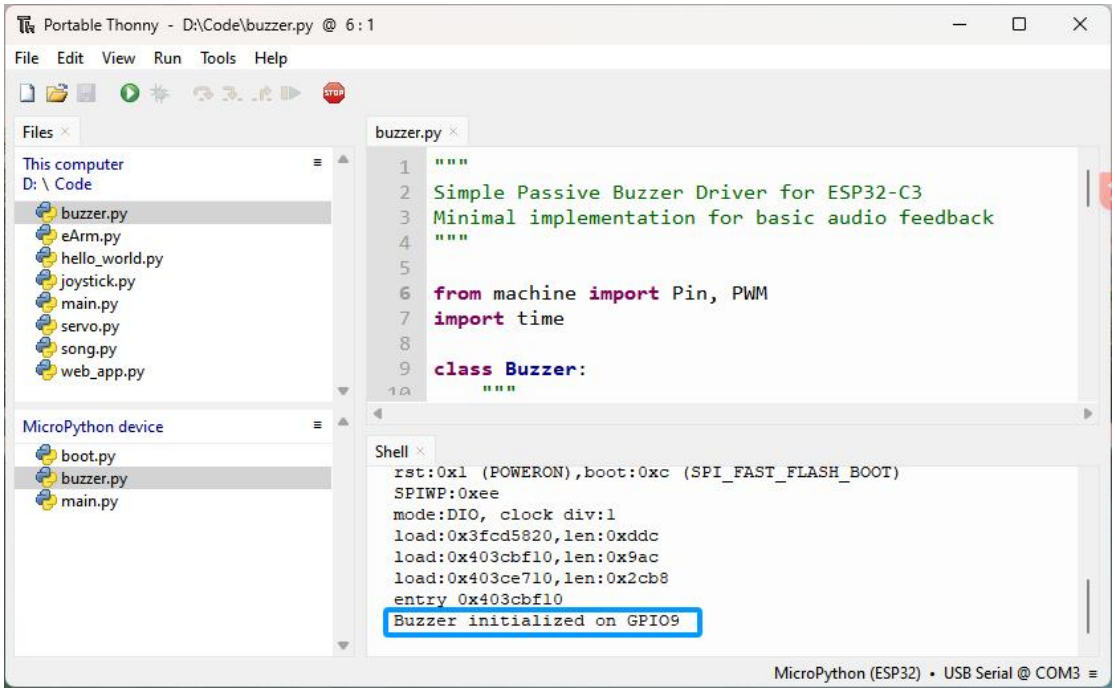
6) Right-click “buzzer.py” in your local files, and click "Upload to/"



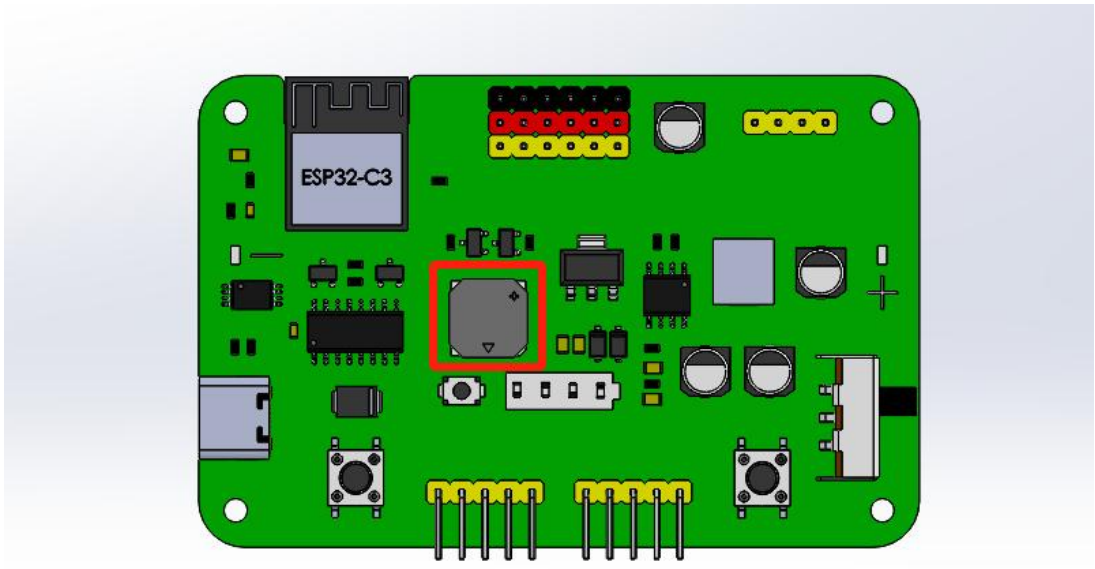


7) Press the reset key of the esp32 board, you will see the code is executed.





8) The buzzer on the ESP32-C3 board will keep making sounds when turn on the power switch.

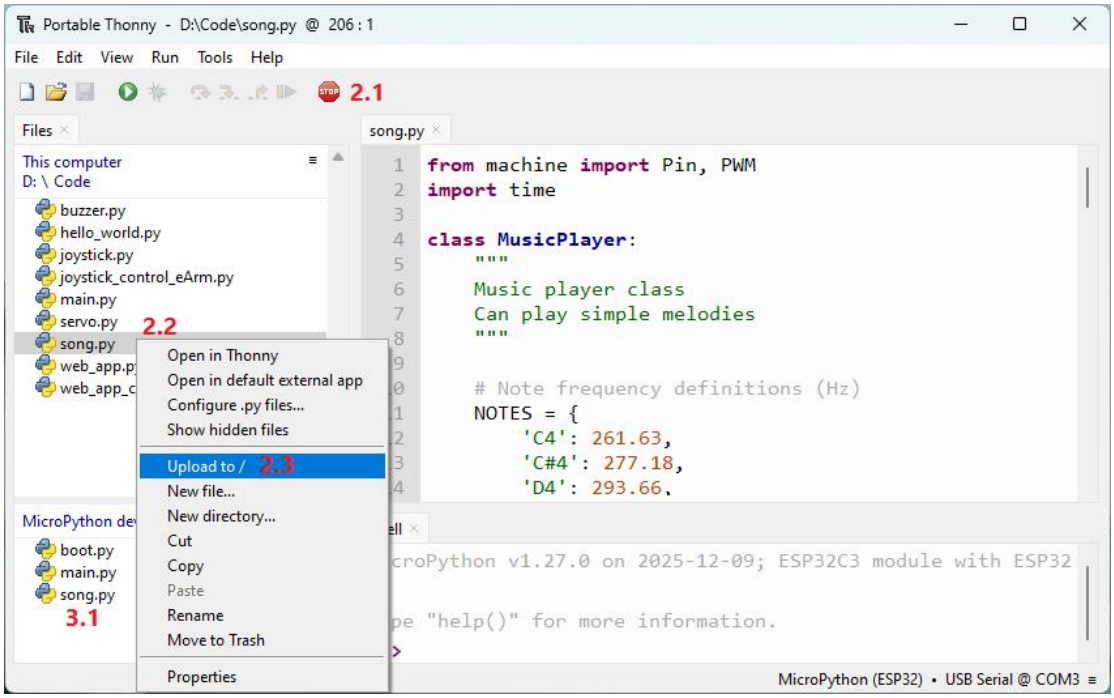


Important: The shared main.py file we provided is designed to call only one of the non - main.py code files at a time. If you upload multiple such files together, pressing reset may cause the system to behave erratically — it might randomly execute a different file instead of the intended one.

4.9 Code Examples

4.9.1 Play Songs

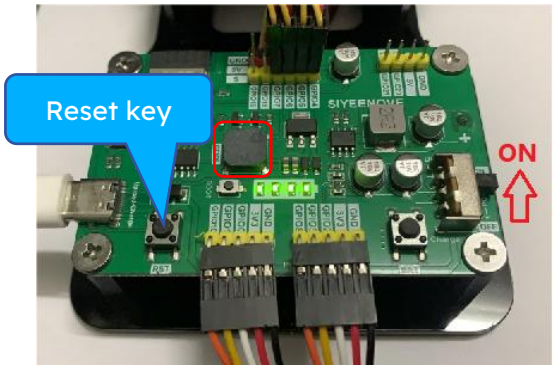
1	Delete the previously uploaded **buzzer.py** from the MicroPython device, because the provided main.py is designed to call only one module file.
2	If any code is running, please click the **STOP (2.1)** button.Then right-click on **song.py (2.2)** and select "**Upload to /**"**(2.3)**.
3	The **song.py (3.1)**file will appear in the MicroPython device file list.
4	Ensure the control board's power switch is turned **ON** and the battery has sufficient charge.
5	Press the **Reset** button on the esp32 control board to restart it. The board's buzzer will then start playing music.
6	To **view** the code: Navigate to the **song.py**, then double-click the file to open it in Thonny.



