

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
4.2 Install Thonny IDE	40

4.2 Basic Configuration of Thonny	44
4.3 Install CH340 USB driver	45
4.3.1 Install the Driver on Windows system	45
4.3.2 Install the Driver on Linux system.....	46
4.3.3 Install the Driver on MAC system.....	46
4.4 Burning Micropython Firmware (Important).....	47
4.4.1 Download MicroPython Firmware	47
4.4.2 Burn MicroPython Firmware	48
4.5 Basic Usage of Thonny	53
4.5.1 Test Shell commands	53
4.5.2 Run Online	53
4.5.3 Run Offline	55
4.5.4 Upload code from computer to ESP32-C3	60
4.5.5 Download the code from ESP32-C3 to computer	61
4.5.6 Deleting Files from your Computer Directory	61
4.5.7 Deleting Files from ESP32-C3' s Root Directory	62
4.5.8 Creating and Saving the code	62
4.6 Code Examples	65
4.5.1 Onboard buzzer	66
4.5.2 To Drive a Servo	70
4.5.3 Read the Value of the Joystick	73
4.5.4 Joystick Control eArm	76
4.5.5 Read the Value of the Web APP	79
4.5.6 Web App Control eArm	84
5. Factory Reset Function	90
5.1 Firmware	90
5.2 Burning Tool	91
5.3 How to Burn Firmware	92
6. QA	93
6.1 Unable to recognize USB serial ports	93
6.2 Robotic arm doesn't work	94
6.3 Other problems	94
7. Contact Us	94

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.

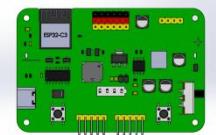
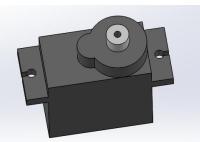
⑤ Web-Based Control

- 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

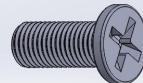
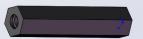
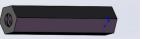
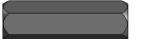
- Includes step-by-step Arduino programming tutorials, helping users advance from basic operations to custom robotics development.

1.1 Product List

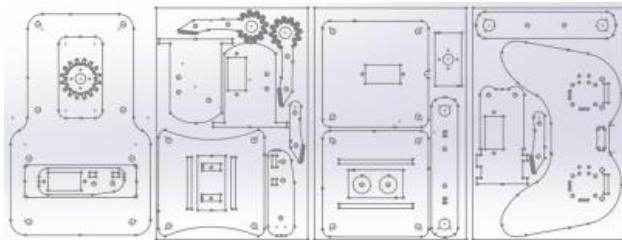
#Part 1 : Electronics Components		
Control Board 1PCS	Servo 4 PCS	5P Dupont Wire 2PCS
		
Joystick 2PCS	USB Cable 1PCS	
		

#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!

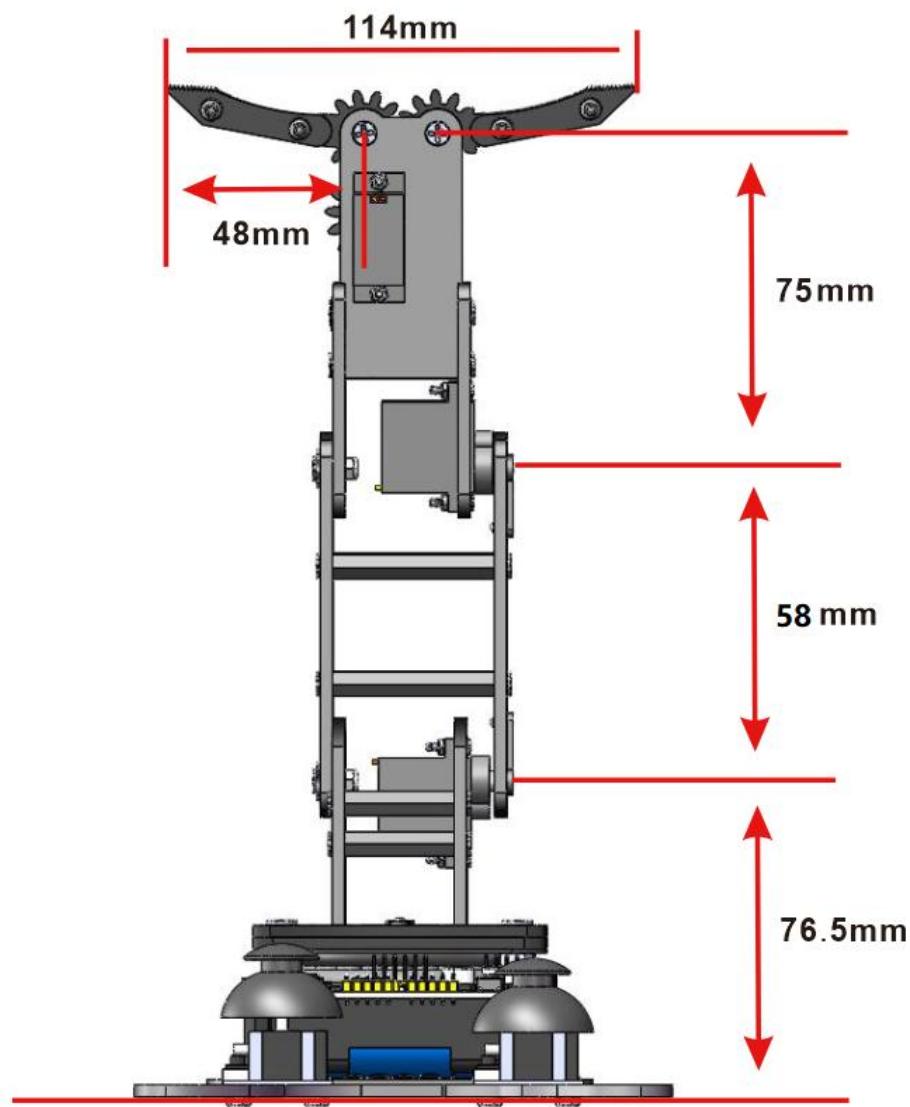
Parameters of 18650 lithium battery



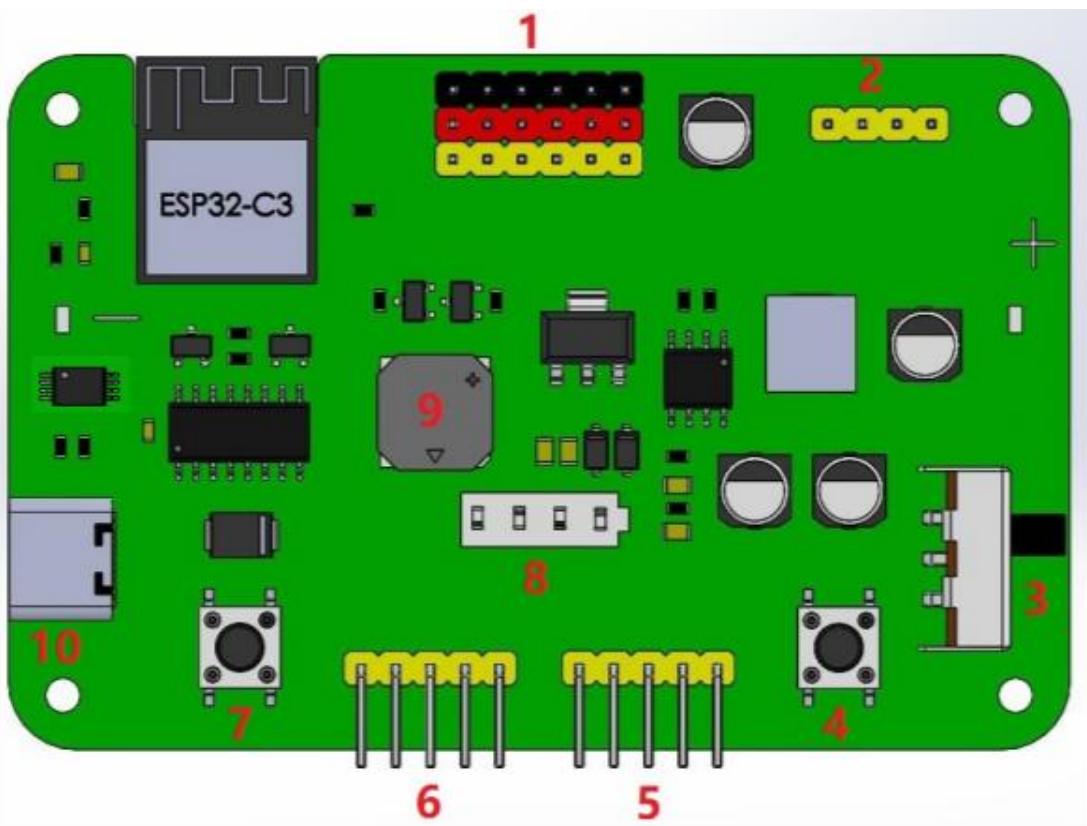
- Model: 18650 lithium battery
- Top type: Both button top and flat top are acceptable
- Capacity: >2000mAh
- Maximum charging voltage: 4.2V
- Nominal voltage: 3.7V
- End-off voltage: 2.75V
- Minimum charging current: >2A
- Minimum discharge current: >4A

1.2 Parameters of the eArm:

Control Board	eArm-ESP32-C3-MINI-1
Programming language	Arduino
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:

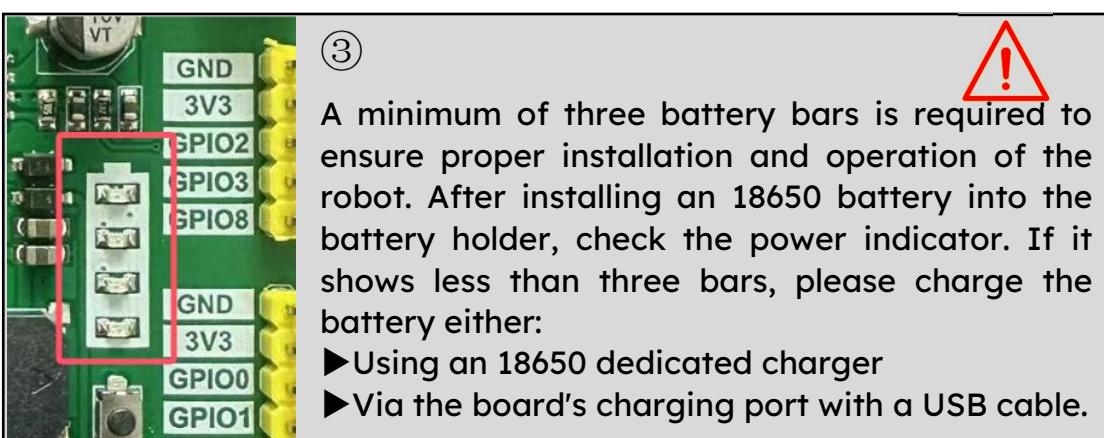


- 1: IO port, 5V/2A driving capability.
- 2: IO port and serial port expansion port.
- 3: Power switch.
- 4: Battery level display button(When the power switch is off, pressing it briefly can display the battery level).
- 5: IO port, 3.3V/500mA driving capability.
- 6: IO port, 3.3V/500mA driving capability.
- 7: Reset button.
- 8: Battery level indicator LED.
- 9: Buzzer.
- 10: USB Type-c Port for charging or uploading code

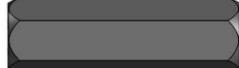
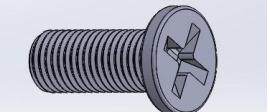
2

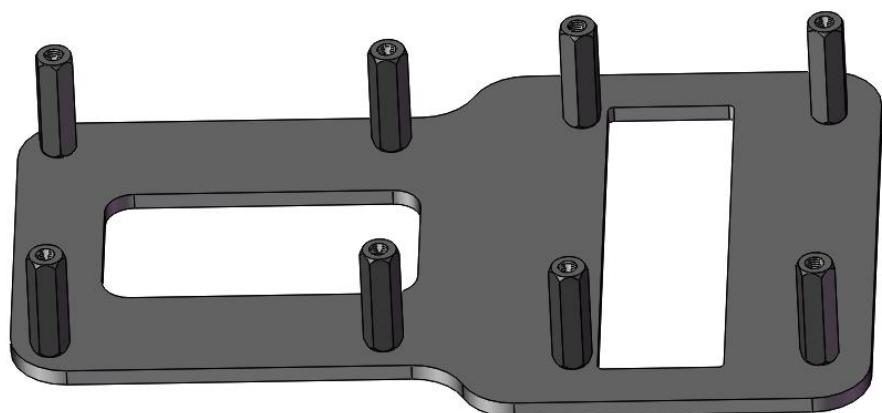
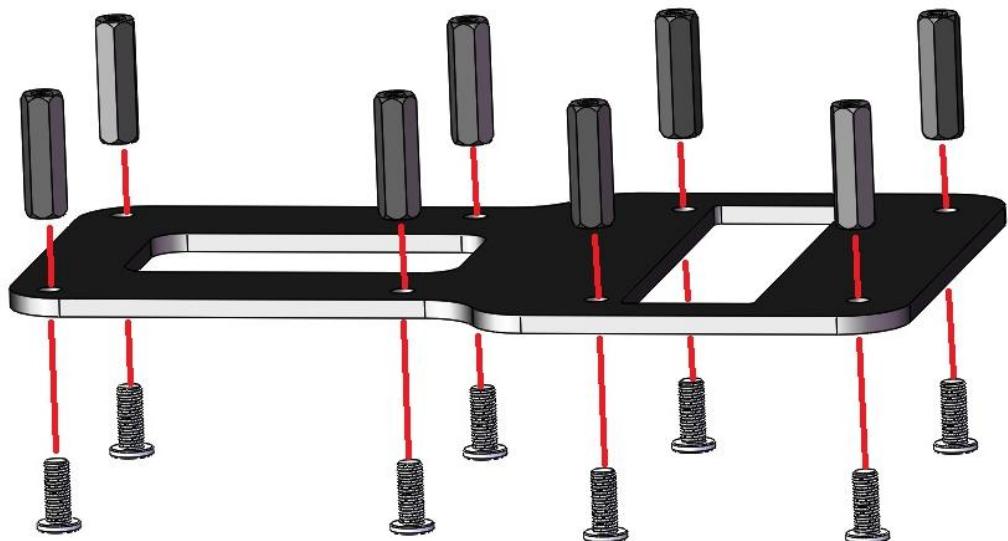
Assemble the eArm

Before You Begin: Important Notes

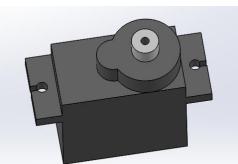
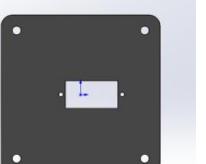


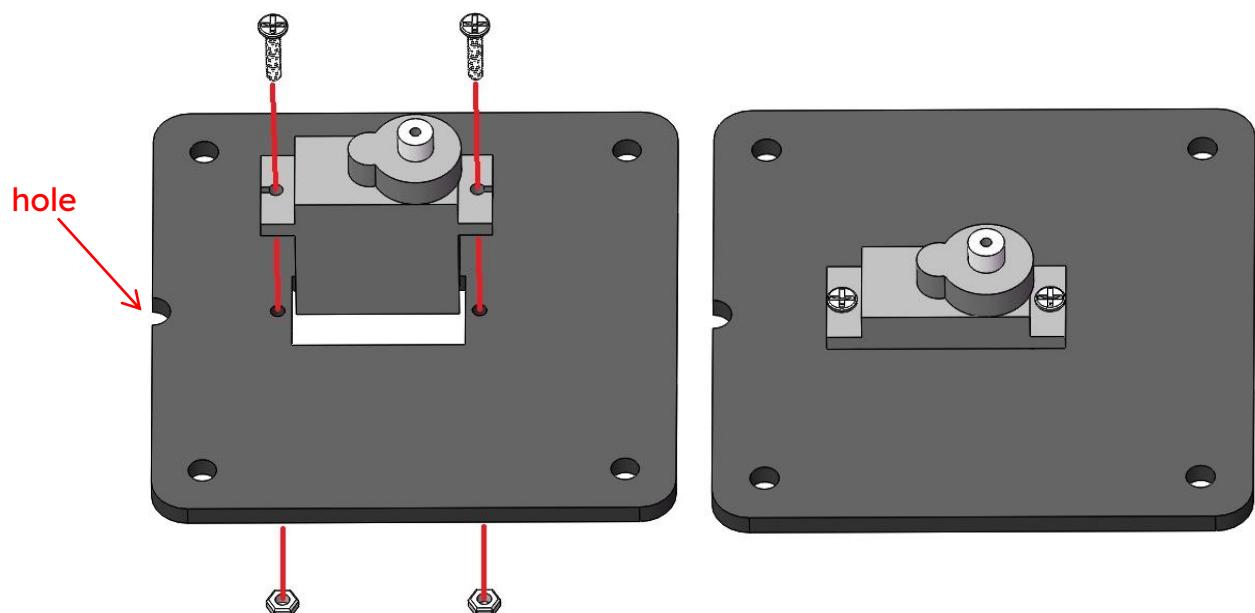
Step 1 Assemble the Robot Base

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

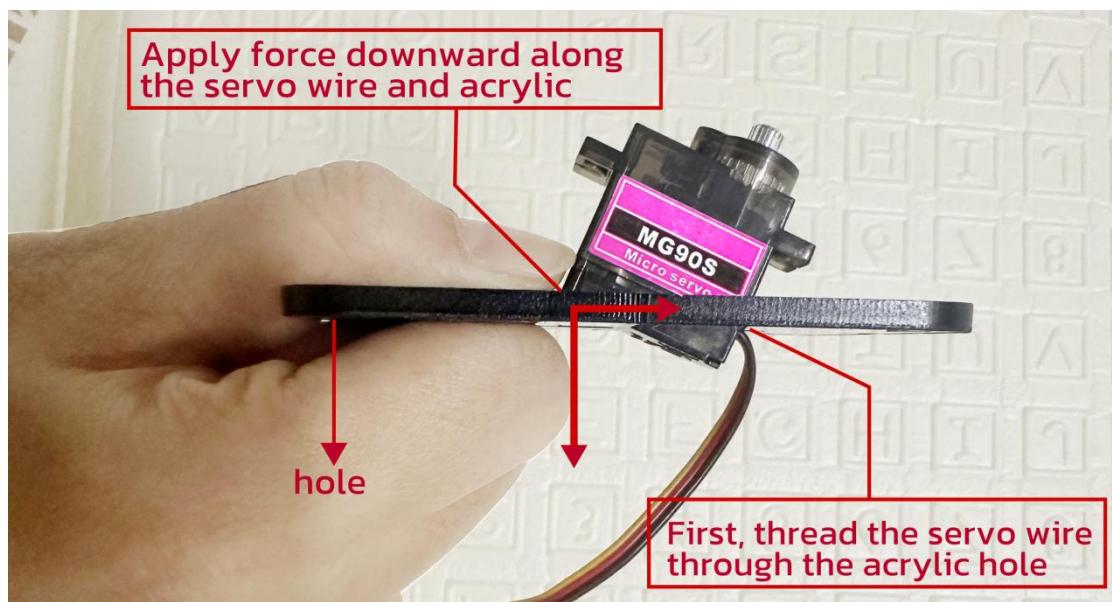


Step 2 Assemble Servo A on Robot Base

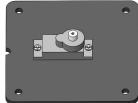
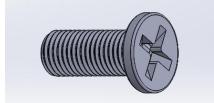
Servo 1 PCS	Bag No.④ M2x10MM Screw 2PCS	Bag No.⑧ M2 Nut 2PCS	Acrylic Sheet 1PCS
			

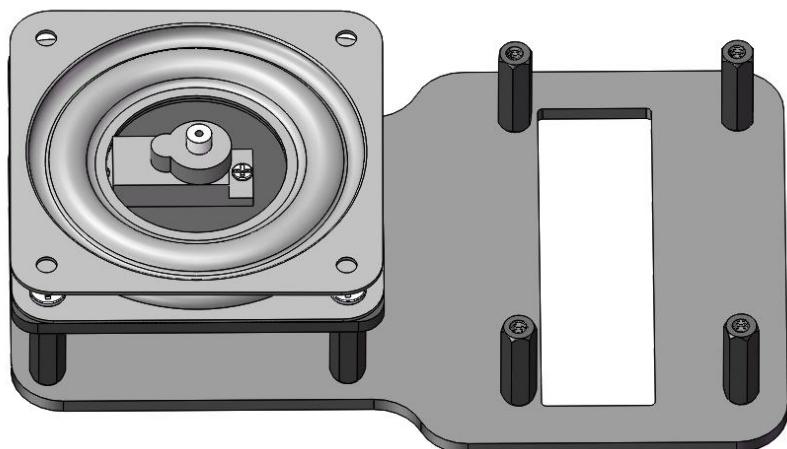
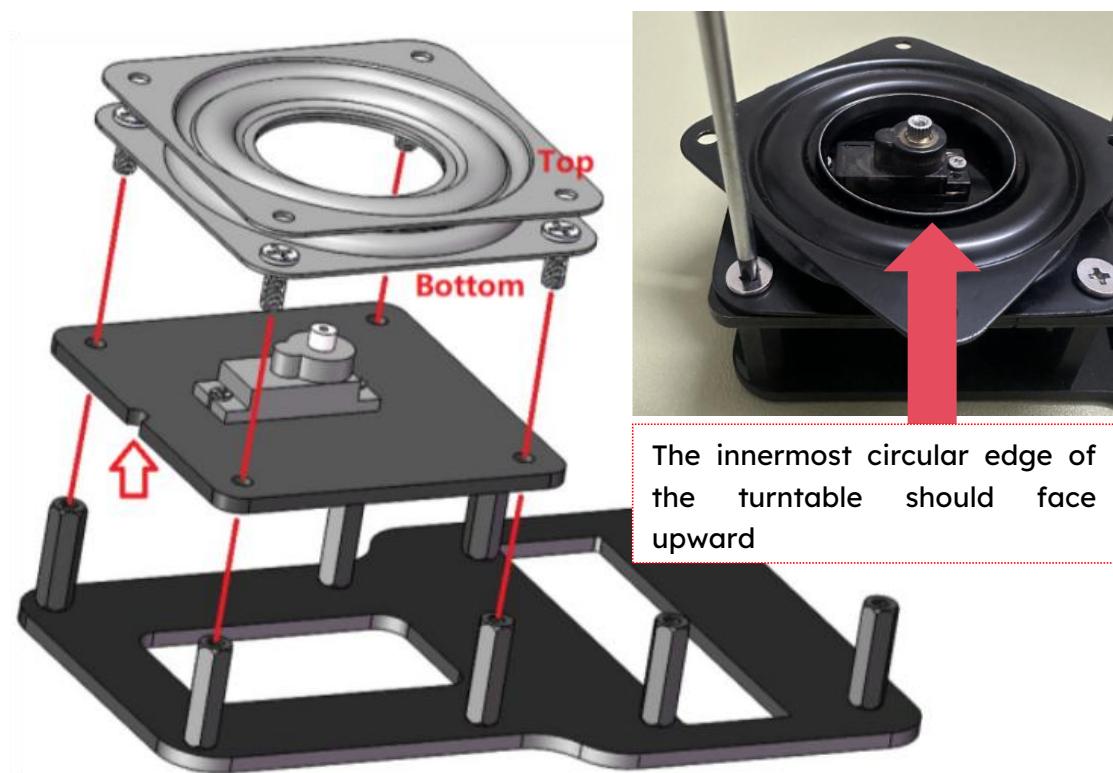


Note:



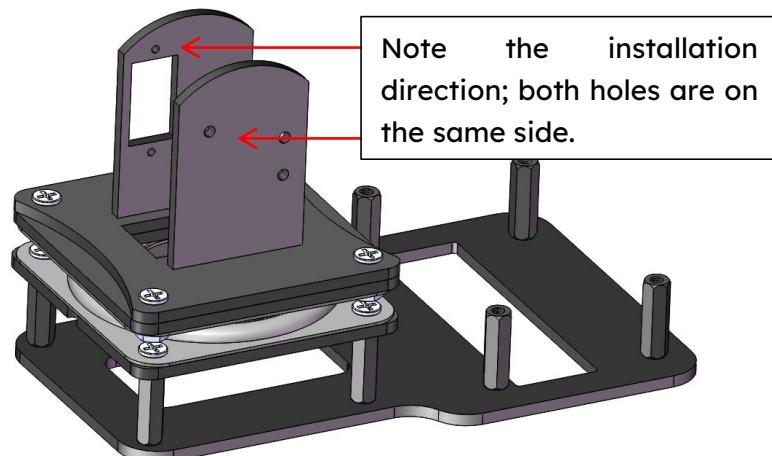
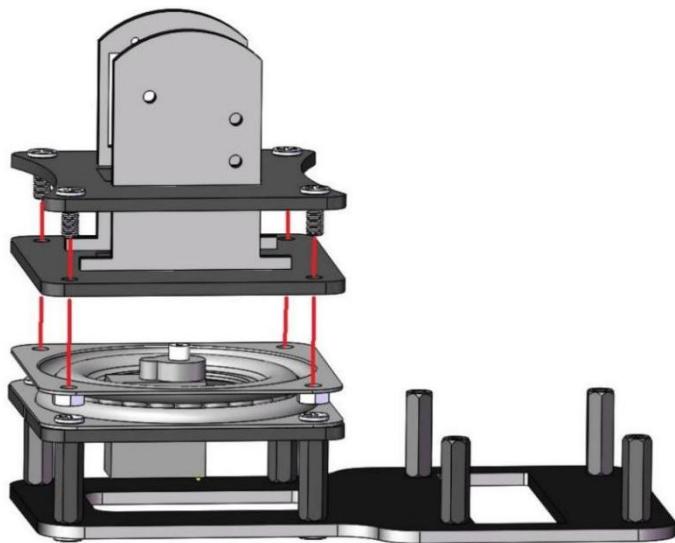
Step 3 Assemble the Turntable

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

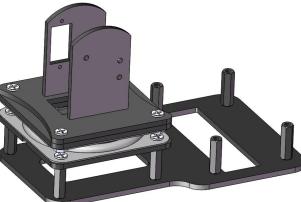


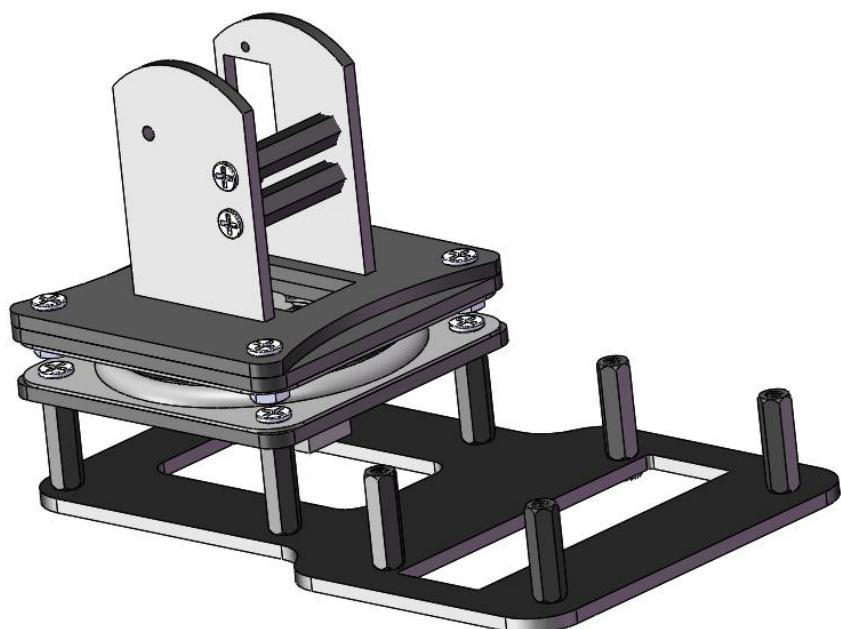
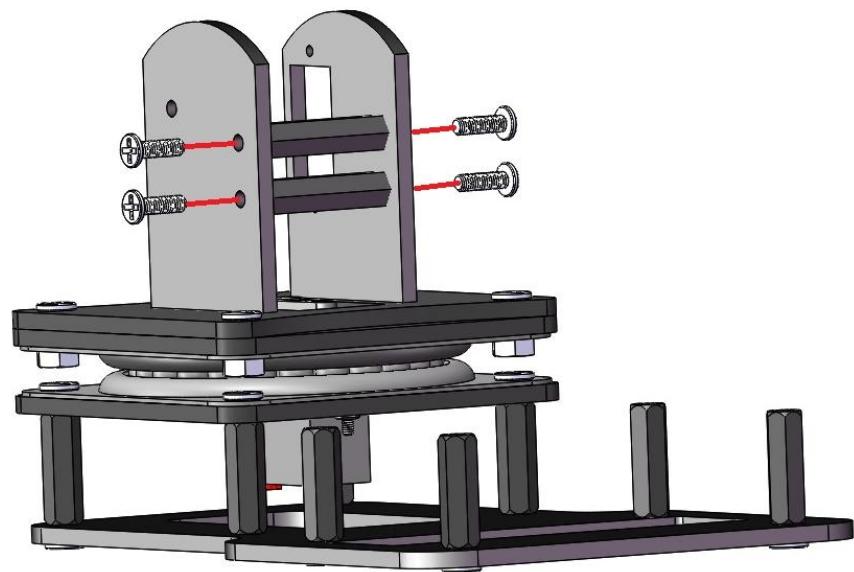
Step 4 Assemble the Lower arm

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

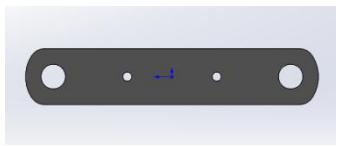
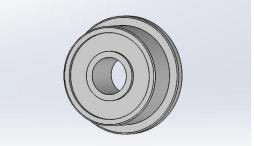


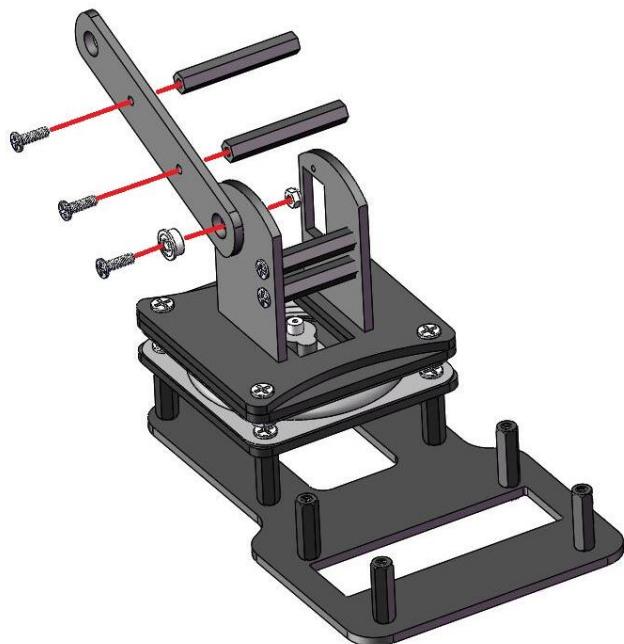
Step 5 Fix the lower arm

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

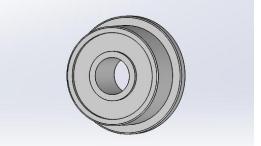
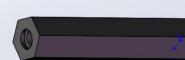


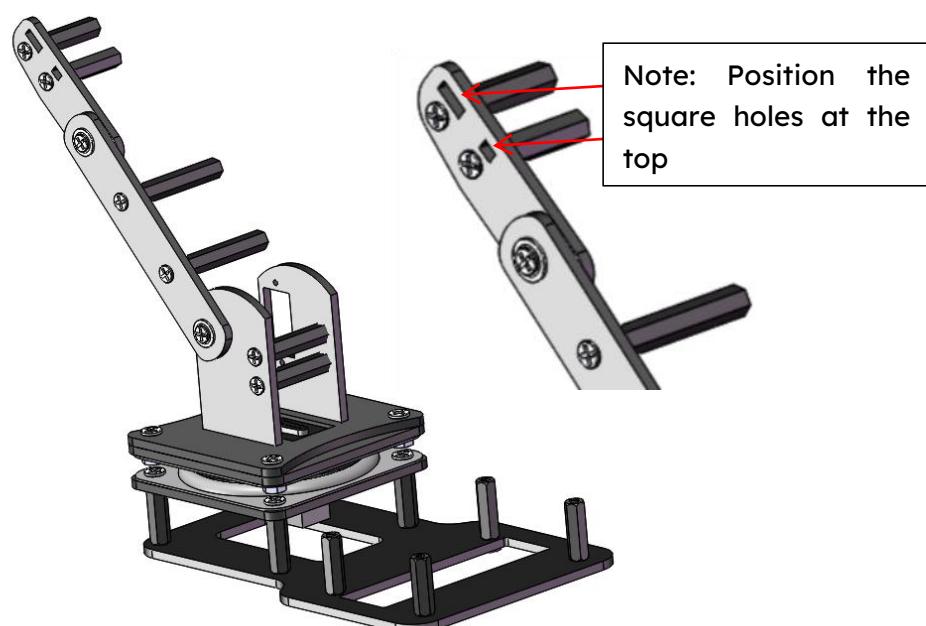
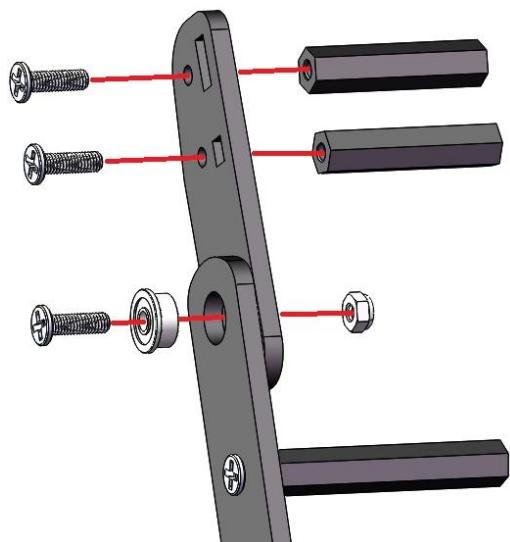
Step 6 Assemble the Middle Arm