Note:

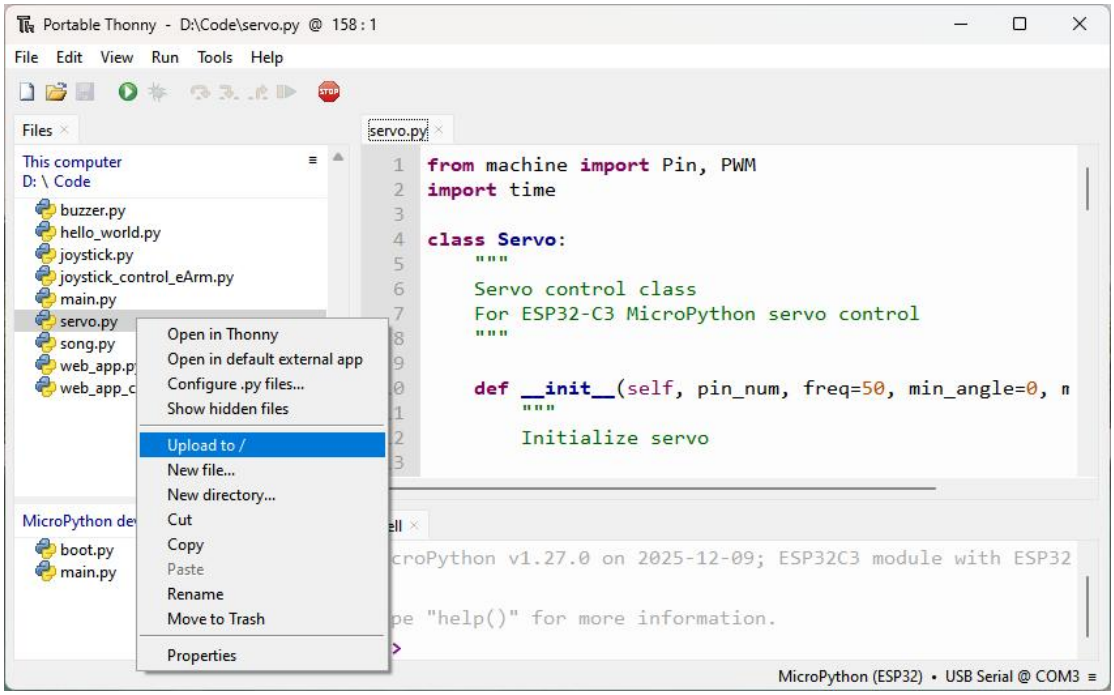
Q：The Thonny file panel doesn't shows "MicroPython Device" ?

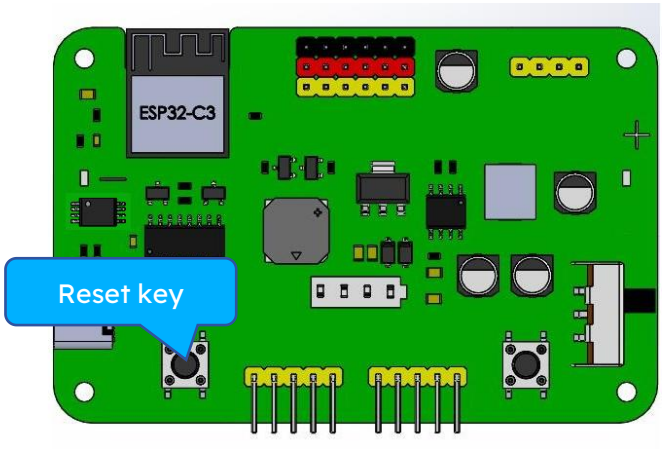
A：Every time the USB port of the computer is connected to the control board, this phenomenon occurs because Thonny does not have the function of automatically detecting the connection of ESP32 devices. Click the **STOP** button or reselect "**MicroPython (ESP32) ▪USB Serial @ COMx**" at the lower right corner of thony to solve this problem.



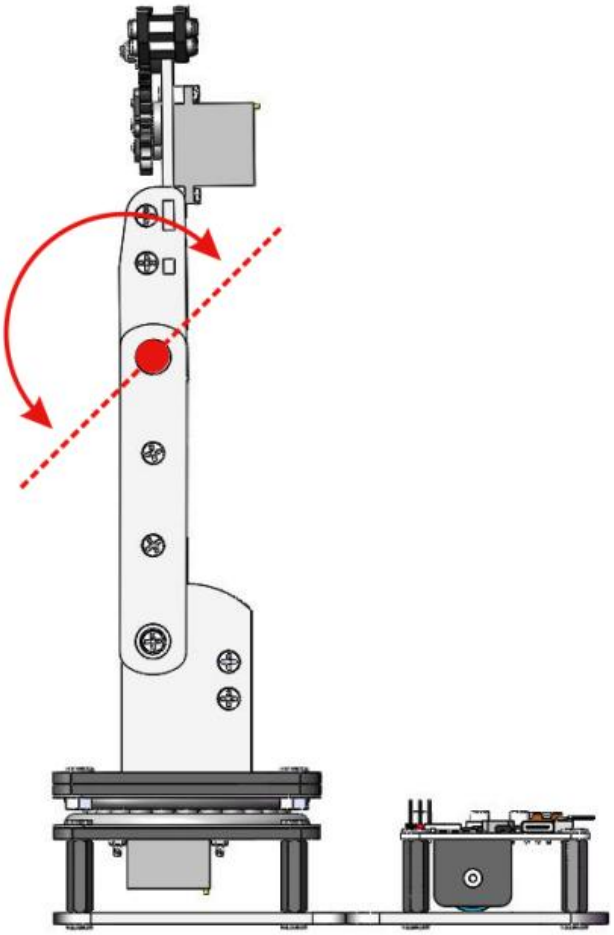
4.9.2 To Drive a Servo

1	Delete the previously uploaded **song.py** from the MicroPython device, because the provided main.py is designed to call only one module file.
2	If any code is running, please click the **Stop** button.Then right-click on **servo.py** and select "**Upload to /**"
3	The **servo.py** file will appear in the MicroPython device file list.
4	Ensure the control board's power switch is turned ON and the battery has sufficient charge.
5	Press the **Reset** button on the esp32 control board to restart it. The eArm's servo C will swing 0-180, 180-0 in a cycle:
6	To **view** the code: Navigate to the **servo.py**, then double-click the file to open it in Thonny.



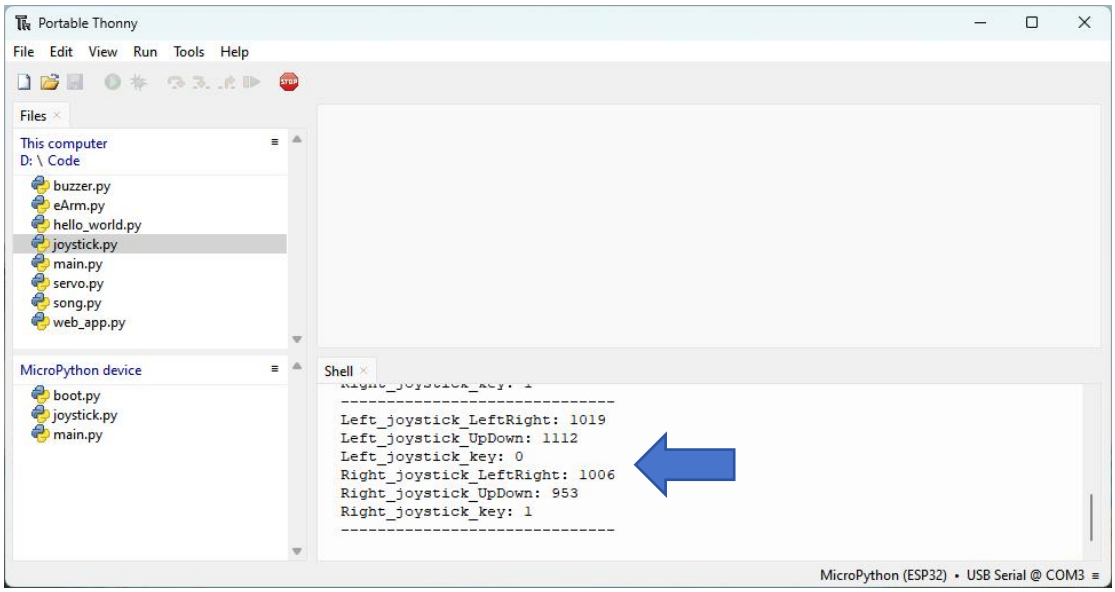


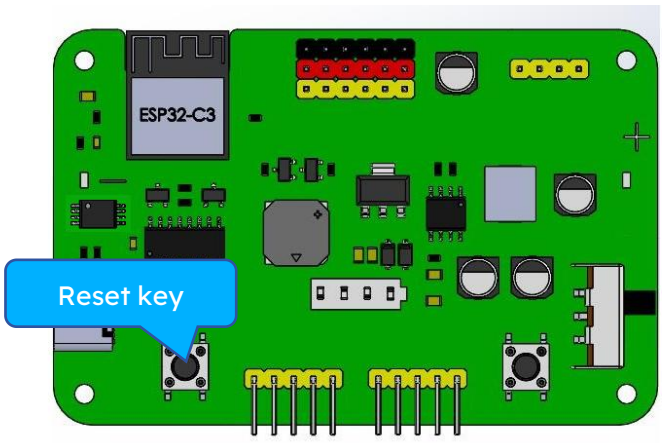
After the code is uploaded successfully, the eArm's servo C will swing 0-180, 180-0 in a cycle:



4.9.3 Read the Value of the Joystick

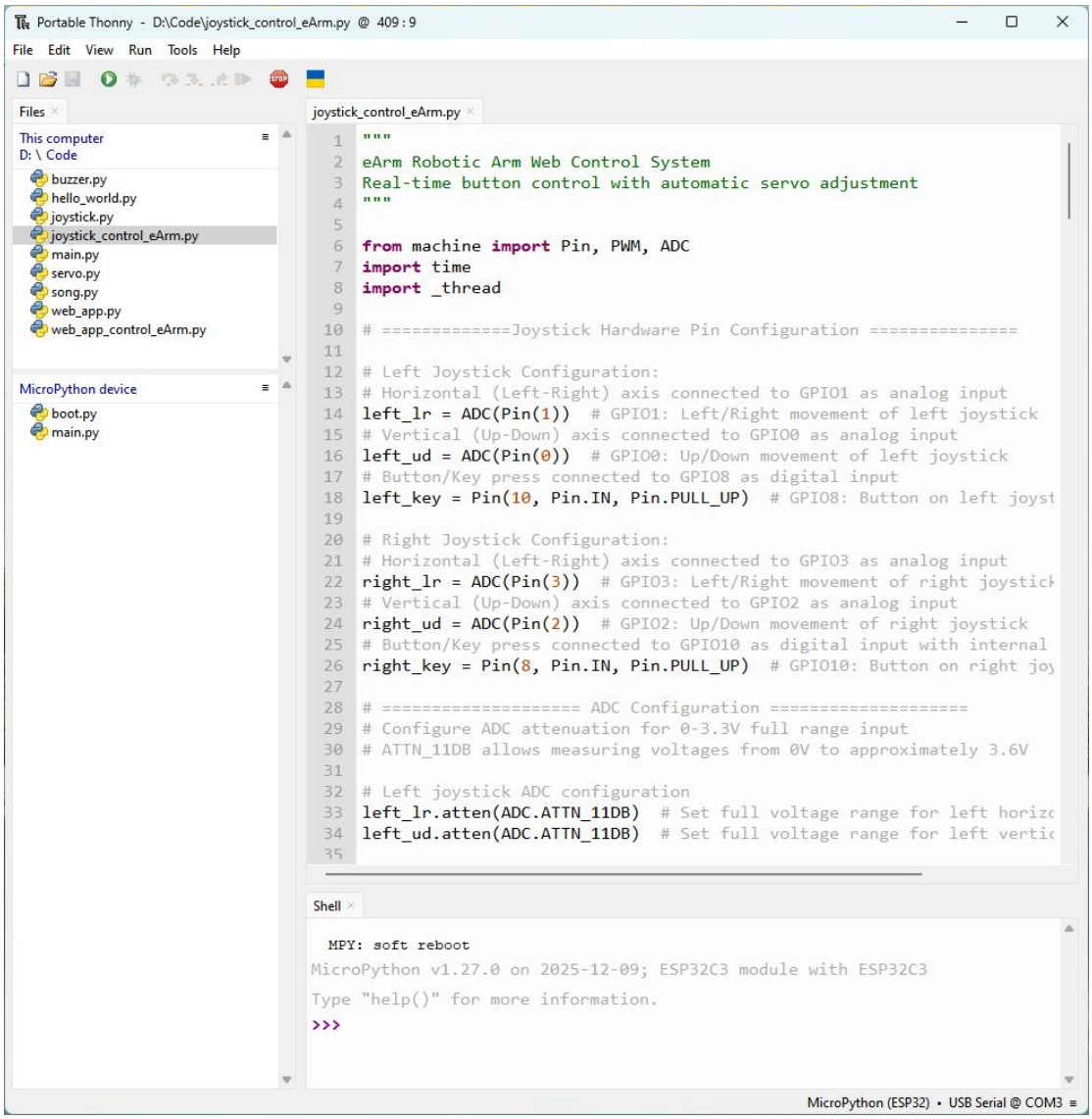
1	Delete the previously uploaded **servo.py** from the MicroPython device, because the provided main.py is designed to call only one module file.
2	If any code is running, please click the **Stop** button.Then right-click on **joystick.py** and select "**Upload to** /"
3	The **joystick.py** file will appear in the MicroPython device file list.
4	Ensure the control board's power switch is turned ON and the battery has sufficient charge.
5	Press the **Reset** button on the esp32 control board to restart it. Push the joystick left, right, up, down, and the shell will print the joystick value
6	To **view** the code: Navigate to the **joystick.py**, then double-click the file to open it in Thonny.



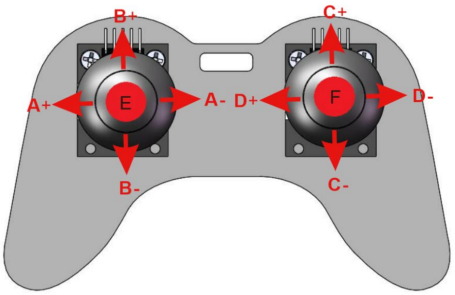


4.9.4 Joystick Control eArm

1	Delete the previously uploaded **joystick.py** from the MicroPython device, because the provided main.py is designed to call only one module file.
2	If any code is running, please click the **Stop** button.Then right-click on **joystick_control_eArm.py** and select "**Upload to** /"
3	The **joystick_control_eArm.py** file will appear in the MicroPython device file list.
4	Ensure the control board's power switch is turned ON and the battery has sufficient charge.
5	Press the Reset button on the esp32 control board to restart it. You can push or press the joystick to control the robot arm
6	To **view** the code: Navigate to the **joystick_control_eArm.py**, then double-click the file to open it in Thonny.



Push the handle left, right, up, down, or press the joystick down:



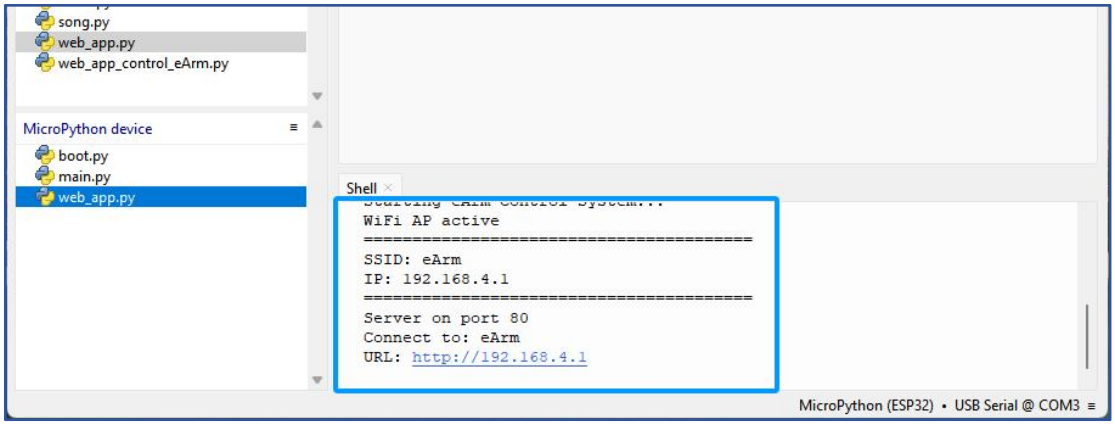
A+: Rotate arm left (Servo A)
A-: Rotate arm right (Servo A)
B-: Raise rear arm (Servo B)
B+: Lower rear arm (Servo B)
C-: Raise the forearm (Servo C)
C+: Lower the forearm (Servo C)
D+: Open the gripper (Servo D)
D-: Close the gripper (Servo D)

F: Reserve
E: Enable/disable the buzzer

4.9.5 Read the Value of the Web APP

1	Delete the previously uploaded **joystick_control_eArm.py** from the MicroPython device, because the provided main.py is designed to call only one module file.
2	If any code is running, please click the **Stop** button.Then right-click on **web_app.py** and select "**Upload to /**"
3	The **web_app.py** file will appear in the MicroPython device file list.
4	Ensure the control board's power switch is turned ON and the battery has sufficient charge.
5	Press the Reset button on the esp32 control board to restart it. You can see the network service startup information of the esp32 in the Shell. Once connected to the ESP32's WiFi, open the web app in a browser. Each button click logs its value in the Shell.
6	To **view** the code: Navigate to the **web_app.py**, then double-click the file to open it in Thonny.

After uploading the code, the shell will print the wifi information.