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
		

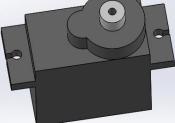


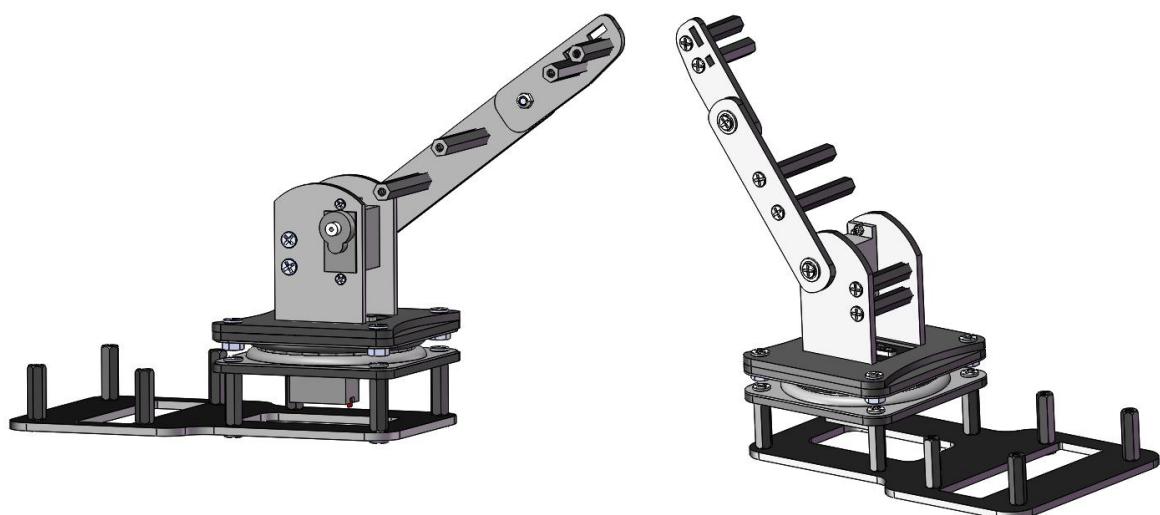
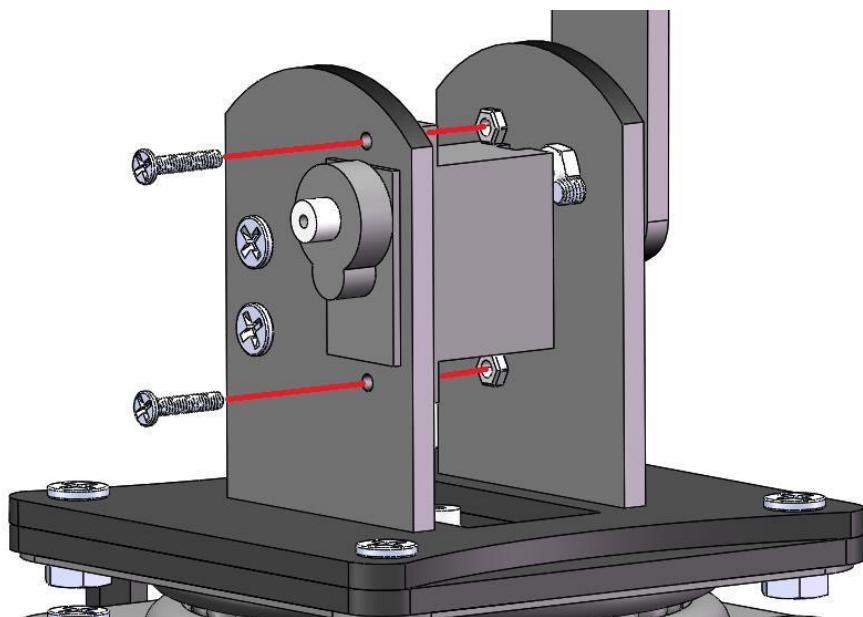
Step 7 Assemble the Upper Arm

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
		

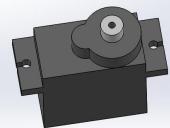
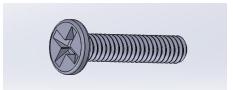


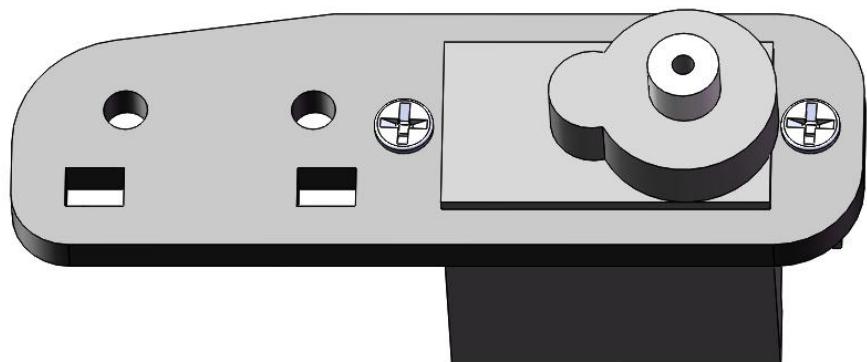
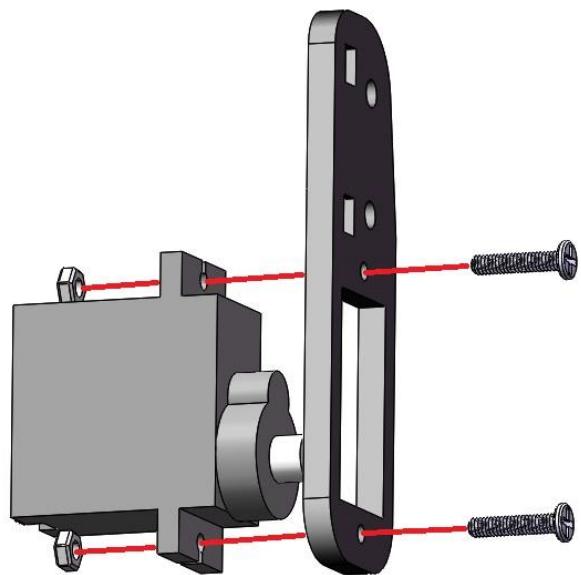
Step 8 Assemble the Servo B on lower arm

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

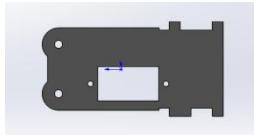
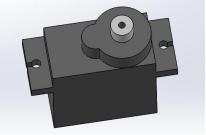
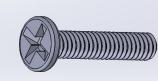


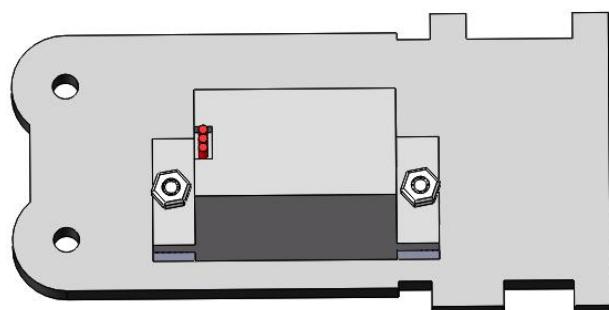
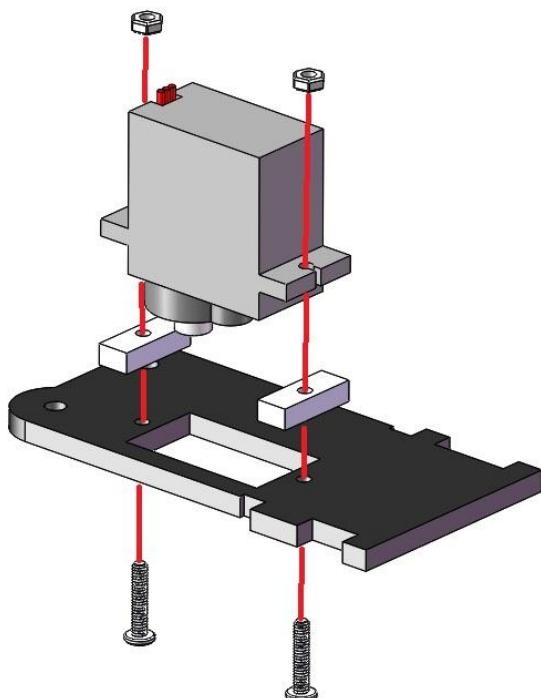
Step 9 Assemble the Servo C on Middle Arm

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



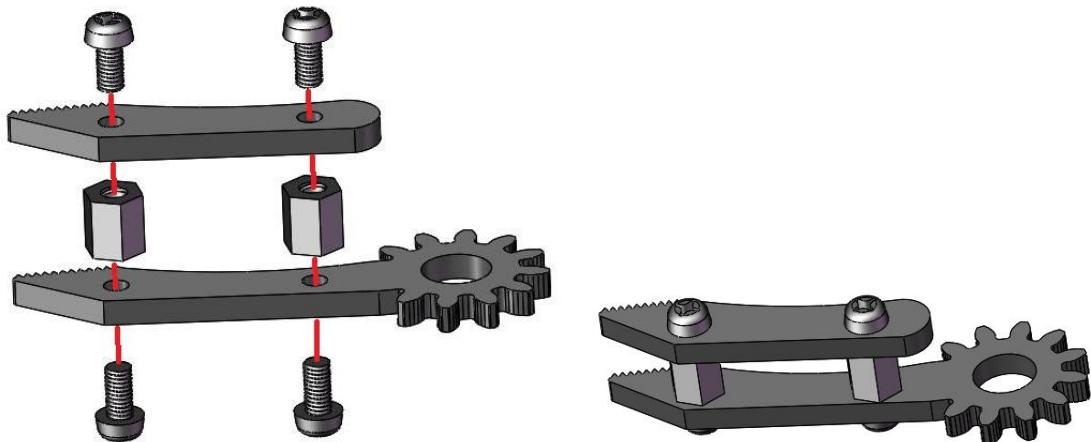
Step 10 Assemble the Servo D on Upper Arm

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



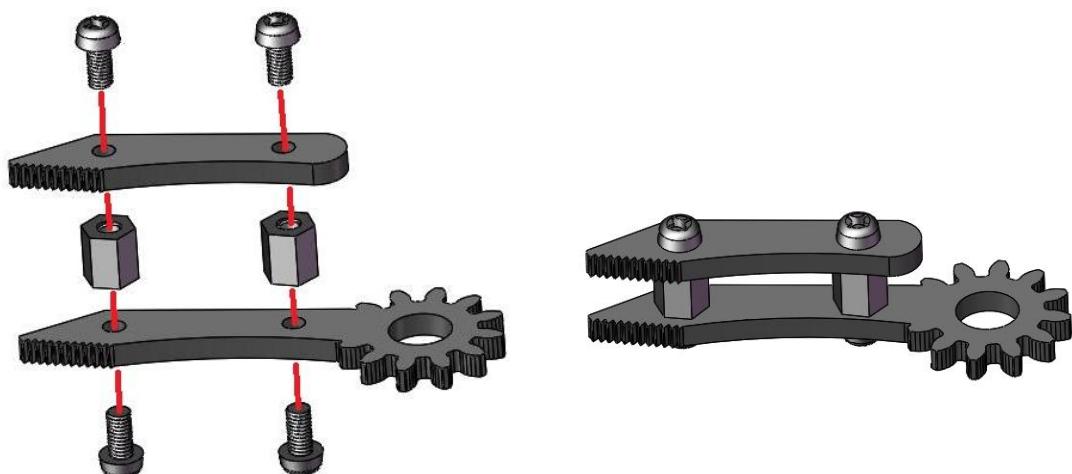
Step 11 Assemble the Left Clip Plate of The Clamp

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

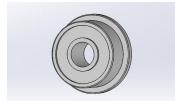


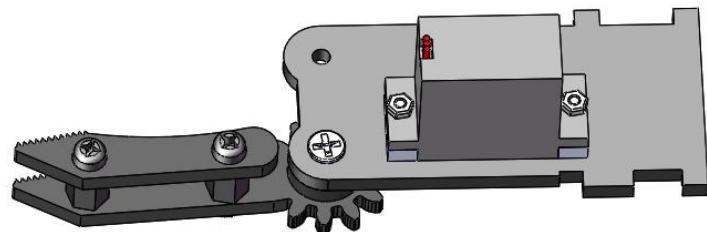
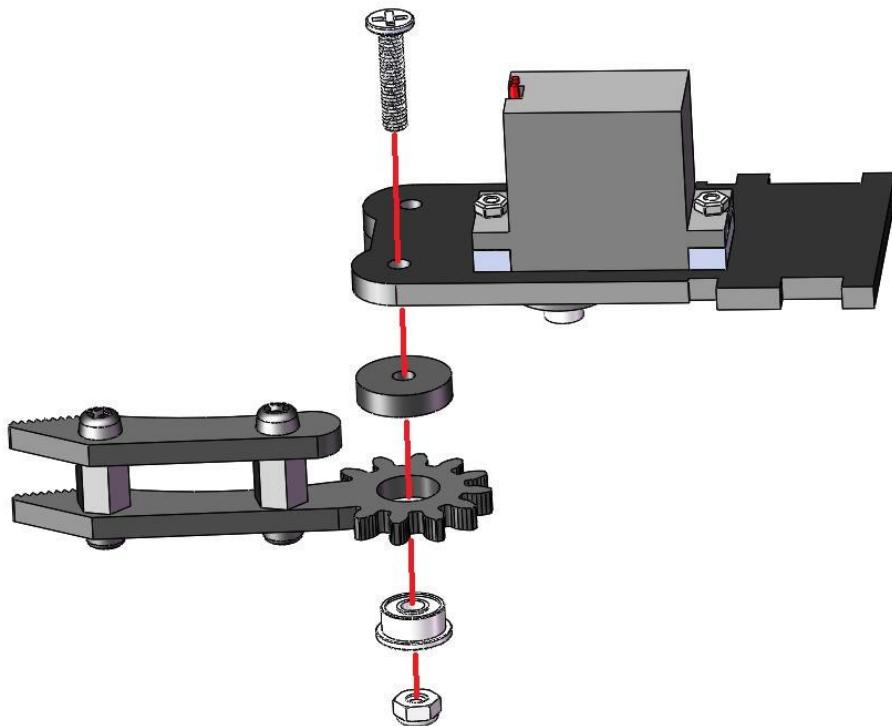
Step 12 Assemble the Right Clip Plate of The Clamp

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



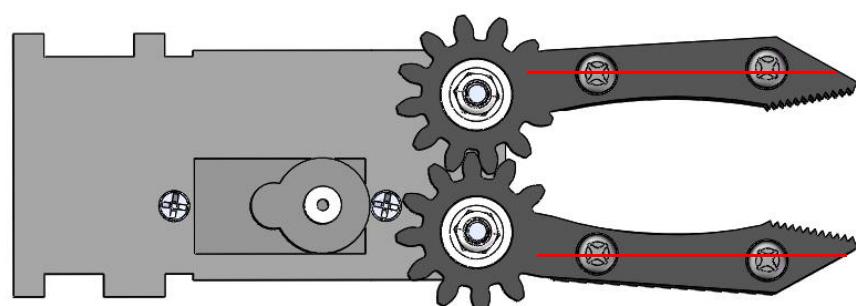
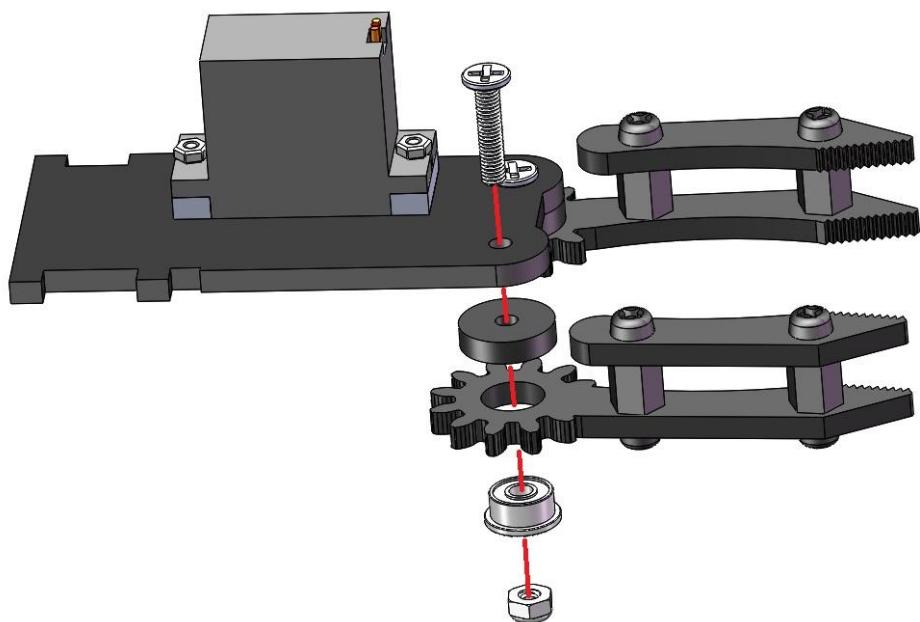
Step 13 Install the Left Clip Plate of The Clamp

Step 10 Structure	Step 11 Structure	Bag No.⑯ 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

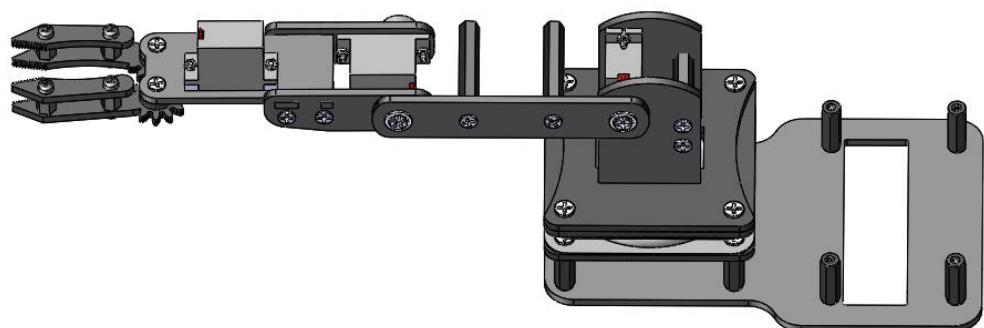
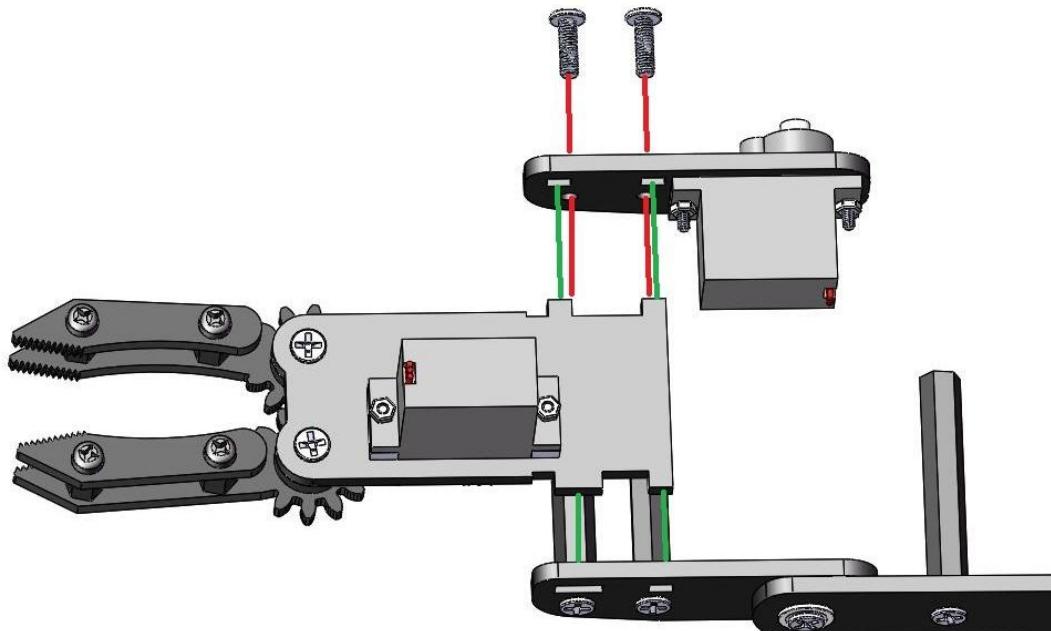
Step 13 Structure	Step 12 Structure	Bag No.⑯ Flange Bearing 1PCS
Acrylic 1PCS	Bag No.② M3X14MM Screw 1PCS	Bag No.⑦ M3 Lock Nut 1PCS



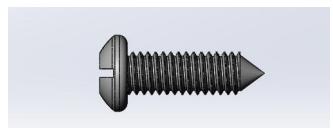
During installation, keep the screw threads horizontal

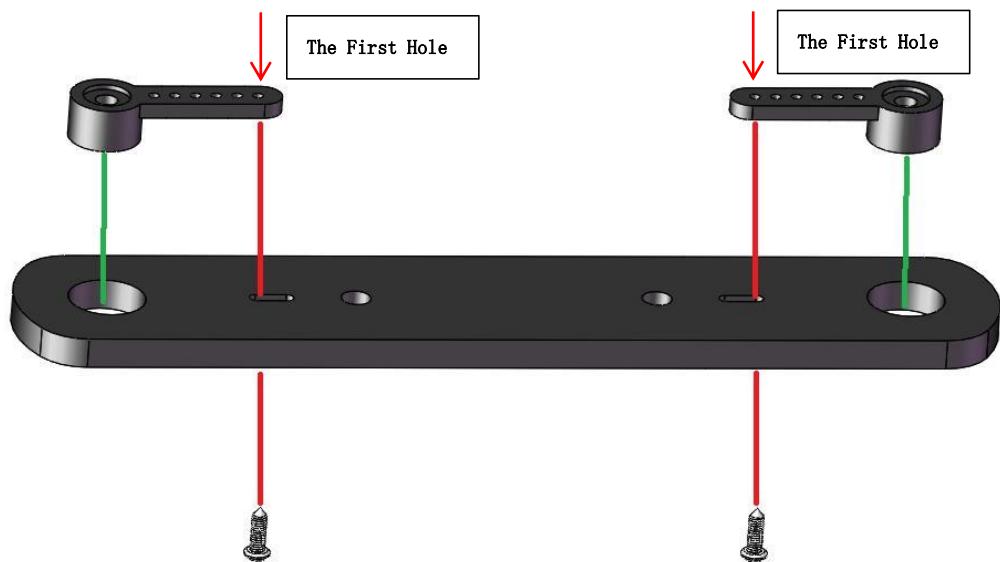
Step 15 Assemble the Clamp and Upper Arm

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

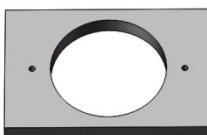


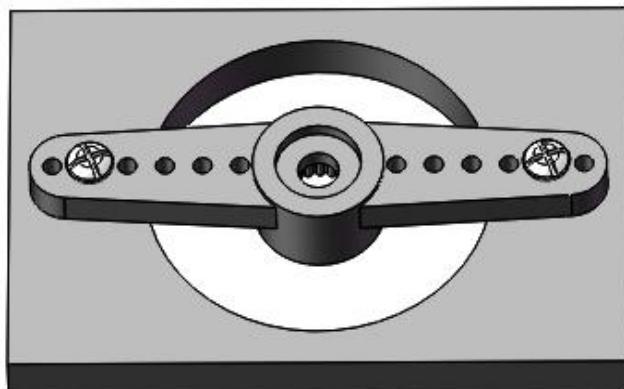
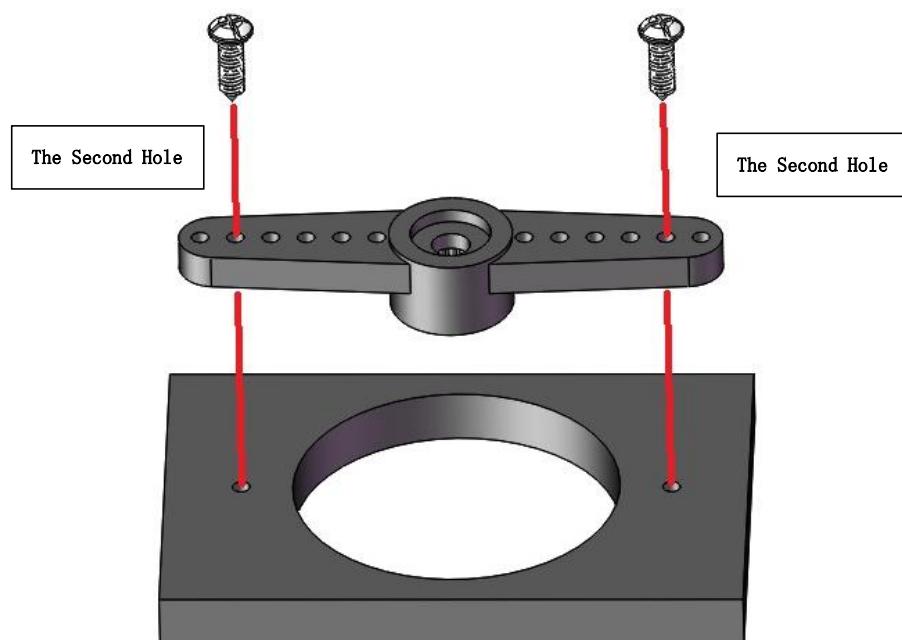
Step 16 Assemble the linkage between Servo B and Servo C

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

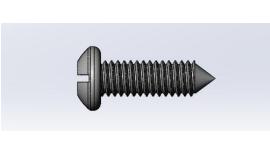


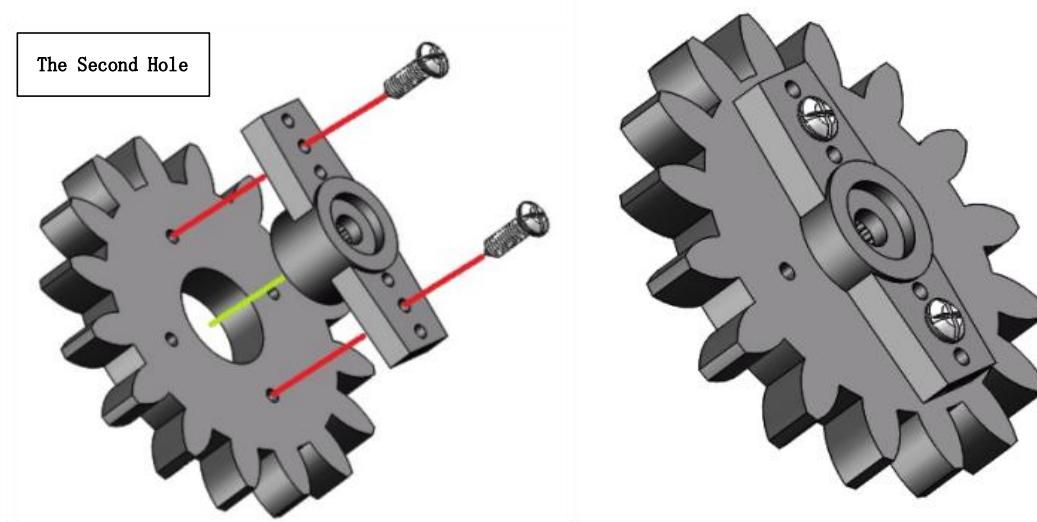
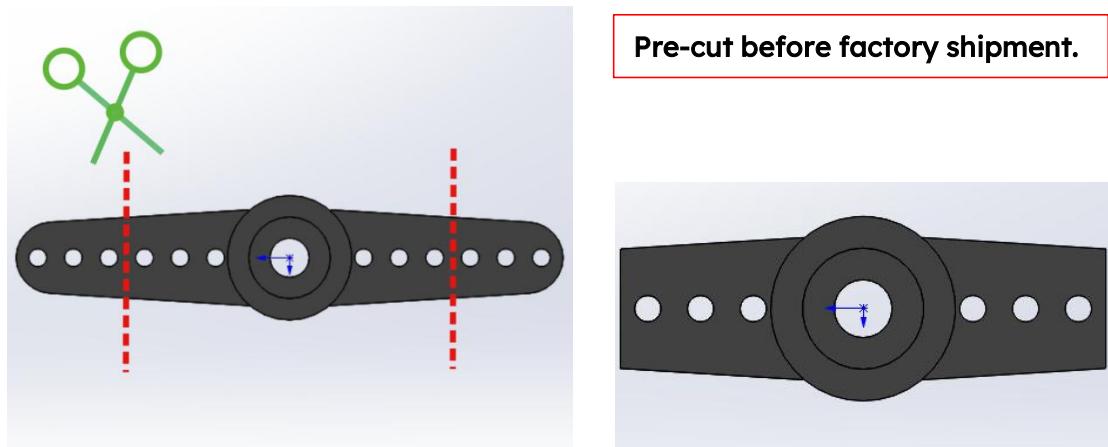
Step 17 Assemble the Mounting Structure for Servo A

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



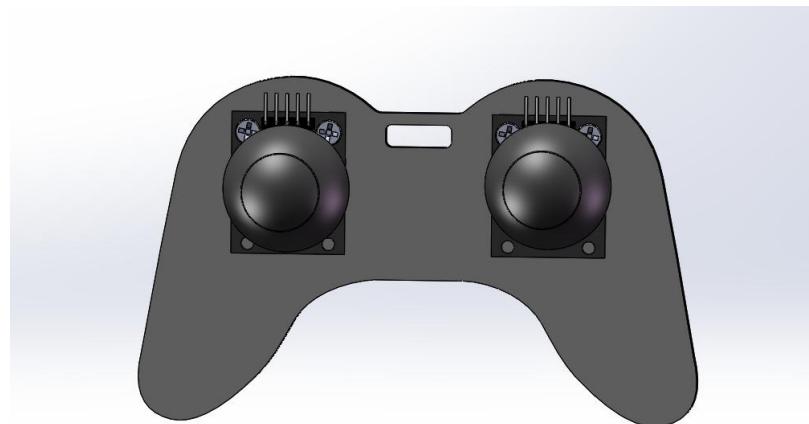
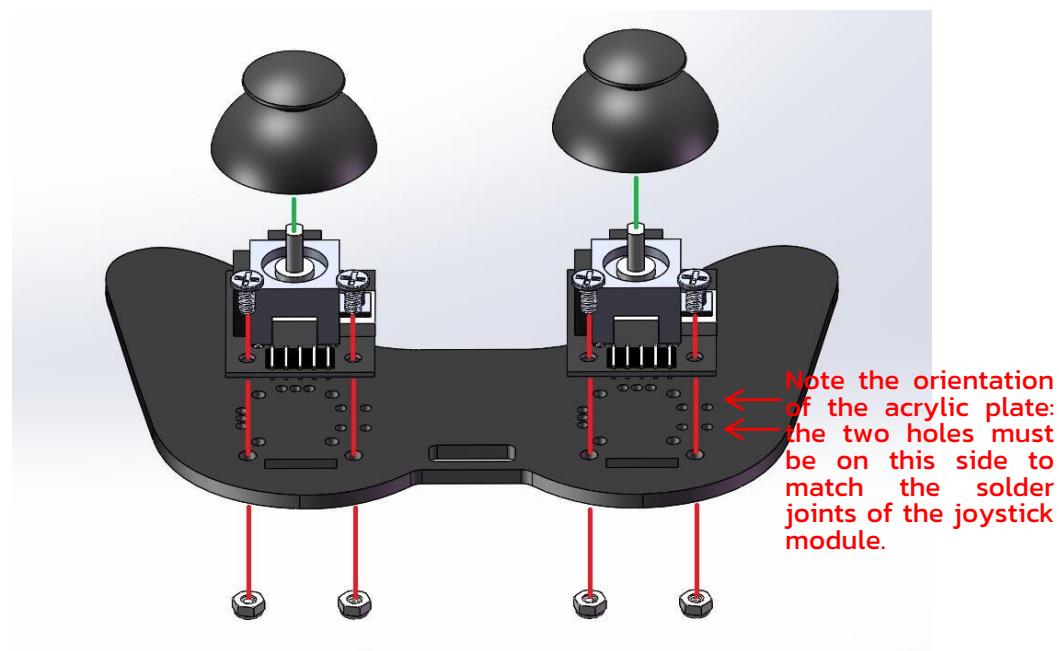
Step 18 Assemble the Mounting Structure for Servo D

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



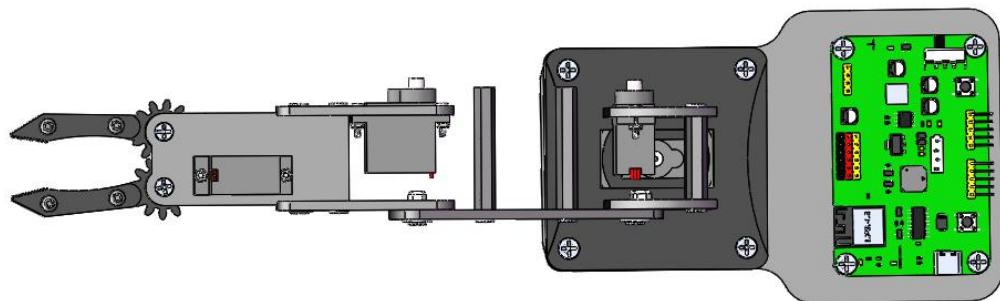
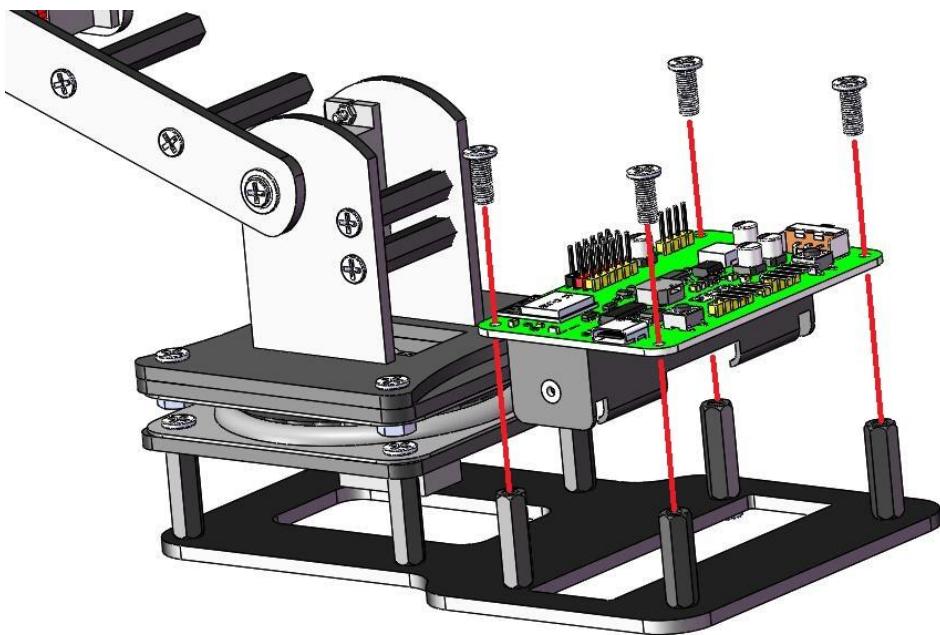
Step 19 Assemble the Joystick