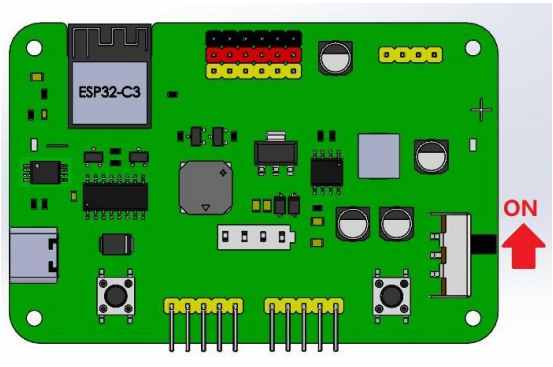


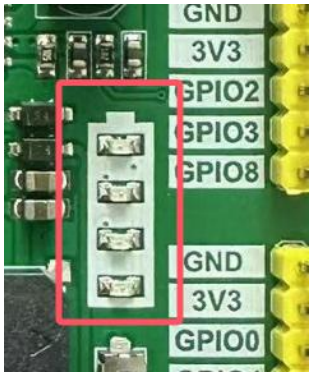
To connect to WiFi, please follow the steps below.

1) Turn the eArm's power switch to the ON position.

■: Make sure the 18650 lithium battery is installed.

■: The battery indicator shows 3 bars or more.





2) Turn on the phone's Wi-Fi and connect to the network named 'eArm'

■: Once connected to the Wi-Fi, you may see a 'Network connection failed' message; just disregard it.

■: Turn off mobile data in your phone settings

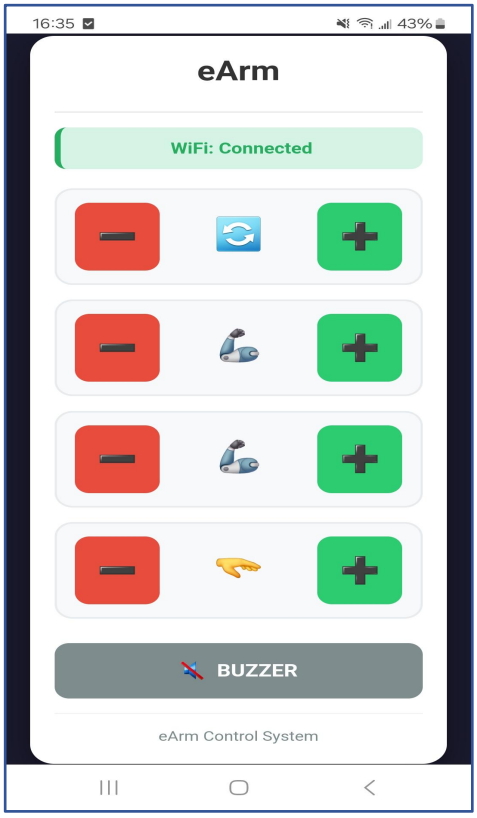
■: This ensures stable connection to the Web App

3) Open the web browser on your phone

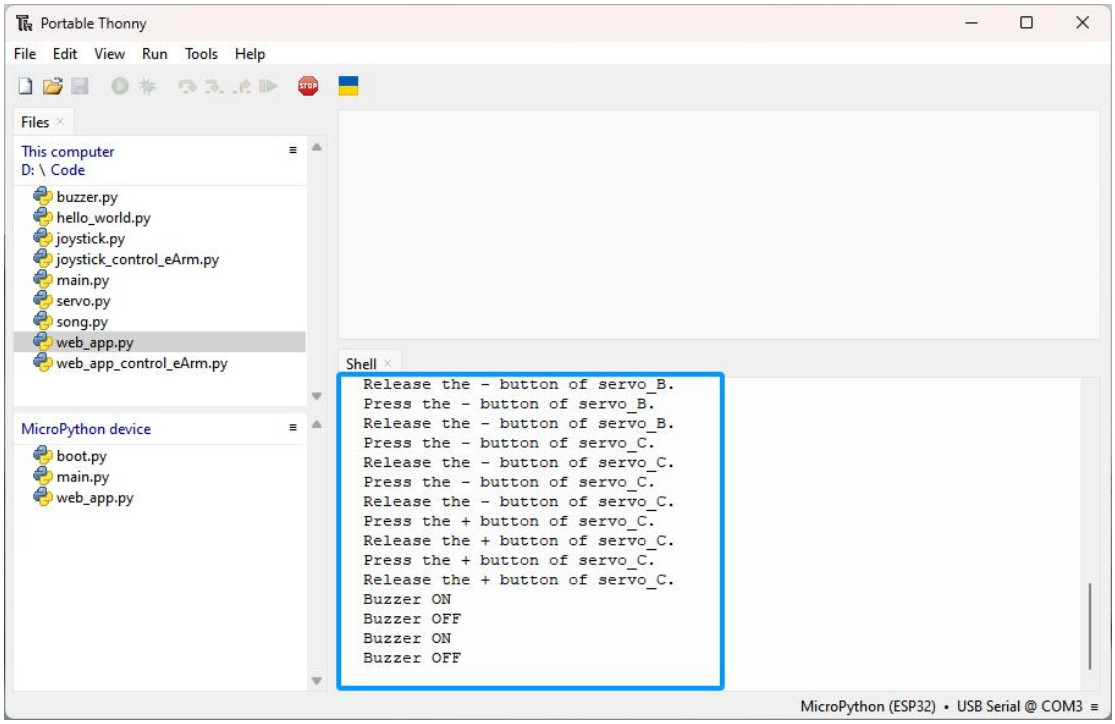
4) Type 192.168.4.1 into the address bar

5) Press Enter to connect

After wifi successful connection, the app control interface will appear in your browser.

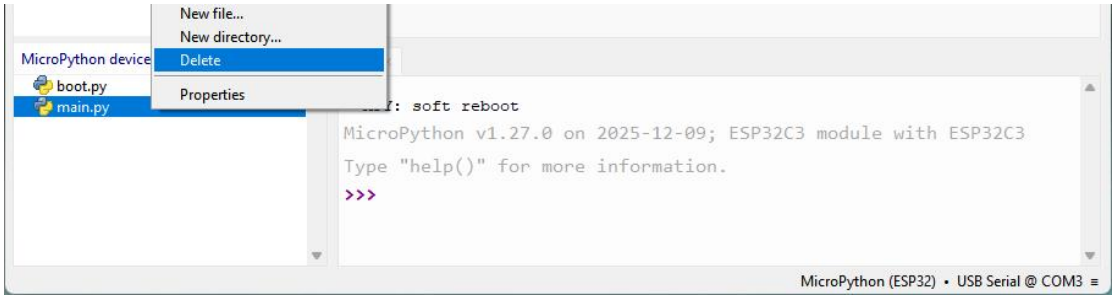


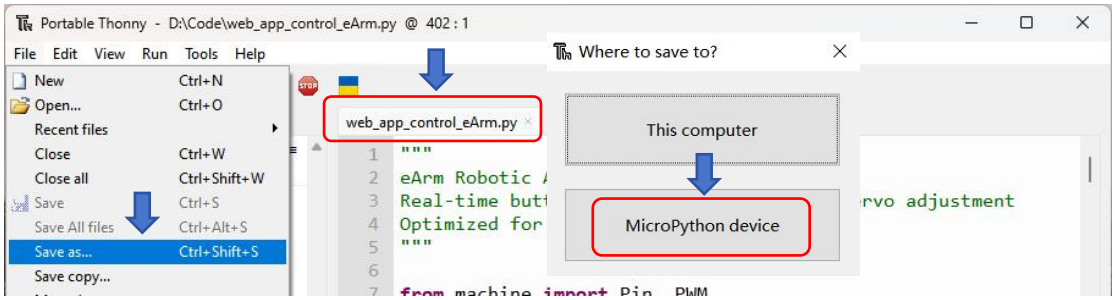
Click the button on the web app and the shell will print the key information.