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

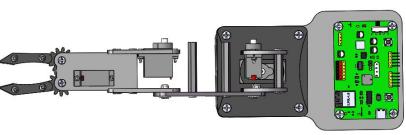


Step 20 Install the Esp32 Board

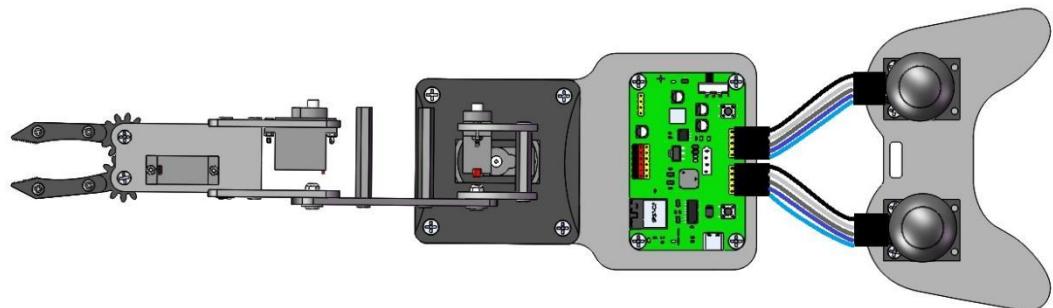
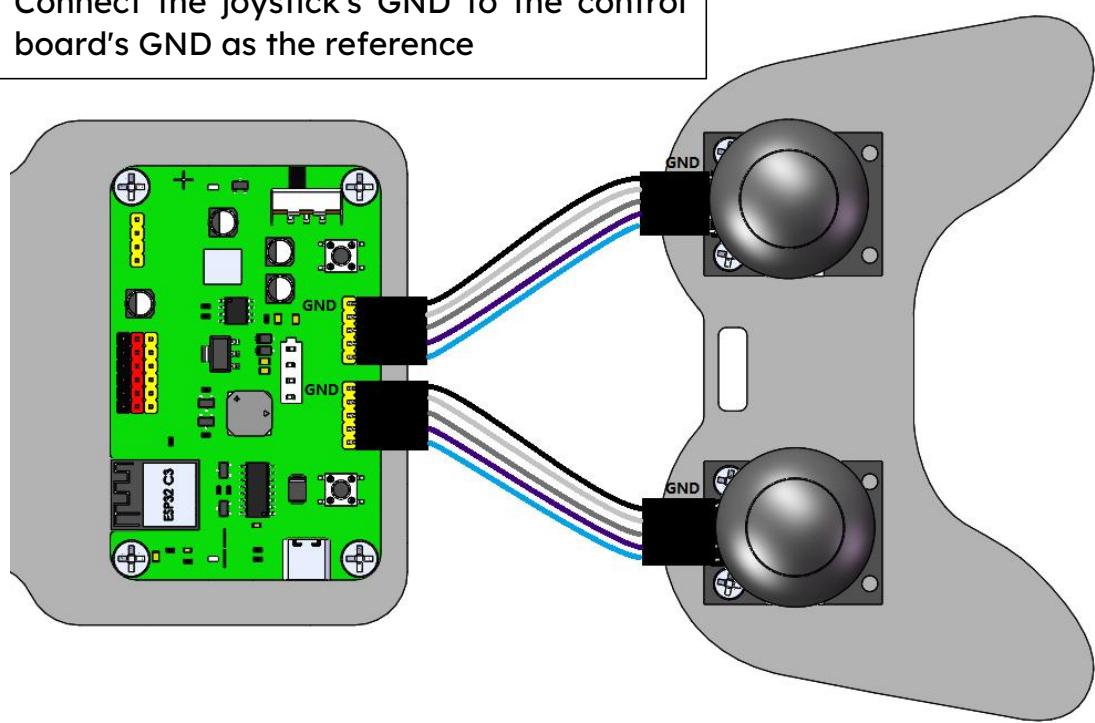
Step 15 Structure	ESP32 Control Board 1PCS	Bag No.① M4X10MM Screw 4PCS
		



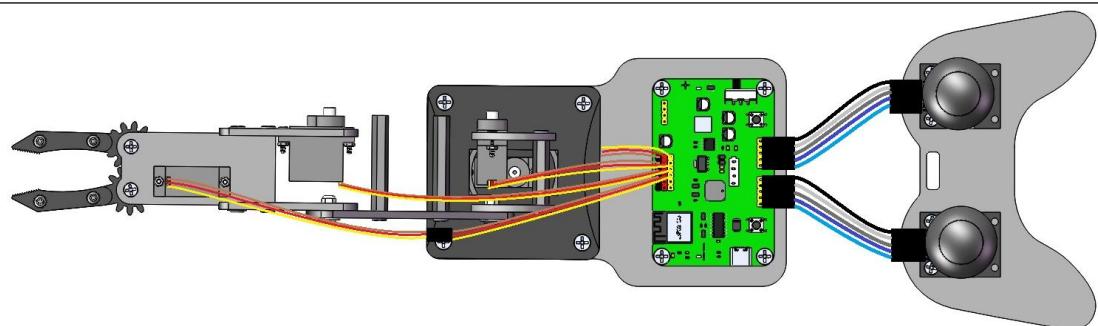
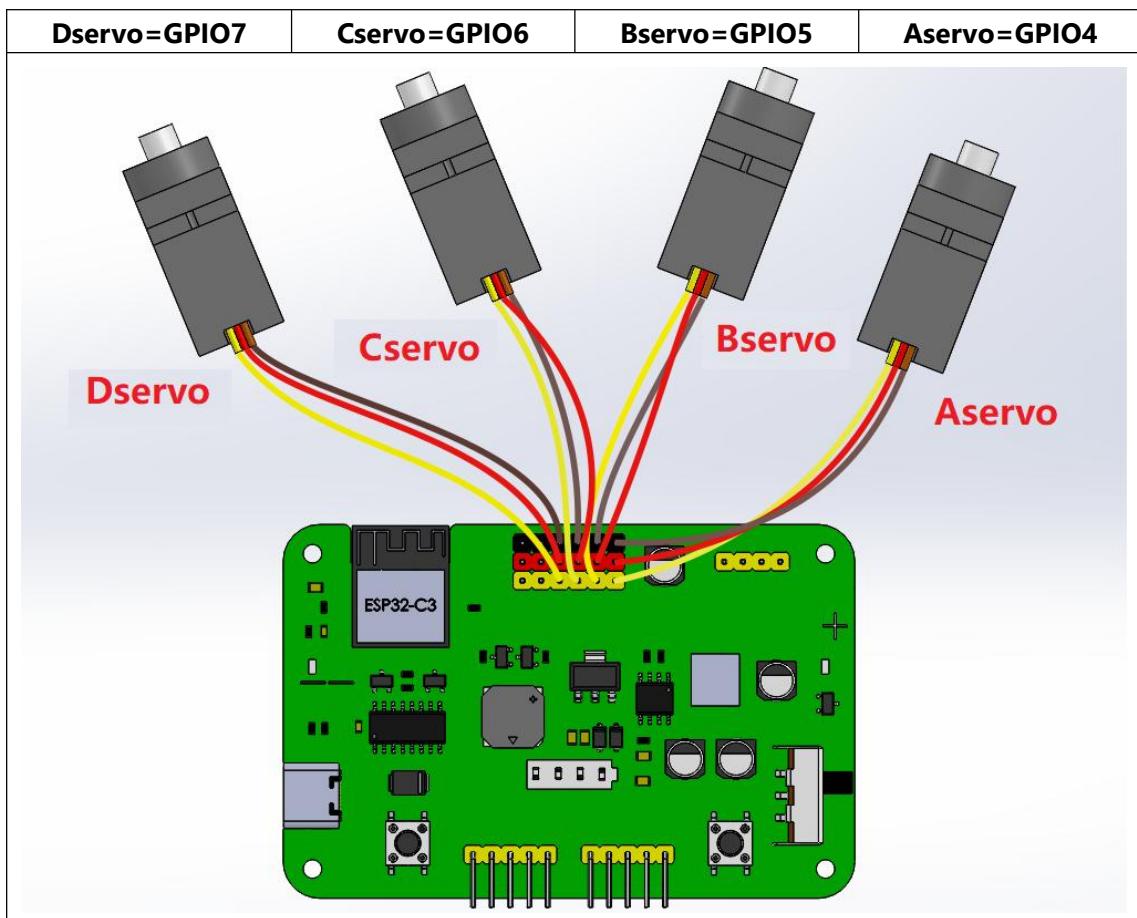
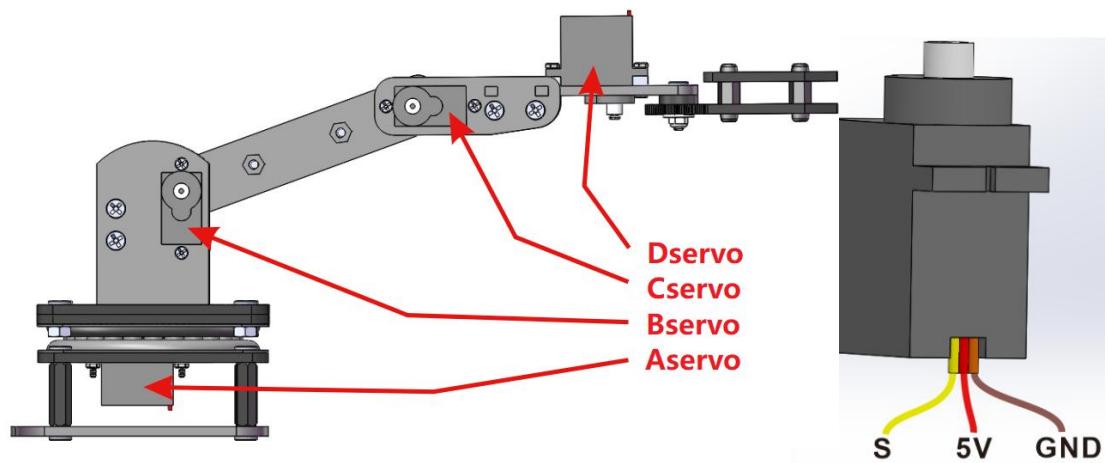
Step 21 Connect the Joystick

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

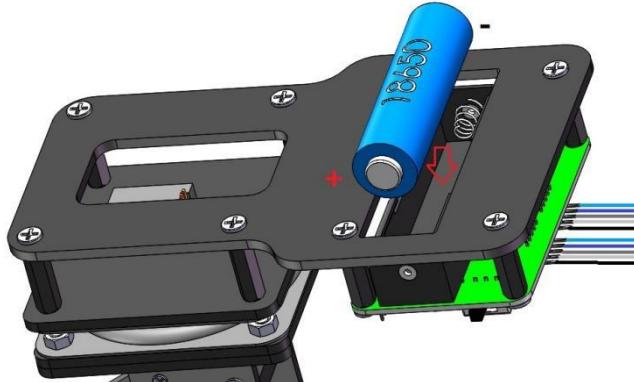


Step 22 Connect the Servos to the ESP32 Board

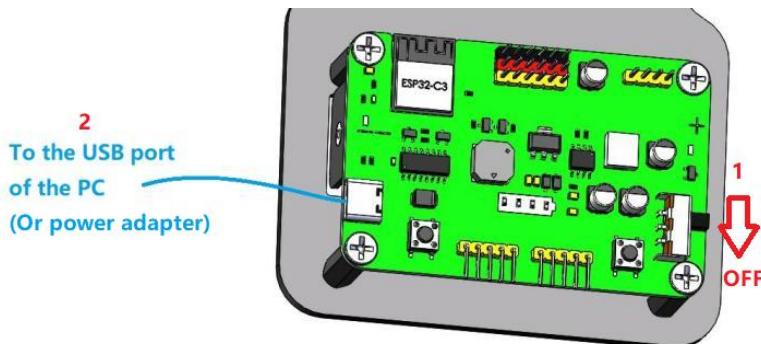


Step 23 Initialize the Servo Angle (important!)

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

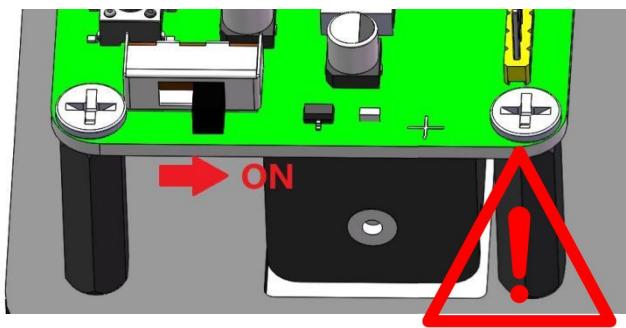


When installing the battery for the first time, please use the USB cable to charge the battery to activate the battery!



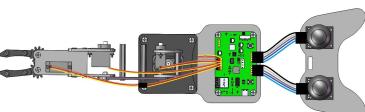
Very important: Turn on the power switch ! ! !

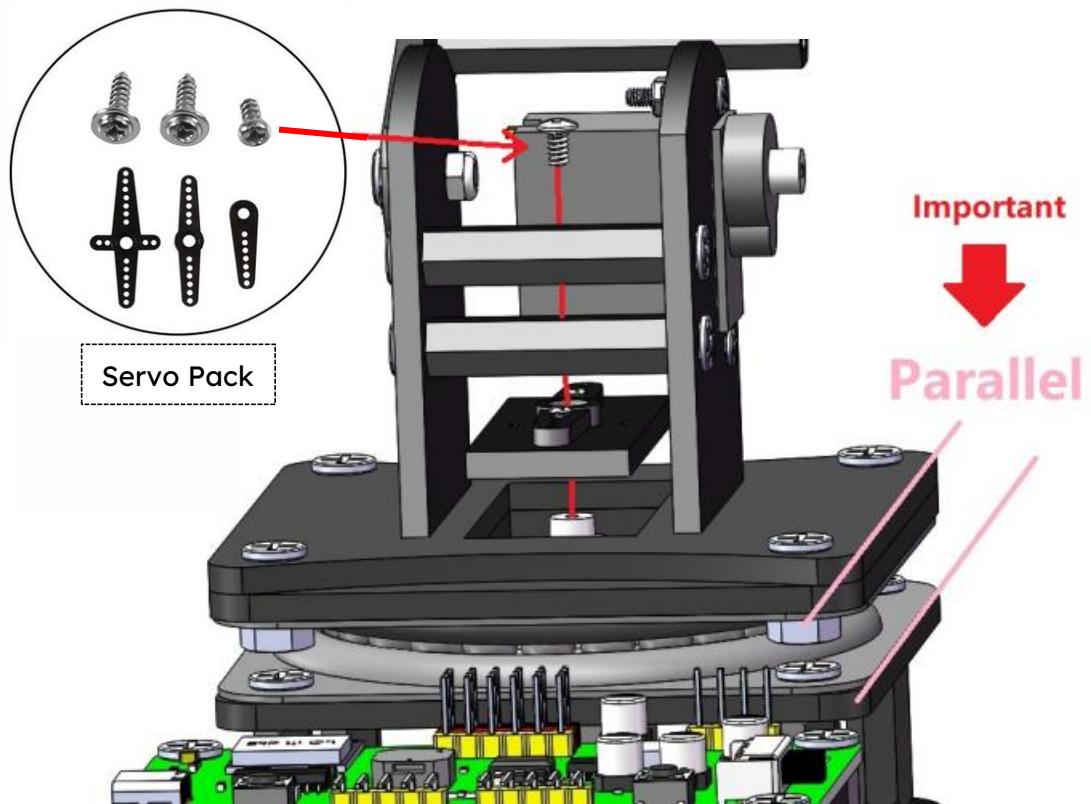
The **firmware** has been successfully programmed into the control board. When powered on, the board will automatically position all servo motors at the **90-degree** default position.



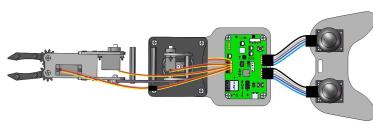
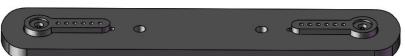
1. Make sure the battery is fully charged!
2. When you turn on the power switch, You will hear the servo motor shaft rotating.
3. In the following installation steps, please keep the power on to prevent the servo angle from being changed!

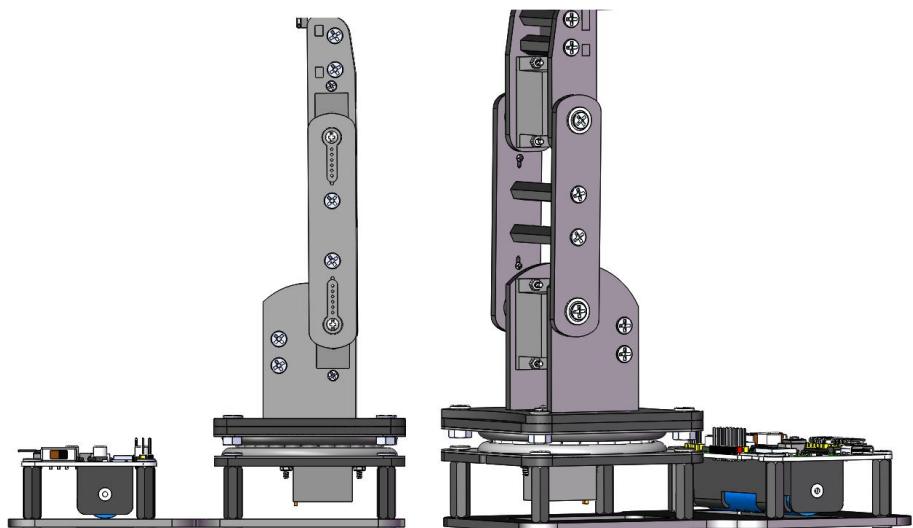
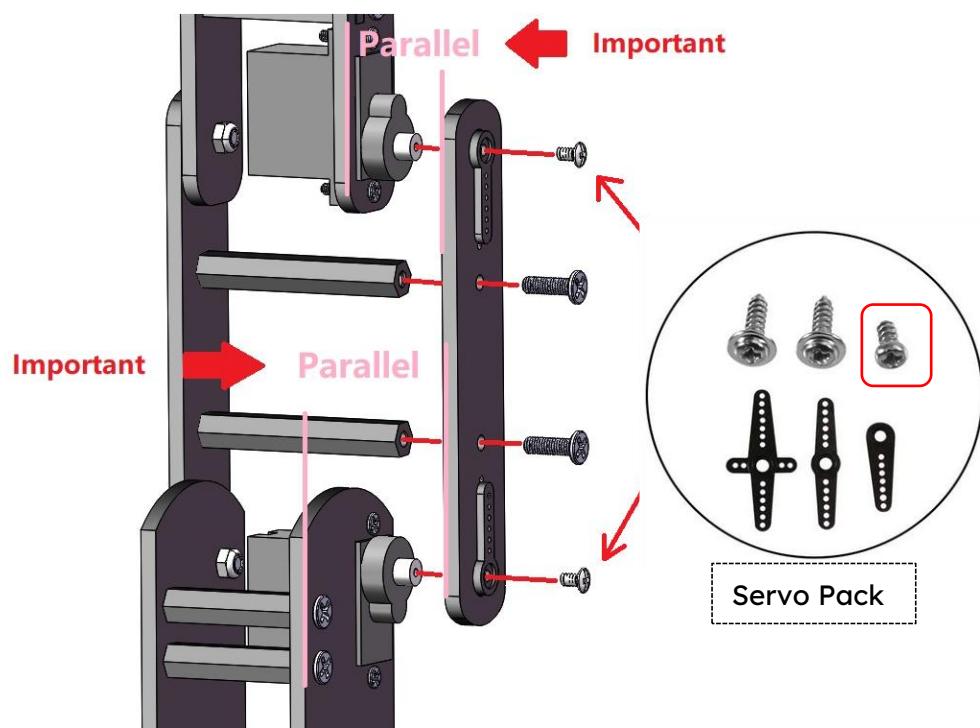
Step 24 Fix the Robot Arm to Servo A of the Base

Step 23 Structure	Step 17 Structure	M2.5X4mm Screw 1PCS
		 A red arrow points to one of the silver screws.