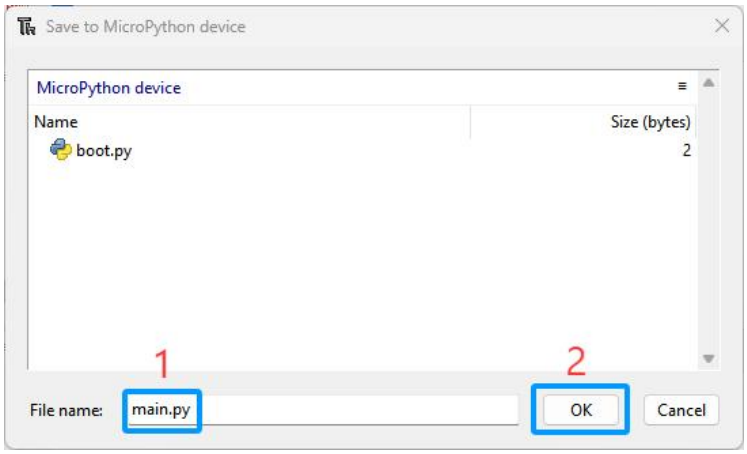


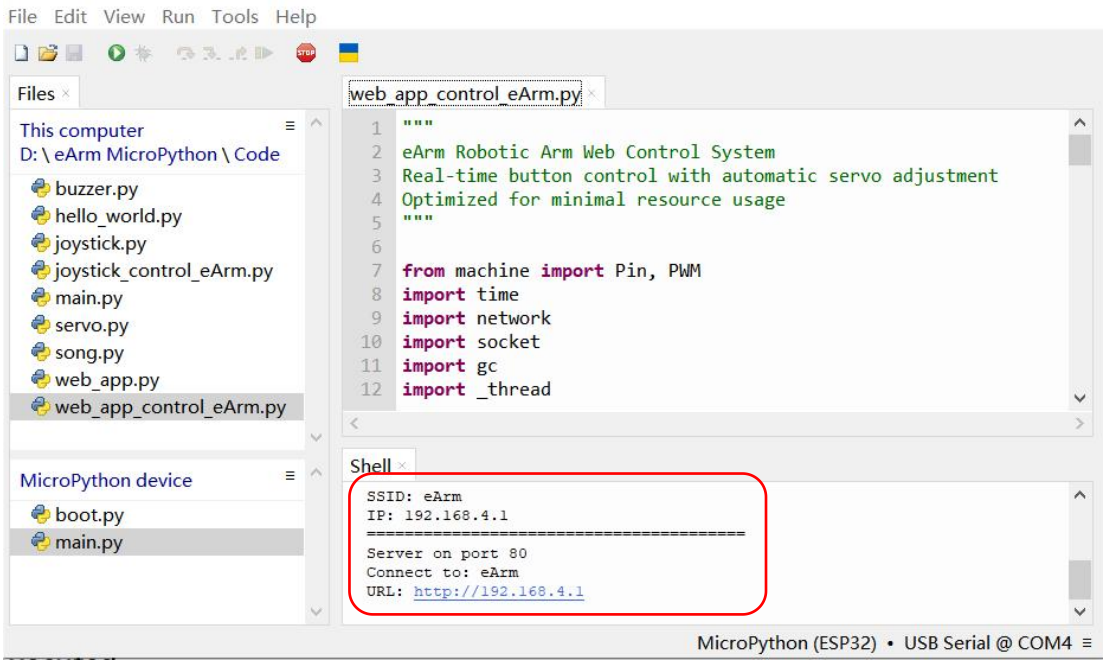
4.9.6 Web App-Controlled eArm Robot

1. Delete the previously uploaded **web_app.py** and **main.py** from the MicroPython device.
2. In this final lesson, directly **save** the code directly **as main.py** and upload to the MicroPython device. Avoid memory overload and ensure smooth operation.
3. Navigate to the **web_app_control_eArm.py**, then double-click the file to open it in Thonny.
4. Click ***File > Save as...*** and select MicroPython device. Name the file **main.py** and click Ok to proceed.
5. Ensure the control board's power switch is turned ON and the battery has sufficient charge.
6. **Press** the Reset button on the esp32 control board to restart it. You can see the network service startup information of the esp32 in the Shell. Once connected to the ESP32's WiFi, you can open the web app in a browser to control the robot arm

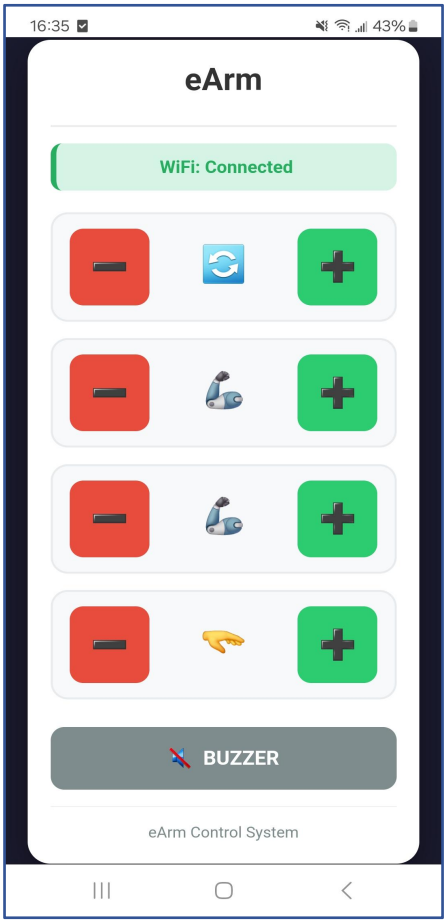


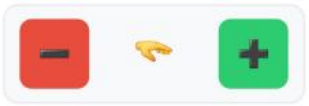






Connect to the ESP32 Wi-Fi using the method from the previous lesson, or click here. Once your device connected to the ESP32's WiFi, you can open the web app in a browser to control the robot arm.



		Robot arm turns left
		Robot arm turns right
		Rear arm raises
		Rear arm lowers
		Front arm raises
		Front arm lowers
		Gripper opens
		Gripper closes
		Buzzer ON
		Buzzer OFF

Important! Critical Gripper Maintenance (Servo Protection)
Continuously gripping an object under load places significant stress on the servo motor (particularly the D-axis driving the gripper), causing it to generate heat. Therefore, we strongly recommend limiting continuous load-bearing gripping to no more than 40 seconds. After completing a gripping operation, release the object immediately by opening the gripper to allow the motor to cool down and rest. Exceeding this duration is the primary cause of motor overheating and burnout.
Our latest optimized firmware/code includes a protective feature. If the gripper servo does not receive a new command within 40 seconds, it will automatically disengage (cease PWM signal transmission) to prevent overheating.
Best Practice: Actively control the gripper. When not manipulating objects, open the gripper or send commands to maintain it in a relaxed state.

5

Factory Reset Function

5.1 Firmware

Providing the device's "factory reset program" directly to users in the form of a firmware file (e.g., .bin or .hex), allowing them to flash it into the device without setting up an Arduino IDE or Thonny IDE, thereby completing the restoration process.

1. Issues with the Traditional Approach

Typically, restoring a device via Arduino requires users to:

- Install the Arduino IDE or related development tools.
- Download the source code (.ino file), possibly requiring additional library setups and dependencies.
- Compile and upload it to the device.

This process has a high barrier for non-technical users and may fail due to environment-related issues.

2. Advantages of Providing Firmware Directly

Simplified User Process:

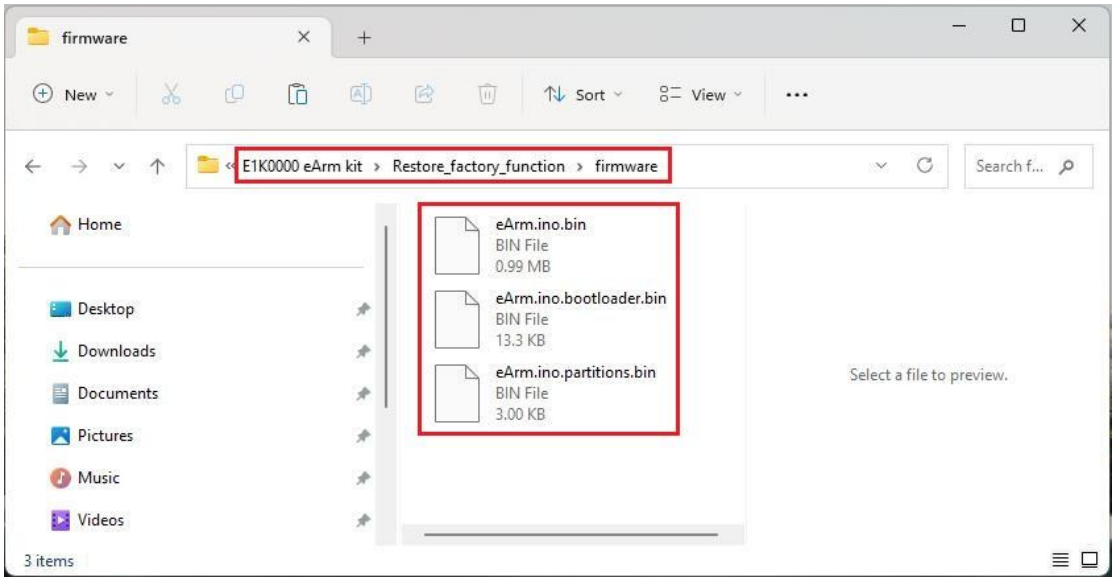
- Provide a pre-compiled firmware file (e.g., firmware.bin), allowing users to flash it with a simple tool (e.g., a serial flasher) in one step.
- No need to install Arduino IDE or Thonny IDE resolve compilation errors, or manage library conflicts.

Use Cases:

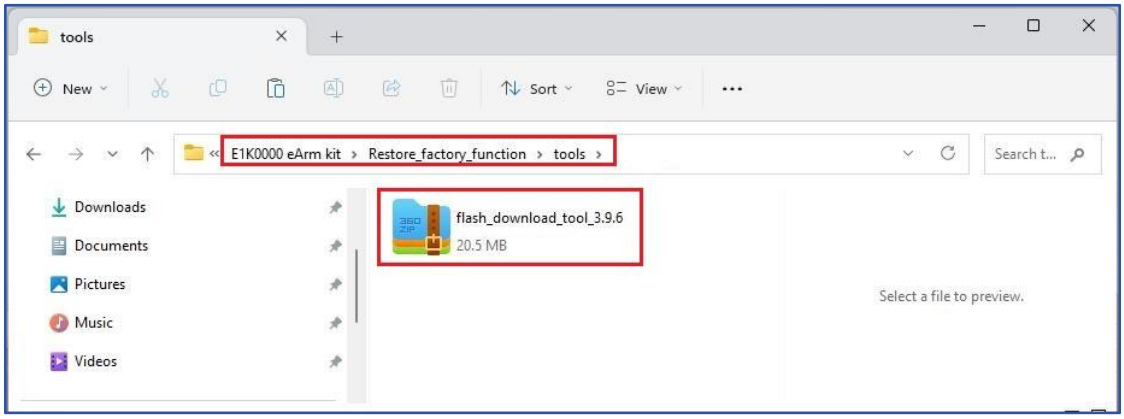
- Restoring a malfunctioning device to factory settings.
- Mass-flashing the same firmware to multiple devices (improves efficiency).

3. How to Implement This?

We provide factory firmware files for this product, which are saved in the following folders:

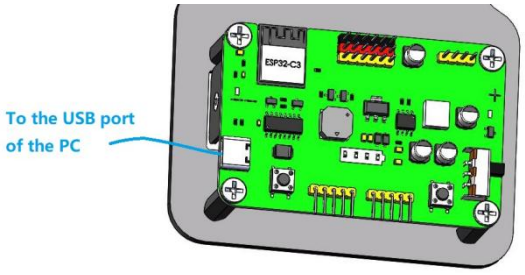
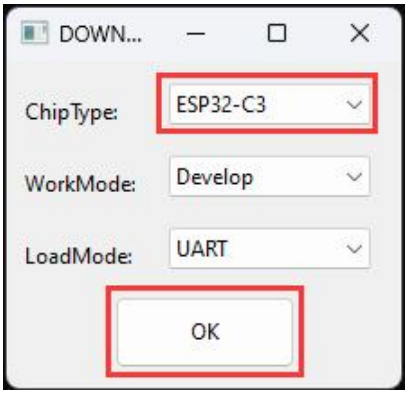


5.2 Burning Tool

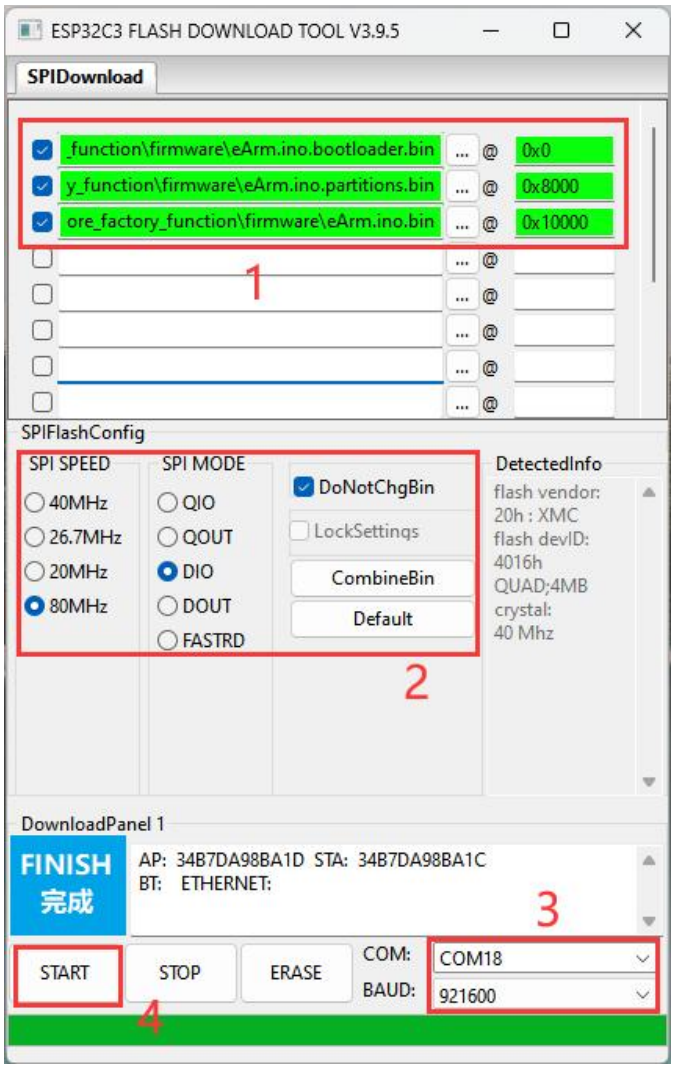


You can also download the latest version of the burning tool from the official website:
https://www.espressif.com.cn/zh-hans/support/download/other-tools

5.3 How to Burn Firmware

Use a USB cable to connect the PC and eArm, as shown below:	Start the burning tool:
	

Select the firmware we provided and fill in the burning address correctly:



You must have installed the CH340 driver and turn on the switch of the ESP32 board, otherwise the COM port will not be found!

6

Common Issues & Solutions (Windows)

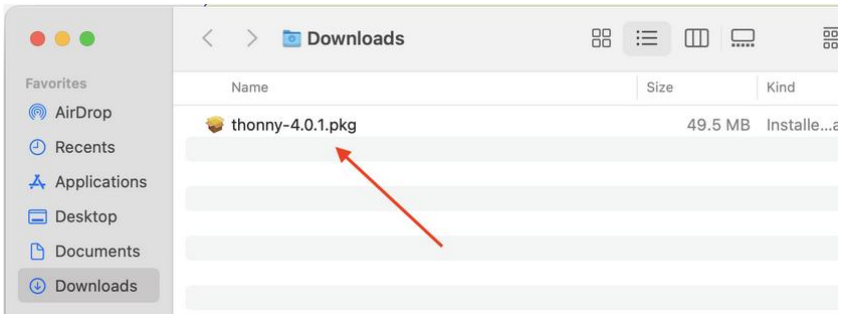
Common Issues & Solutions for ESP32-C3 Programming with Thonny IDE (Windows Version) - Beginner's QA

This QA compiles the **most frequent errors, unexpected behaviors, and operation problems** beginners encounter when programming ESP32-C3 with Thonny IDE on Windows. Each section includes **error symptoms, root causes, step-by-step solutions**, and practical error cases/screenshots description (for easy identification). All operations are beginner-friendly and aligned with MicroPython development workflows.

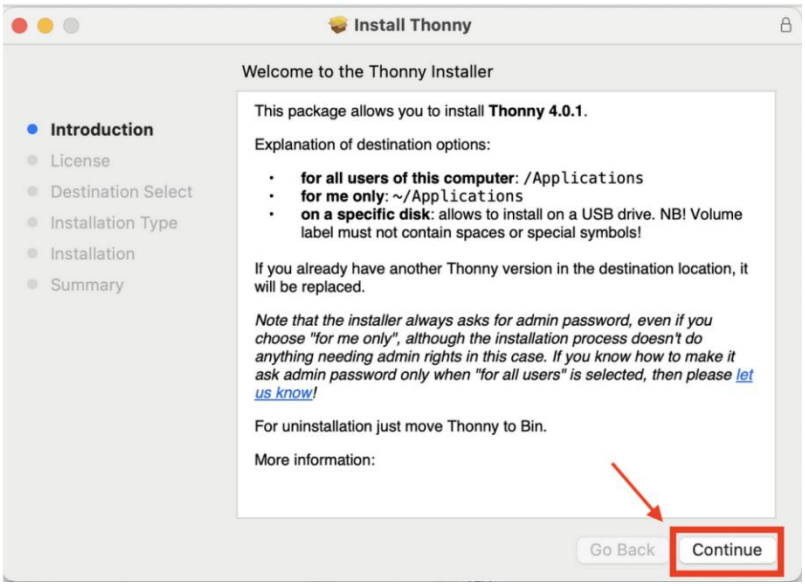
6.1 How to Install Thonny IDE on MacOS?

1) Go to the Thonny office website: https://thonny.org/
2) Click to download the **thonny-4.1.7.pkg (42 MB)** installer.