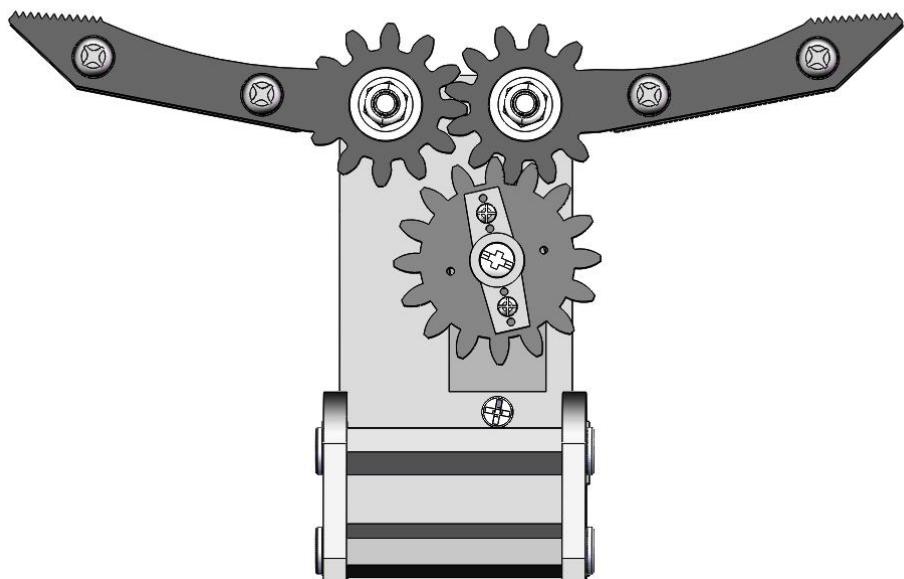
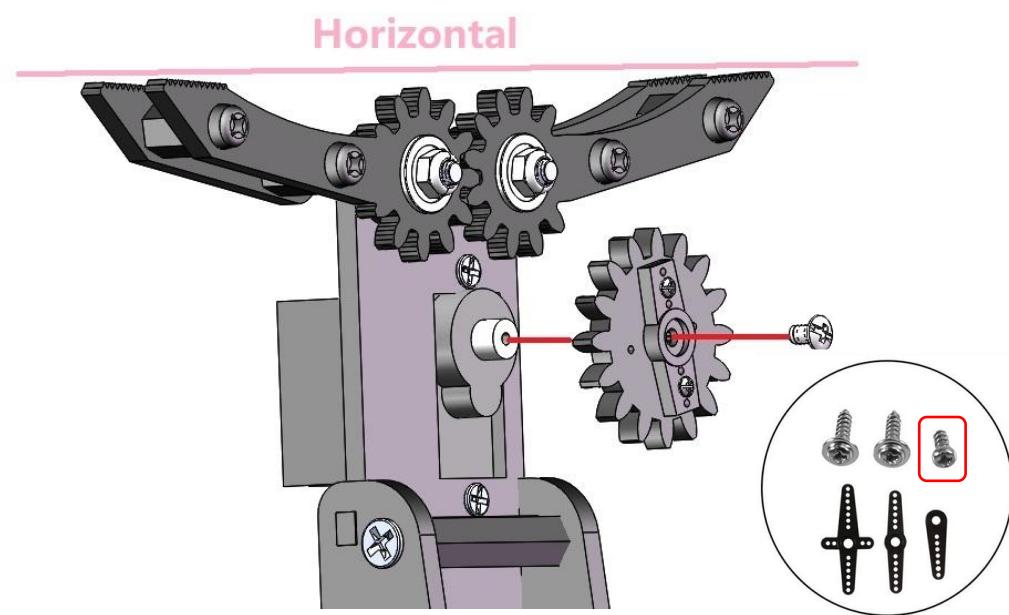
Step 25 Assemble the Middle Arm

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

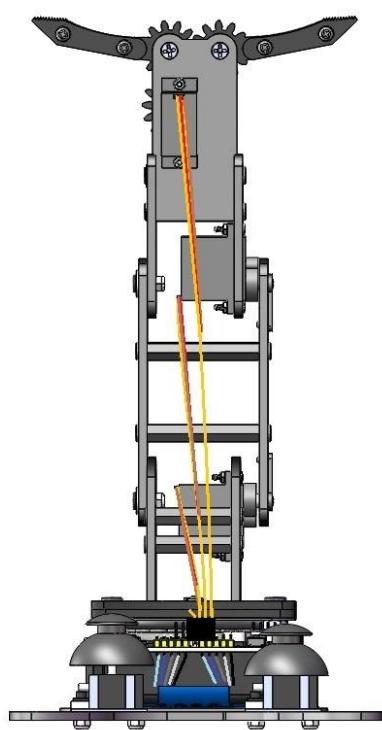
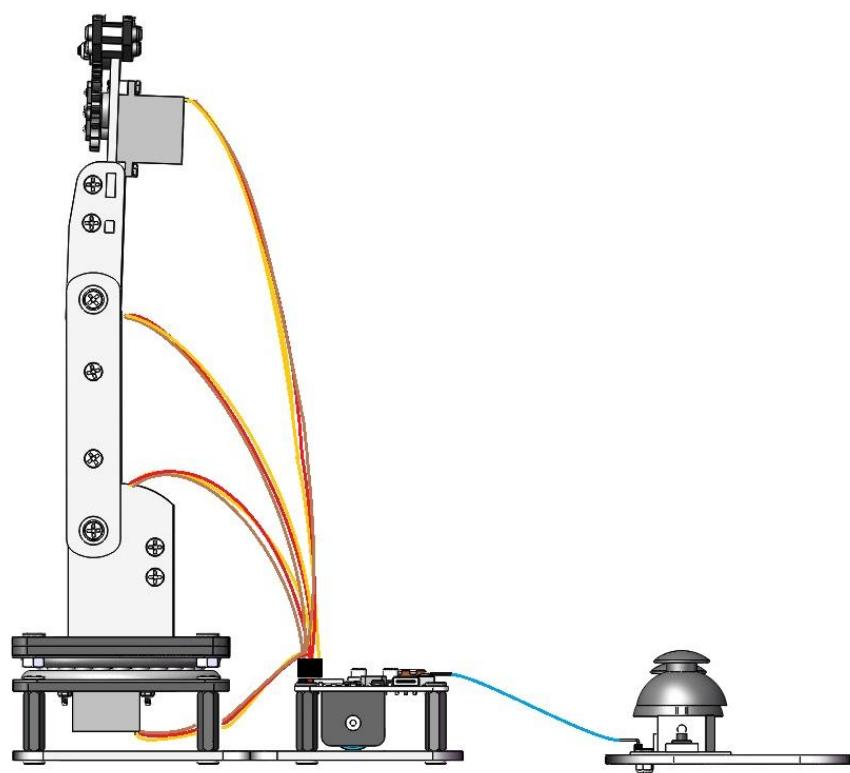


Step 26 Fix the Clip Plates to Servo D

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 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

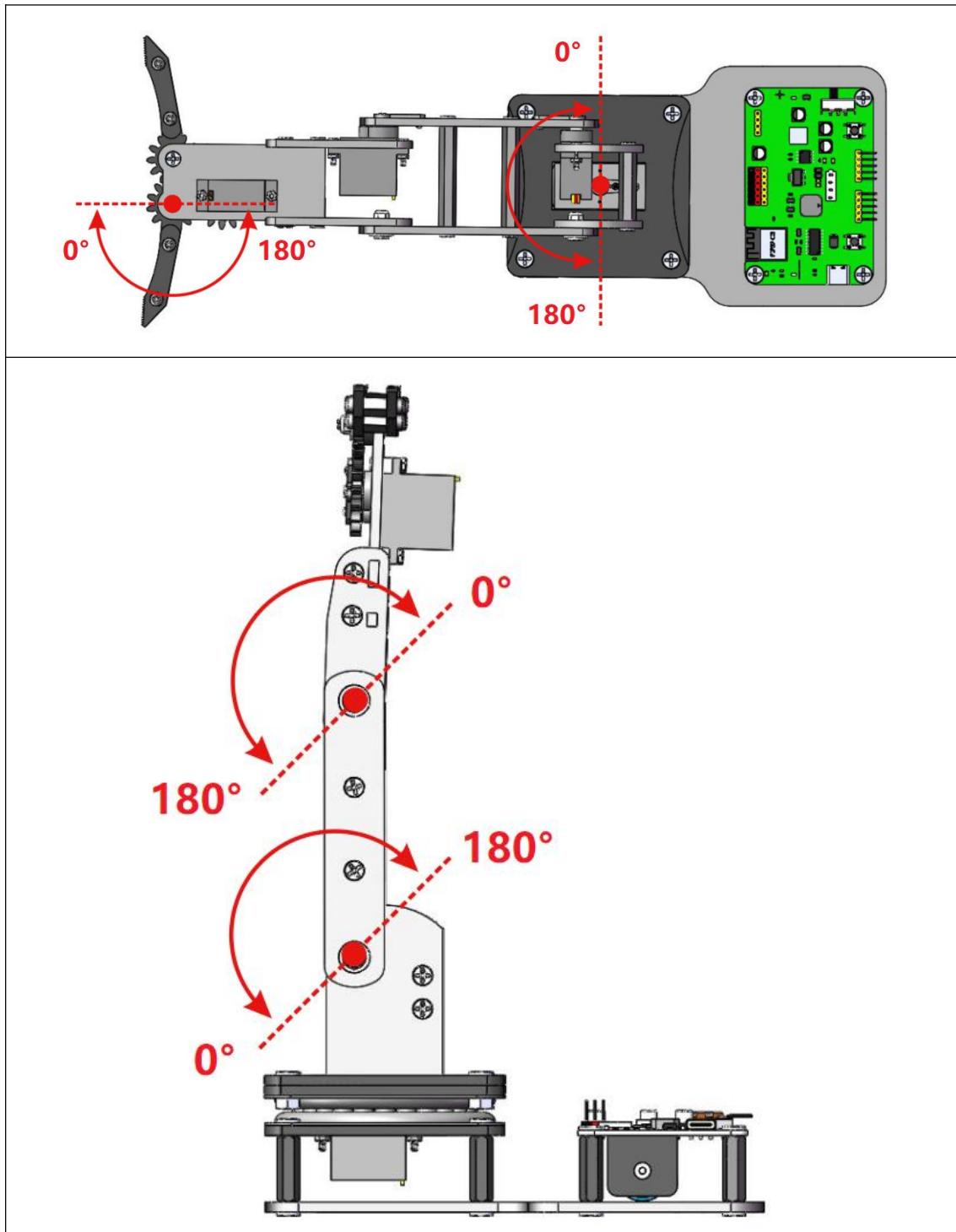
Power off the unit when not in use.

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.

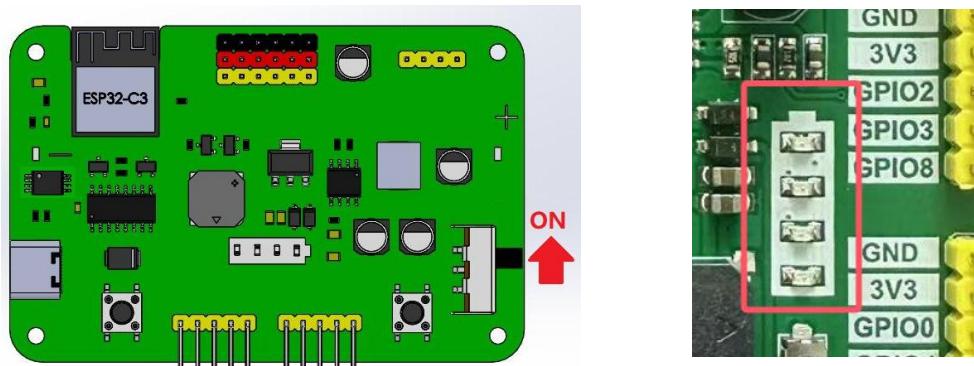
3.1 Degrees of Freedom



3.2 Control the eArm with a Joystick

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

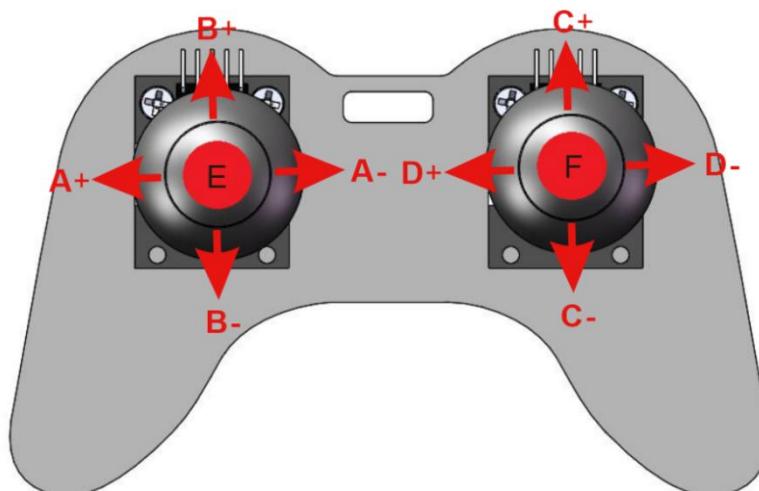
- 1: Make sure the 18650 lithium battery is installed.
- 2: The battery indicator shows 3 bars or more.



After powering on the robot, the default control mode is **Joystick Mode**. To switch between control modes, press the "**F**" button on the right joystick. Available modes include:

- Joystick Mode (default)
- Web App Mode

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)

B+: Raise rear arm (Servo B)

C+: Raise the forearm (Servo C)

D+: Open the gripper (Servo D)

A-: Rotate arm right (Servo A)

B-: Lower rear arm (Servo B)

C-: Lower the forearm (Servo C)

D-: Close the gripper (Servo D)

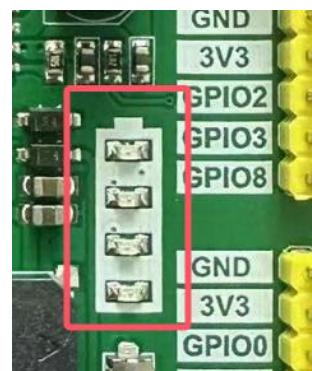
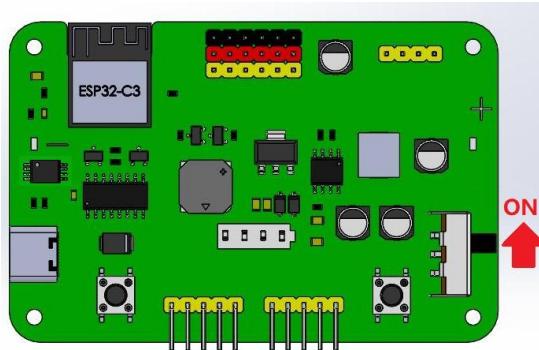
F: Toggle between Joystick and Web App control modes

E: Enable/disable the buzzer

3.3 Control the eArm with Web App

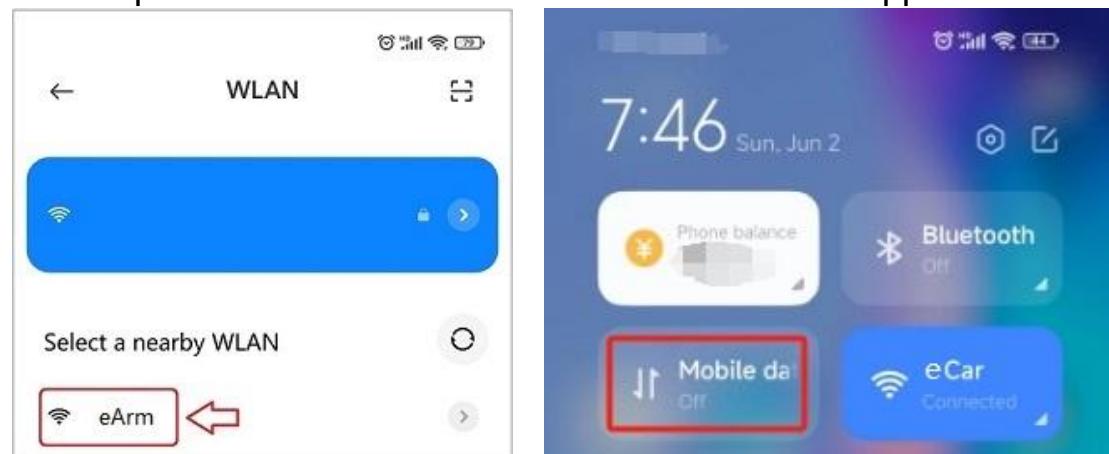
Turn the eArm's power switch to the ON position.

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

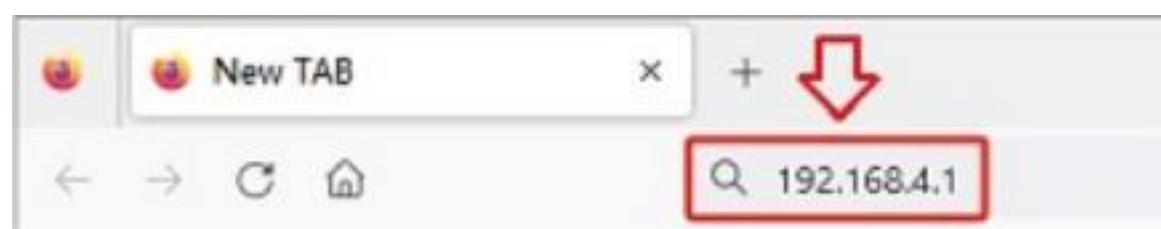


Turn on the phone's Wi-Fi and connect to the network named 'eArm'

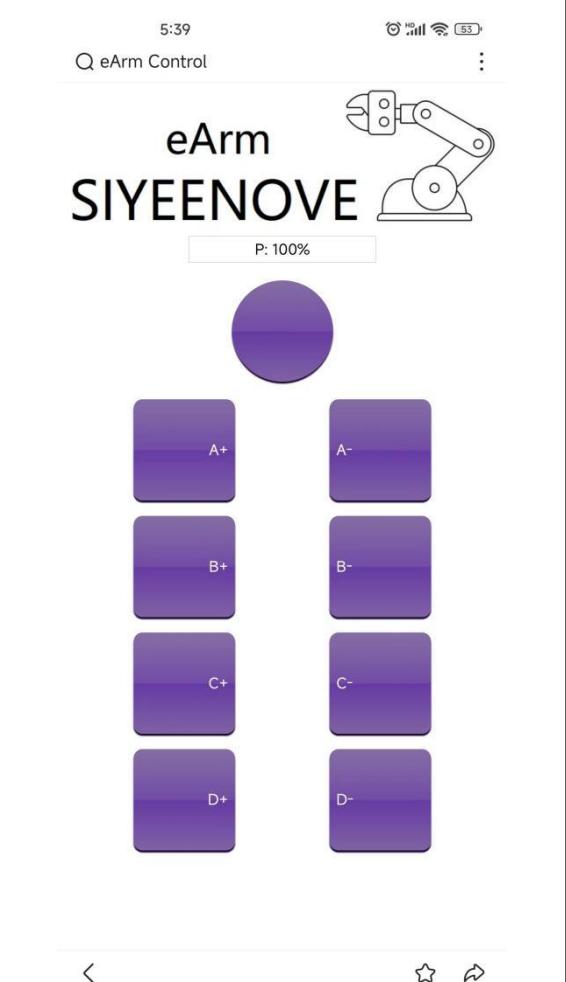
- Once connected to the Wi-Fi, you may see a 'Network connection failed' message; just disregard it.
- Step 1: Turn off mobile data in your phone settings
- Step 2: This ensures stable connection to the Web App



1. Open the web browser on your phone
2. Type 192.168.4.1 into the address bar
3. Press Enter to connect



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



The screenshot shows a mobile browser interface for 'eArm Control'. At the top, it displays the time '5:39' and signal strength. Below the header, there's a logo featuring a stylized robot arm and the text 'eArm SIYEENOVE' with a progress bar at 'P: 100%'. The main area contains a 4x2 grid of buttons. A large purple circle is positioned above the first column of buttons. The buttons are labeled as follows:

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

At the bottom of the screen, there are navigation icons: a left arrow, a star, and a right arrow.

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 upload new code.

Important notes about code uploading:

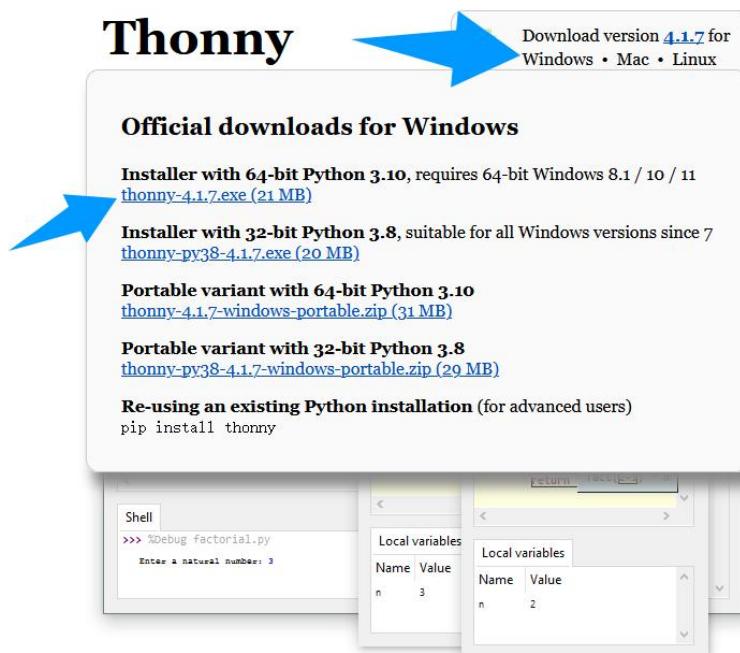
- 1: The ESP32-C3 board always executes the most newly uploaded program
- 2: Each new code upload overwrites the previous program

This section will guide you through programming the robot using code examples ranging from basic to advanced. The final example contains our factory-preloaded firmware - the same code that enables the arm functionality shown in the previous section.

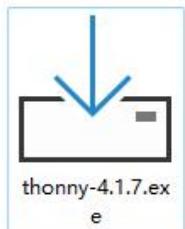
4.2 Install Thonny IDE

The latest version of Arduino IDE can be downloaded from the Arduino official website: <https://thonny.org/>

In the following we will demonstrate the installation of **Windows Win 11 and newer, 64 bits** version IDE on PC.



After the installation package is downloaded, double-click the file to install the software.

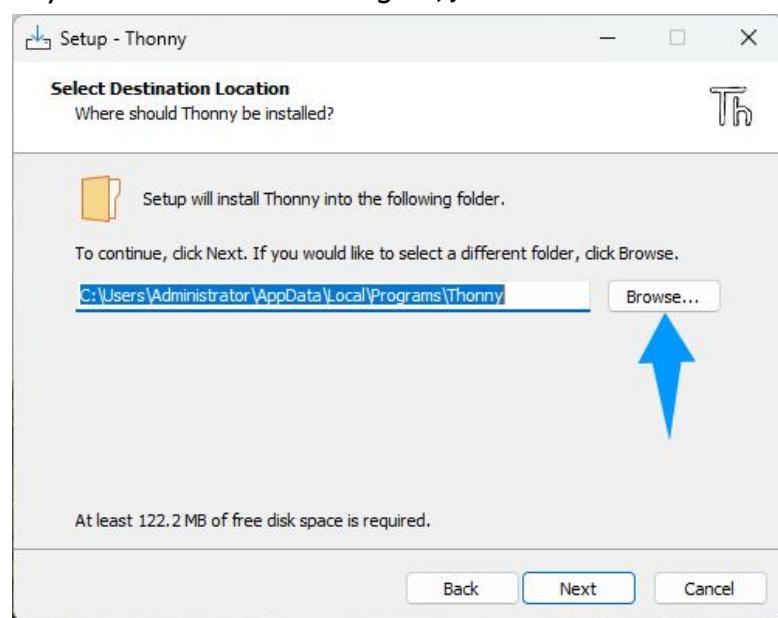


If you're not familiar with computer software installation, you can simply keep clicking "Next" until the installation completes.

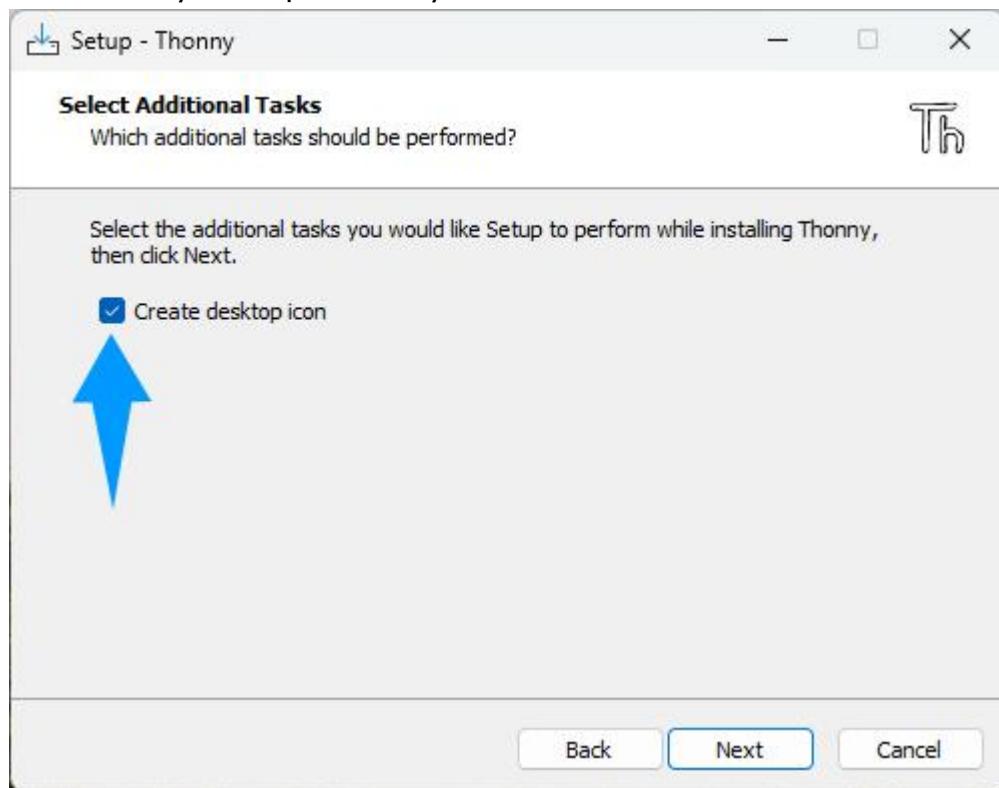


If you want to change Thonny's installation path, you can click "Browse" to modify it. After selecting installation path, click "OK".

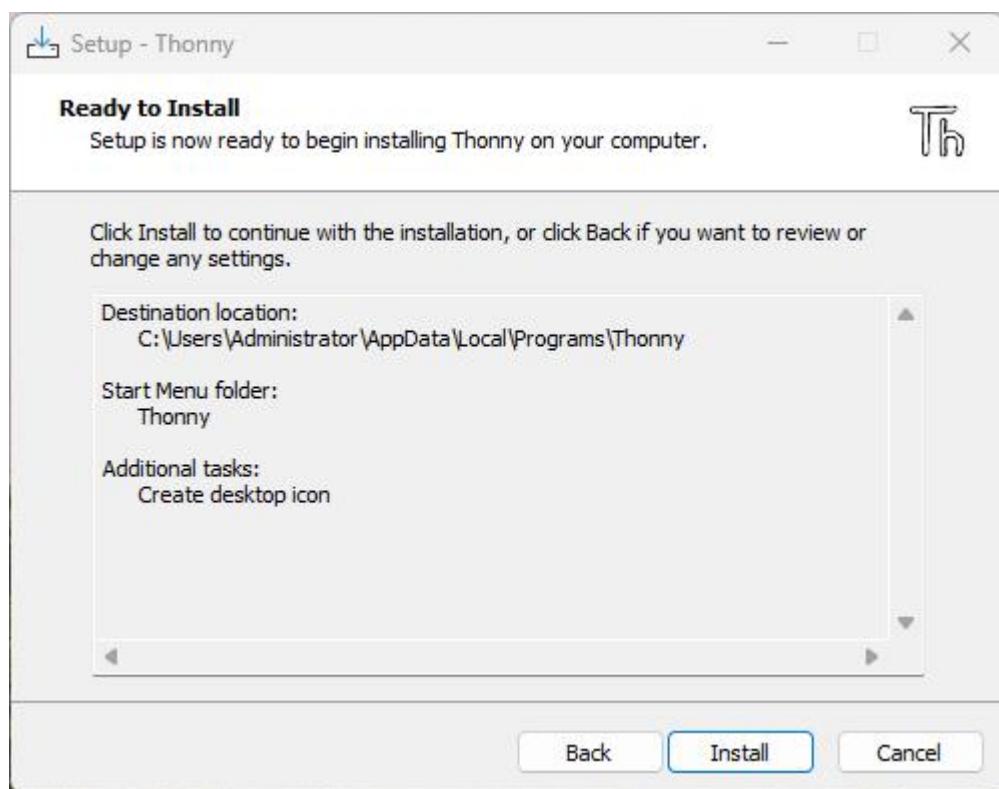
If you do not want to change it, just click "Next".



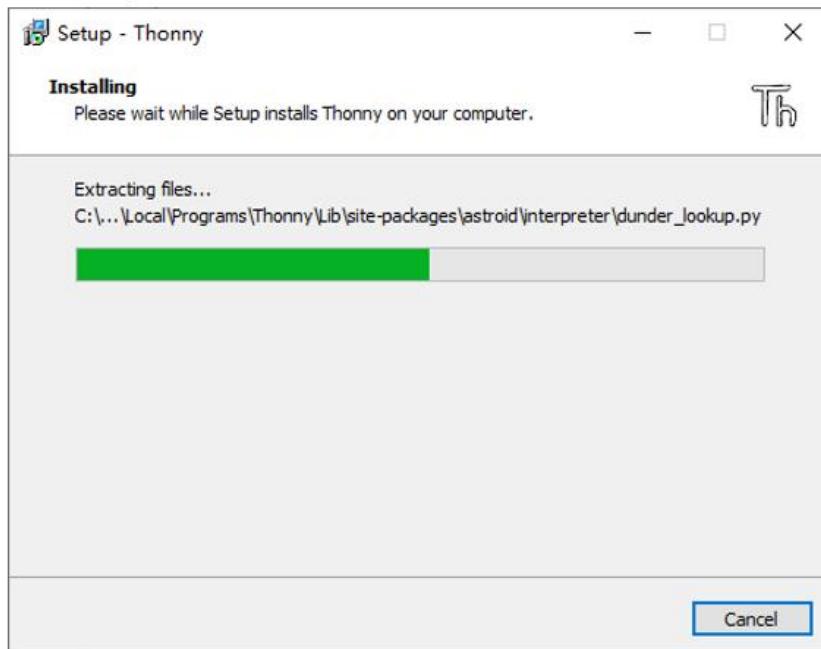
Check “Create desktop icon” and then it will generate a shortcut on your desktop to facilitate you to open Thonny later.



Click “install” to install the software.



During the installation process, you only need to wait for the installation to complete, and you must not click “Cancel”, otherwise Thonny will fail to be installed.



Once you see the interface as below, Thonny has been installed successfully.

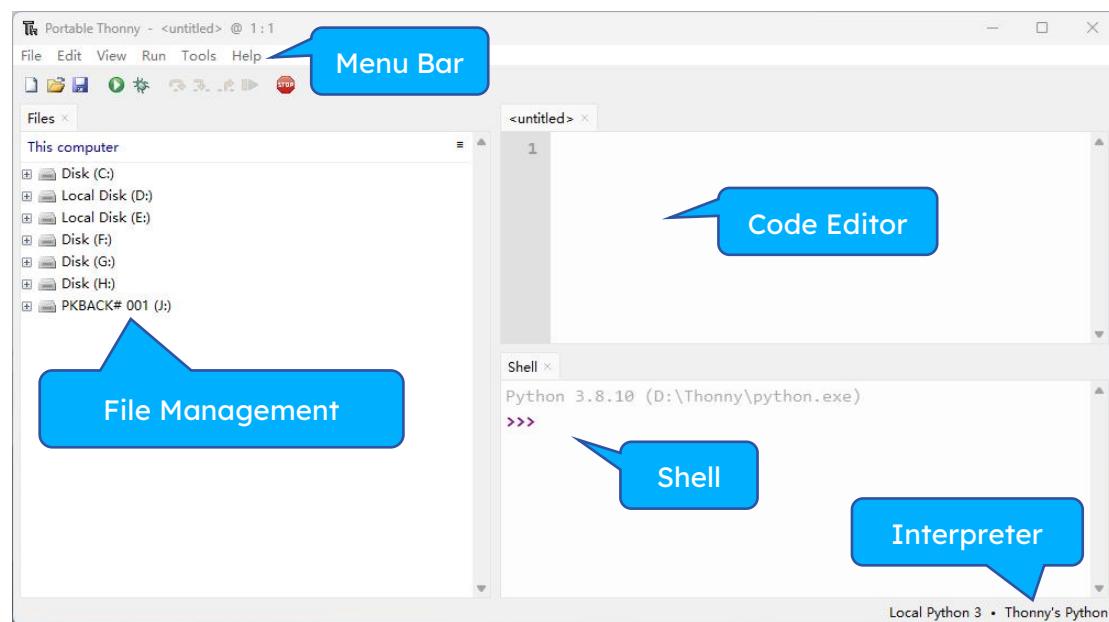
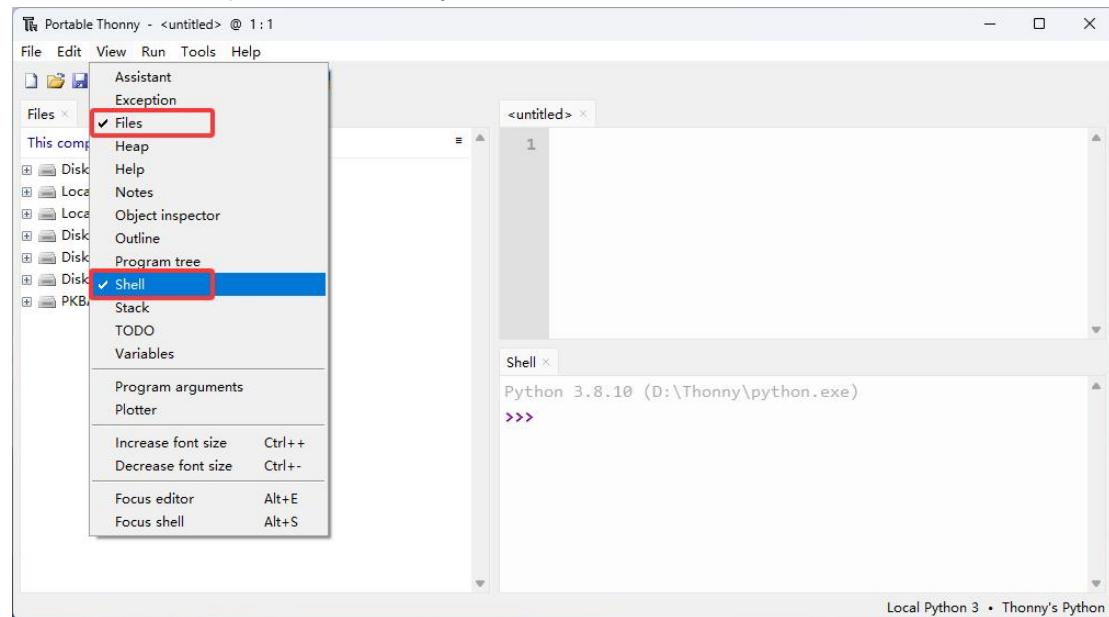


If you've checked “Create desktop icon” during the installation process, you can see the below icon on your desktop.