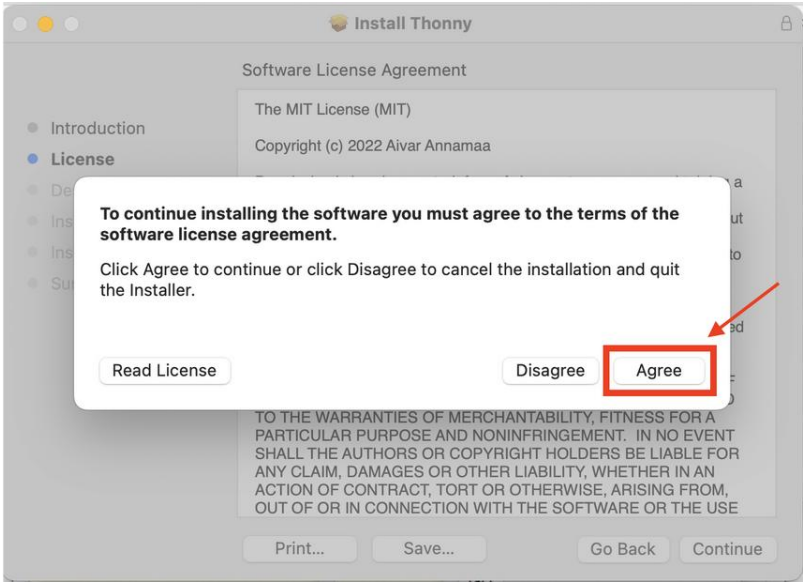
3) Open and Run the Installer and click "Allow"



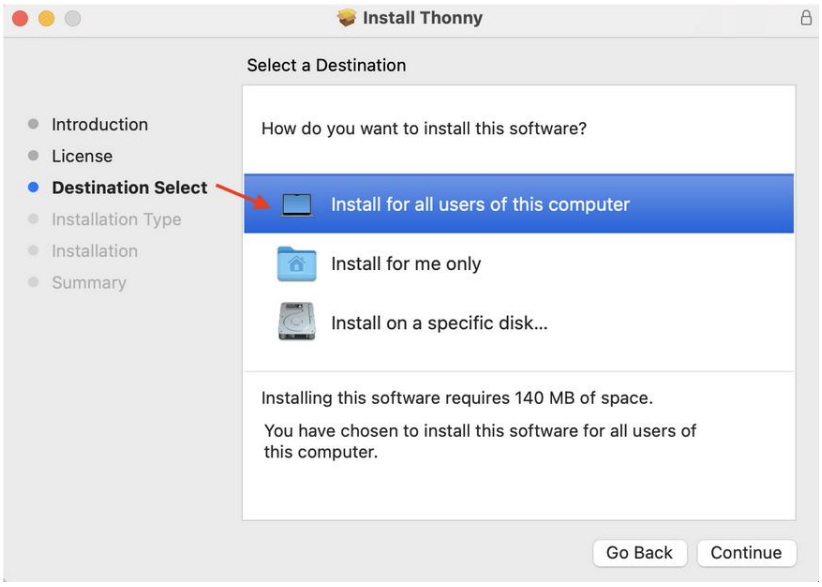
4) Click "Continue" to Confirm the Installation



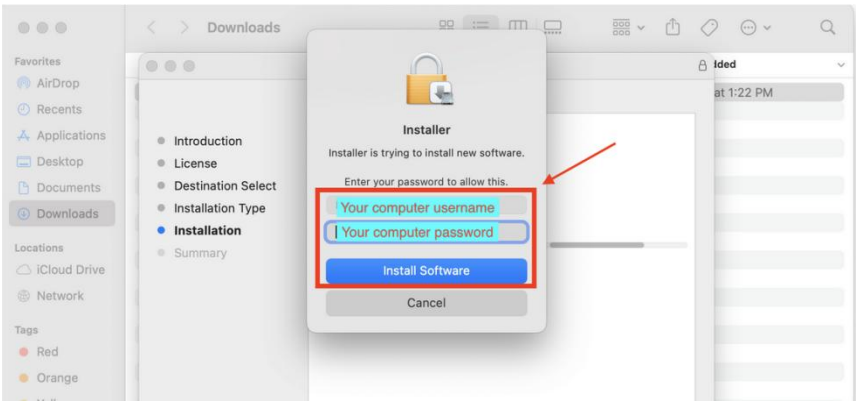
5) Keep clicking “Continue” and click “Agree” to the license agreements



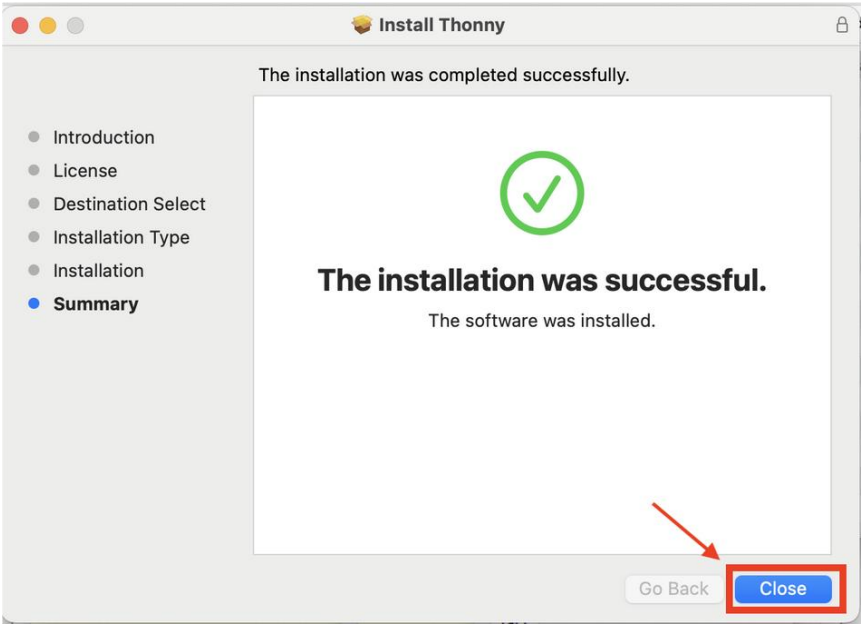
6) “Select a destination” for installation. It is okay to chose the default option.



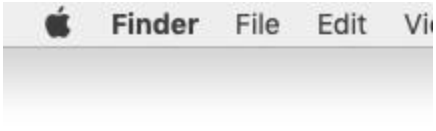
7) Click “Install” and “OK” to allow the installer to access the folders, if necessary. Allow the installation to complete



8) Click “Close”



9) Thonny IDE is now installed and you can double-click to open it:



6.2 Robotic arm doesn't work

☛Whether to turn on the power switch during installation to initialize the angle of the servo.

☛Please check whether the assembly is correct?

☛Please check whether the battery power is sufficient?

☛Please check whether the battery used in accordance with the specifications?

6.3 Port Connection & Device Recognition Errors

Q1: The port menu in the bottom right corner of Thonny does not have the "MicroPython (ESP32)" option.

Error Symptoms

- Thonny IDE bottom-right corner displays **"Local Python 3.x"** only, no **MicroPytho (ESP32)** option.
- When opening **Tools → Options → Interpreter → MicroPython (ESP32)**, there is no "**USB SeriaI @ COMx**" Port selection in the "**Port or WebREPL**" drop-down menu.

Root Causes

1. Missing **CH340 USB-to-Serial driver** (ESP32-C3 development boards use this chip for USB communication, Windows does not have it preinstalled);
2. Loose USB cable (using a charge-only cable instead of a data cable);
3. ESP32-C3 not powered on (USB not plugged in or board power switch off);
4. Damaged USB port (front panel USB ports on desktops are often unstable).

Step-by-Step Solutions

1. **Install the USB-to-Serial driver** (critical step):

- Follow 4.4 Install CH340 USB driver to install the

2. **Check the USB cable and port**:
   - Replace with a **data USB cable** (charge-only cables have no data pins and cannot transmit serial data);
   - Plug the USB cable into a **rear panel USB port** (desktop) or a direct laptop USB port (avoid USB hubs);
3. **Verify port in Windows Device Manager**:
   - Press `Win + X` → select **Device Manager** → expand the **Ports (COM & LPT)** section;
   - If ESP32-C3 is recognized, you will see `USB-SERIAL CH340 (COMx)` (x = number like 3,4,5);
   - If a yellow exclamation mark appears on the port, right-click → **Uninstall device** → re-plug the USB cable to reinstall the driver.

Q2: Thonny displays "could not open port 'COMx'" when you click STOP

Error Symptoms

- When you click **STOP** button in Thonny, the Shell will pop up the error "could not open port 'COM3'".
- The port is visible in Device Manager but cannot be used in Thonny.

Root Causes

1. The serial port is occupied by **other software** (e.g., Arduino IDE, Serial Monitor, Putty, VS Code serial plugins);
2. Thonny's previous serial connection was not closed properly (IDE cache issue);
3. Windows background process is holding the serial port.

Step-by-Step Solutions

1. **Close all serial port-related software**:
   - Close Arduino IDE, Serial Monitor, Putty, and any other tools that may use the serial port;

- Check the Windows taskbar for hidden background software (right-click taskbar → Task Manager → end processes related to serial tools).

2. **Restart Thonny IDE:**
   - Close Thonny completely (click X in the top-right corner, ensure no Thonny process in Task Manager);
   - Reopen Thonny and reselect the ESP32-C3 port.
3. **Replug the ESP32-C3 USB cable:**
   - Unplug the USB cable, wait 2 seconds, then plug it back in—this resets the serial port and releases the occupation.

6.4 MicroPython Firmware & Interpreter Configuration Errors

Q1: Thonny prompts "Device is busy or does not respond" when running code

Error Symptoms

- Click the **Run\STOP** button. Shell prompt : **Device is busy or does not respond.**

Root Causes

1. ESP32-C3 has **no MicroPython firmware installed** (brand-new boards usually have no firmware preloaded).
2. Wrong interpreter selected in Thonny (e.g., "MicroPython（ESP8266）" instead of "MicroPython (ESP32)").
3. Corrupted MicroPython firmware on the board (due to incomplete flashing or power failure during firmware installation).

Step-by-Step Solutions

1. **Flash firmware in Thonny:**
   i. Refer to Section 4.5 to re-flash the firmware(do not unplug the

USB cable during flashing.

ii. After completion, click **Close** and restart the ESP32-C3 (press RESET button).

2. **Verify firmware installation**:

- Open Thonny's Shell terminal (**View → Shell**);
- Press the ESP32-C3's RESET button—you will see the MicroPython version information and the `>>>` prompt (success).

Q2: Thonny shows "Wrong board model" during firmware installation

Error Symptoms

- When installing MicroPython firmware, a pop-up error: **"Wrong board model selected. The firmware is not compatible with ESP32-C3"**;
- Firmware flashing fails with a "Verification error" or "Write error".

Error Case Screenshot Description

Firmware installation progress bar stops at 10-50%, and a red warning pops up: "Firmware for ESP32 (generic) is not compatible with this device. Please select ESP32-C3 as the board model".

Root Cause

Selected the wrong **board model** in the firmware installation page (e.g., "ESP32 (generic)" instead of "ESP32-C3")—ESP32-C3 uses a RISC-V core, and generic ESP32 firmware (Xtensa core) is incompatible.

Step-by-Step Solution

1. Refer to Section 4.5 to re-flash the firmware(do not unplug the USB cable during flashing.

6.5 Runtime Errors (Code Execution on ESP32)

Q1: Code shows "ImportError: no module named 'xxx'" when running on ESP32-C3

Error Symptoms

- Clicking Run or restarting the ESP32-C3 shows a red error in the REPL terminal: **"ImportError: No module named 'led'"**.
- The main code (e.g., `main.py`) fails to execute because it cannot import a custom module.

Error Case Screenshot Description

Shell displays:

```
Plain Text
>>> import led
ImportError: No module named 'led'
>>> main.py:1: ImportError: No module named 'sensor'
```

Root Causes

Case 1: Custom module (e.g., led.py, sensor.py) → Not uploaded to ESP32-C3

- The imported module file is only on the Windows local computer, not in the ESP32-C3's file system.

Case 2: Custom module → Filename case mismatch

- MicroPython is **case-sensitive** for filenames (e.g., `Led.py` ≠ `led.py`, `SENSOR.py` ≠ `sensor.py`).

Step-by-Step Solutions

1. **For custom modules: Upload the module file to ESP32-C3**
    - In Thonny, open the missing module file (e.g., `led.py`);
    - Click **File → Upload to/** to upload it to the ESP32-C3;
    - Verify the file is visible in the Thonny File panel (under the MicroPython device);

- Re-run the main code or restart the board.

2. **For custom modules: Fix filename case mismatch**
   - Ensure the **import filename matches the actual filename exactly** (case and spelling);

     Example: If the file is `LedBlink.py`, the import must be `import LedBlink` (not `import ledblink`);
   - Rename the file to **all lowercase** (recommended for beginners) to avoid case errors (e.g., `ledblink.py`).

Q2: Code shows "OSError: [Errno 2] ENOENT" when running on ESP32

Error Symptoms

- Shell terminal displays **"OSError: [Errno 2] ENOENT"** when executing code;
- This error usually occurs with file operations (e.g., opening a file) or module imports.

Error Case Screenshot Description

```
Plain Text
>>> f = open("data.txt", "r")
OSError: [Errno 2] ENOENT
>>> from led import run
OSError: [Errno 2] ENOENT
```

Root Cause

ENOENT = Error NO ENTity → The file/module the code is trying to access **does not exist** on the ESP32's file system (the most common error for beginners).

- For file operations: The target file (e.g., `data.txt`) is not uploaded to the board;
- For imports: The module file (e.g., `led.py`) is missing or the path is wrong.

Step-by-Step Solutions

1. **Verify the file/module exists on the ESP32-C3**:
    - Check the Thonny File panel to confirm the file (e.g., `data.txt`, `led.py`) is present under the MicroPython device;
    - If missing, upload the file to the board.
2. **Fix the file path in code**:
    - Use **only the filename** for root directory files (no absolute/relative paths like `/flash/led.py` or `./led.py`);

    Correct: `import led` / `f = open("data.txt", "r")`;

    Incorrect: `import flash.led` / `f = open("./data.txt", "r")`.
3. **Check for typos in the filename**:
    - Ensure the filename in the code matches the actual filename (no extra spaces, wrong letters);

    Example: `data.txt` ≠ `data .txt`, `led.py` ≠ `ledd.py`.

Q3: Code runs in Thonny (Run button) but not after ESP32 restart

Error Symptoms

- Clicking the **Run** button in Thonny makes the ESP32's buzzer sounds normally;
- After restarting the board (unplug USB/repress RESET), the buzzer stops emits sound (no code runs automatically).

Error Case Screenshot Description

Shell terminal shows the code executing successfully when clicking Run: `>>> # buzzer...`; after restart, the terminal has only the MicroPython version prompt and no code execution.

Root Causes

1. The code is **not saved as `main.py`** on the ESP32-C3 (MicroPython runs `main.py` automatically on startup);
2. `main.py` is saved on the **Windows local computer**, not uploaded to the board;

3. main.py has **syntax errors** (causes automatic execution to fail on startup).

Step-by-Step Solutions

1. **Ensure the entry code is saved as main.py and uploaded** (core fix):
   - If using single-file code: Save the code as main.py (**File → Save As**) → Upload to the ESP32(**File → Upload to**);
   - If using modular code: Upload all custom modules (e.g., buzzer.py) → Create a minimal main.py (e.g., import buzzer; buzzer.run()) → Upload main.py to the board.
2. **Verify main.py is in the ESP32 root directory**:
   - Check the Thonny File panel to confirm main.py is visible under the MicroPython device (no subfolders);
   - If missing, re-upload main.py.
3. **Check main.py for syntax errors**:
   - In Thonny, open the main.py file on the board (double-click in the File panel);
   - Look for **red underlines** (Thonny's syntax check) → fix errors (e.g., missing colons :, indentation errors, wrong pin numbers);
   - Re-upload the corrected main.py and restart the board.
4. **Test automatic execution**:
   - After uploading the correct main.py, restart the ESP32 → the code runs automatically (buzzer) with no need to click Thonny's Run button.

6.6 File System & Device Display Abnormalities

Q1: The Thonny file panel doesn't shows "MicroPython Device" ?

Error Case Screenshot Description

Thonny File panel displays the MicroPython device entry, but the right pane has red text: "Could not list files on device. Check connection and try again."

Root Causes

1. This phenomenon occurs every time the computer's USB port is connected to the control board, because Thonny lacks the automatic detection feature for ESP32 device connection.
2. Serial port communication is unstable (loose USB cable, bad USB port).

Step-by-Step Solutions

1. **Making Thonny recognize the ESP32 device:**
    - Click the **STOP** button, or reselect "**MicroPython (ESP32) • USB Serial @ COMx**" in the bottom right corner of Thonny.
2. **Stabilize the serial connection:**
    - Replace the USB data cable → plug into a rear desktop USB port (avoid hubs);
    - Ensure the USB cable is not loose (no intermittent LED blinking on the board).

6.7 Serial Port Occupation & Connection Drop Issues

Q1: ESP32-C3 disconnects randomly from Thonny (COM port disappears)

Error Case Screenshot Description

Thonny bottom-right corner suddenly switches from "MicroPython on COM3" to "Local Python 3.x"; Device Manager's Ports section no longer shows the USB-SERIAL CH340 port.

Root Causes

1. **USB power supply instability** (laptop/USB hub provides insufficient power to the ESP32);
2. Damaged USB-to-Serial chip on the ESP32 board (hardware issue);
3. Windows USB driver conflict (outdated CH340 driver);

Step-by-Step Solutions

1. **Improve USB power supply** (most common fix):
    - For laptops: Plug the USB cable into a **powered USB hub** (or connect the laptop to a power adapter);
    - For desktops: Always use **rear panel USB ports** (they provide more stable power than front panels);
    - Avoid using unpowered USB hubs (they cannot supply enough current for the ESP32-C3).

7
Contact Us

If you couldn't find a solution above, please contact our support team.

To help us assist you quickly, please have the following information ready:

Your order number.
Product model (e.g., Robot Arm Kit E1R0000) and software version
A detailed description of the issue or your question.
Steps you have already tried.
Any relevant error message screenshots, photos, or code snippets.

Other Inquiries?

Please feel free to reach out for:

Tutorial Errors & Feedback: Help us make our documentation better.
Product Ideas & Suggestions: We'd love to hear your great ideas.
Partnerships & Collaboration: Interested in working together? Let's talk.
Discounts & Promotions: Inquire about educational or bulk pricing.
Anything Else: For all other non-technical questions.

 support@siyeenove.com

 https://siyeenove.com