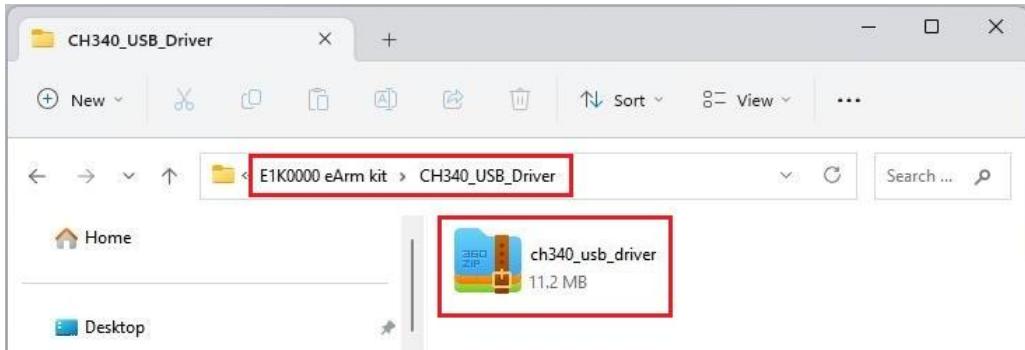
4.2 Basic Configuration of Thonny

Click the desktop icon of Thonny and select “View” -> “Files” and “Shell”.



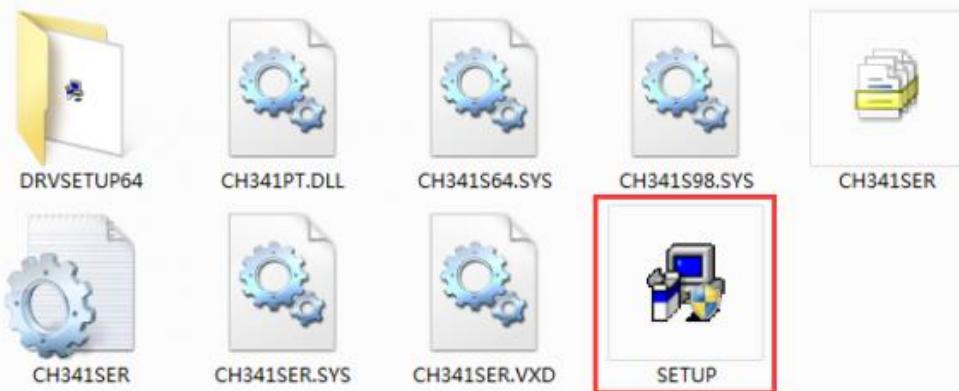
4.3 Install CH340 USB driver

Driver file is provided in our tutorial package:

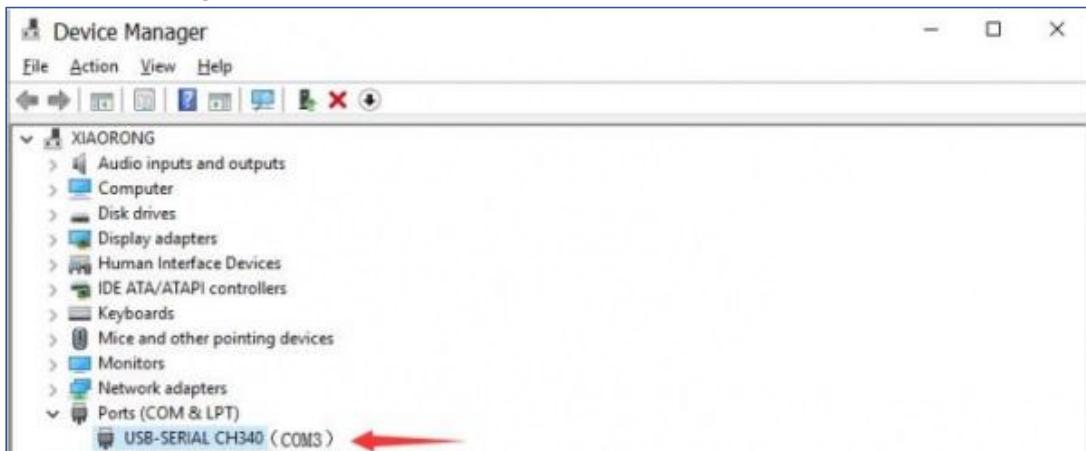


4.3.1 Install the Driver on Windows system

Unzip the file we provided and double-click “SETUP” to run the installer:



If the driver is successfully installed, when the esp32 board is connected to the computer through the USB cable, under the path “Computer” -> “Properties” -> “Device manager”, you can find “USB-SERIAL CH340 (COM3)”, as shown 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.

4.3.2 Install the Driver on Linux system

Drivers are almost certainly built into your Linux kernel already and it will probably just work as soon as you plug it in. If not you can download the Linux CH340 Driver (but I'd recommend just upgrading your Linux install so that you get the "built in" one).

If you must install it by yourself, check the "**README.md**" file in the zip package.



4.3.3 Install the Driver on MAC system

Link to download the driver:

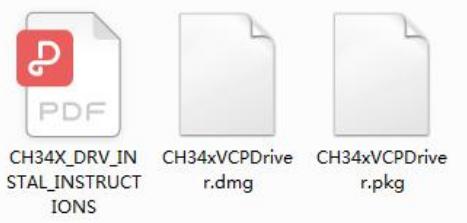
http://www.wch-ic.com/downloads/CH341SER_EXE.html

[download](#)

relation files

file name	file content
CH341SER.ZIP	CH340/CH341 USB to serial port Windows driver, supports Windows XP/Vista/7/8/8.1/10/11/SERVER 2003/2008/2012/2016/2019/2022 -32/64bit, Microsoft WHQL Certified, supports USB to 3-line and 9-line serial port.
CH341SER_LINUX.ZIP	Linux driver for USB to serial port, supports CH340 and CH341, supports 32/64-bit operating systems.
CH341SER_MAC.ZIP	For CH340/CH341/CH342/CH343/CH344/CH347/CH9101/CH9102/CH9103/CH9104/CH9143, USB to serial port VCP vendor driver of macOS , supports OS X 10.9~10.15 , OS X 11 (Big Sur) and above , contains installation guide documents.
CH340DS1.PDF	CH340 datasheet, USB bus converter chip which converts USB to serial port, printer port, IrDA SIR etc. Integrated clock, supports various operating systems, chip information can be customized, this datasheet is about USB to serial port and USB to Infrared Adapter SIR.
CH341DS1.PDF	CH341 datasheet, USB bus converter chip with multiple communication interfaces, such as USB to serial port/parallel port/printer port/I2C interface etc. Drivers support for Windows/Linux/Android/Mac, etc. The datasheet is the description of USB to serial port and printer port.
CH341SER_ANDROID.ZIP	CH340/CH341/CH342/CH343/CH344/CH347/CH9101/CH9102/CH9103/CH9104/CH9143 USB to serial port Android driver-free application library and software, for Android OS 4.4 and above in USB Host mode, without loading Android kernel driver and without root access operation. Includes apk installer, lib library file (Java Driver), App Demo routines (USB to UART Demo project SDK).

Download the driver from the website and unzip the file to a local installation directory. You will get a PDF installation guide and two installation packages in different formats. For details, see the PDF installation guide.



4.4 Burning Micropython Firmware (Important)

To run Python programs on ESP32-C3, we need to burn a firmware to ESP32-C3 first.

4.4.1 Download MicroPython Firmware

Official website of microPython:

<http://micropython.org/>

Webpage listing firmware of microPython for ESP32-C3:

https://micropython.org/download/ESP32_GENERIC_C3/

Firmware

Releases

v1.27.0 (2025-12-09) .bin / [.app-bin] / [.elf and .map] / [Release notes] (latest)
v1.26.1 (2025-09-11) .bin / [.app-bin] / [.elf and .map] / [Release notes]
v1.26.0 (2025-08-09) .bin / [.app-bin] / [.elf and .map] / [Release notes]
v1.25.0 (2025-04-15) .bin / [.app-bin] / [.elf and .map] / [Release notes]
v1.24.1 (2024-11-29) .bin / [.app-bin] / [.elf and .map] / [Release notes]
v1.24.0 (2024-10-25) .bin / [.app-bin] / [.elf and .map] / [Release notes]
v1.23.0 (2024-06-02) .bin / [.app-bin] / [.elf and .map] / [Release notes]
v1.22.2 (2024-02-22) .bin / [.app-bin] / [.elf and .map] / [Release notes]
v1.22.1 (2024-01-05) .bin / [.app-bin] / [.elf and .map] / [Release notes]
v1.22.0 (2023-12-27) .bin / [.app-bin] / [.elf and .map] / [Release notes]
v1.21.0 (2023-10-05) .bin / [.app-bin] / [.elf and .map] / [Release notes]
v1.20.0 (2023-04-26) .bin / [.elf and .map] / [Release notes]
v1.19.1 (2022-06-18) .bin / [.elf and .map] / [Release notes]
v1.18 (2022-01-17) .bin / [.elf and .map] / [Release notes]
v1.17 (2021-09-02) .bin / [.elf and .map] / [Release notes]

Preview builds

These are automatic builds of the development branch for the next release.

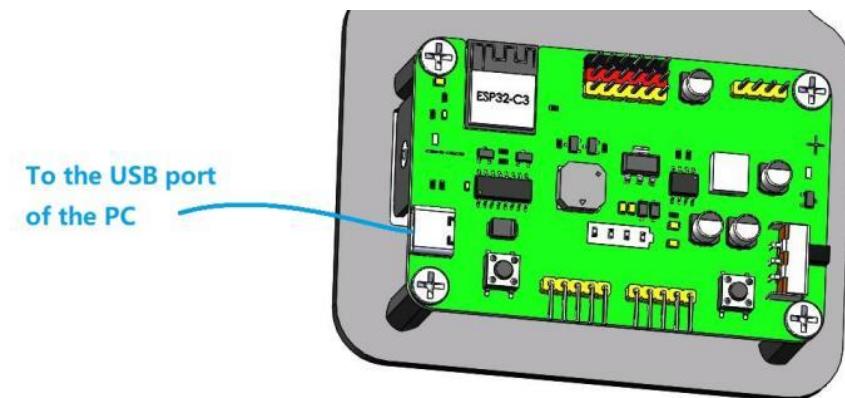
v1.28.0-preview.37.g610552010d (2026-01-02) .bin / [.app-bin] / [.elf] / [.map]
v1.28.0-preview.18.g6341258207 (2025-12-19) .bin / [.app-bin] / [.elf and .map]
v1.28.0-preview.4.gef567dc928 (2025-12-16) .bin / [.app-bin] / [.elf and .map]
v1.27.0-preview.503.g2bd337ef18 (2025-12-08) .bin / [.app-bin] / [.elf and .map]

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

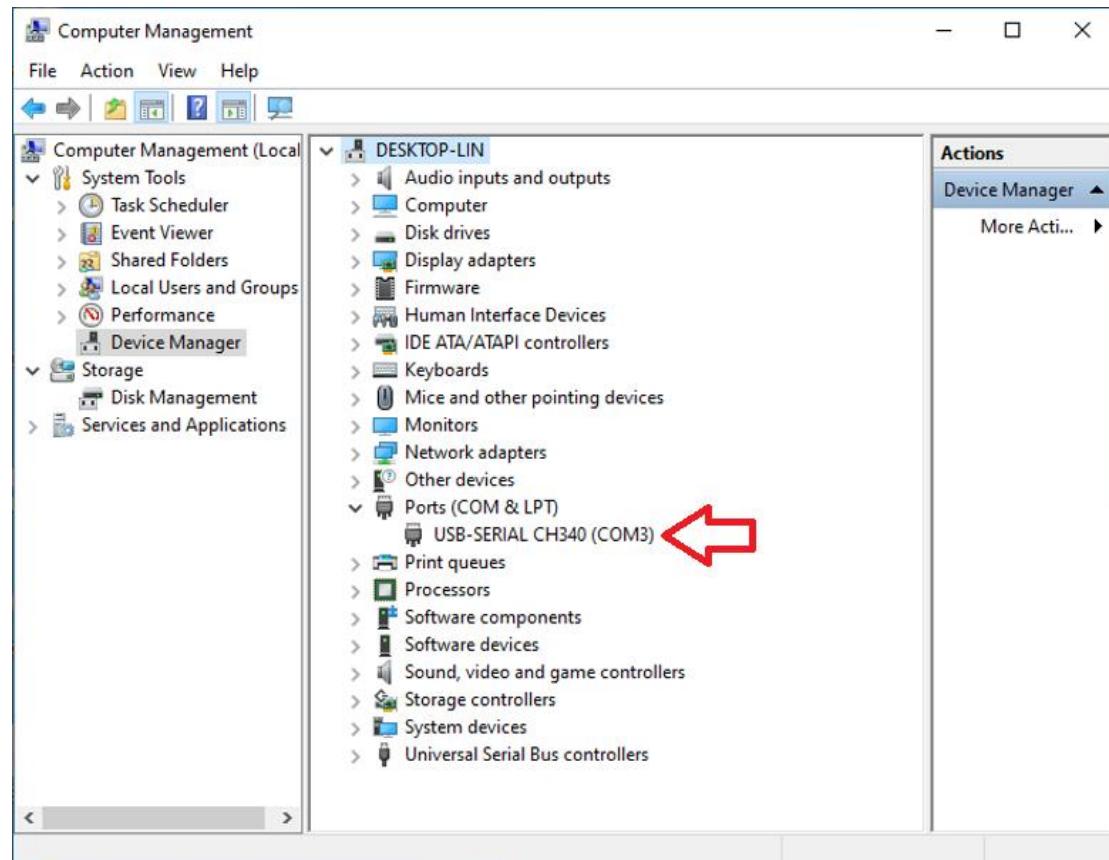
This file is also provided in our tutorial folder : “**MicroPython > Firmware**”.

4.4.2 Burn MicroPython Firmware

Use the USB cable to connect the PC and eArm, as shown below:

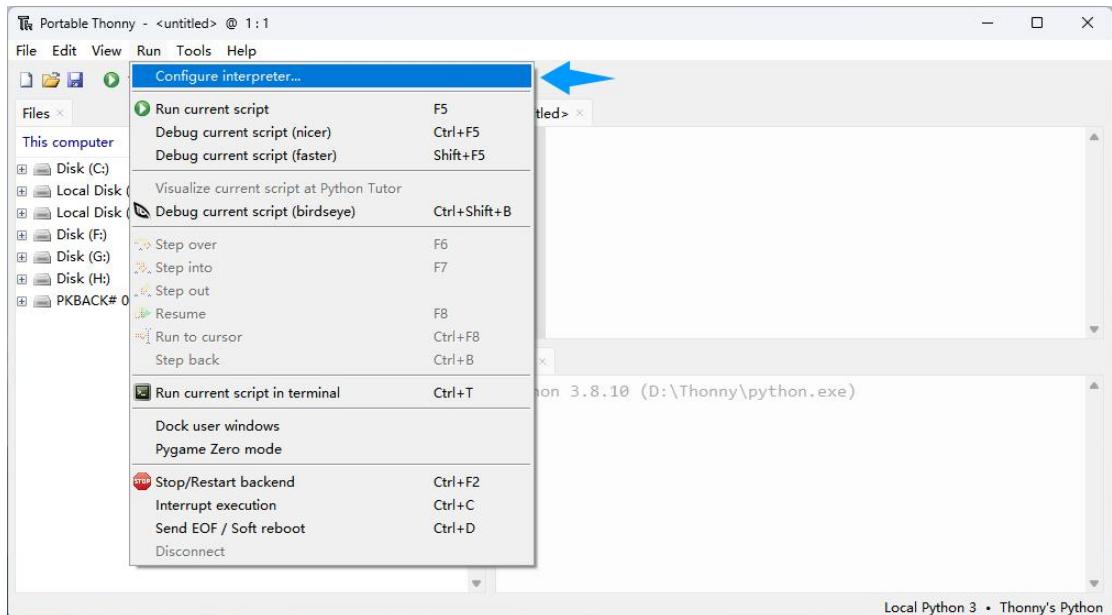


Make sure that the driver has been installed successfully and that it can recognize COM port correctly. Open device manager and expand “Ports”.

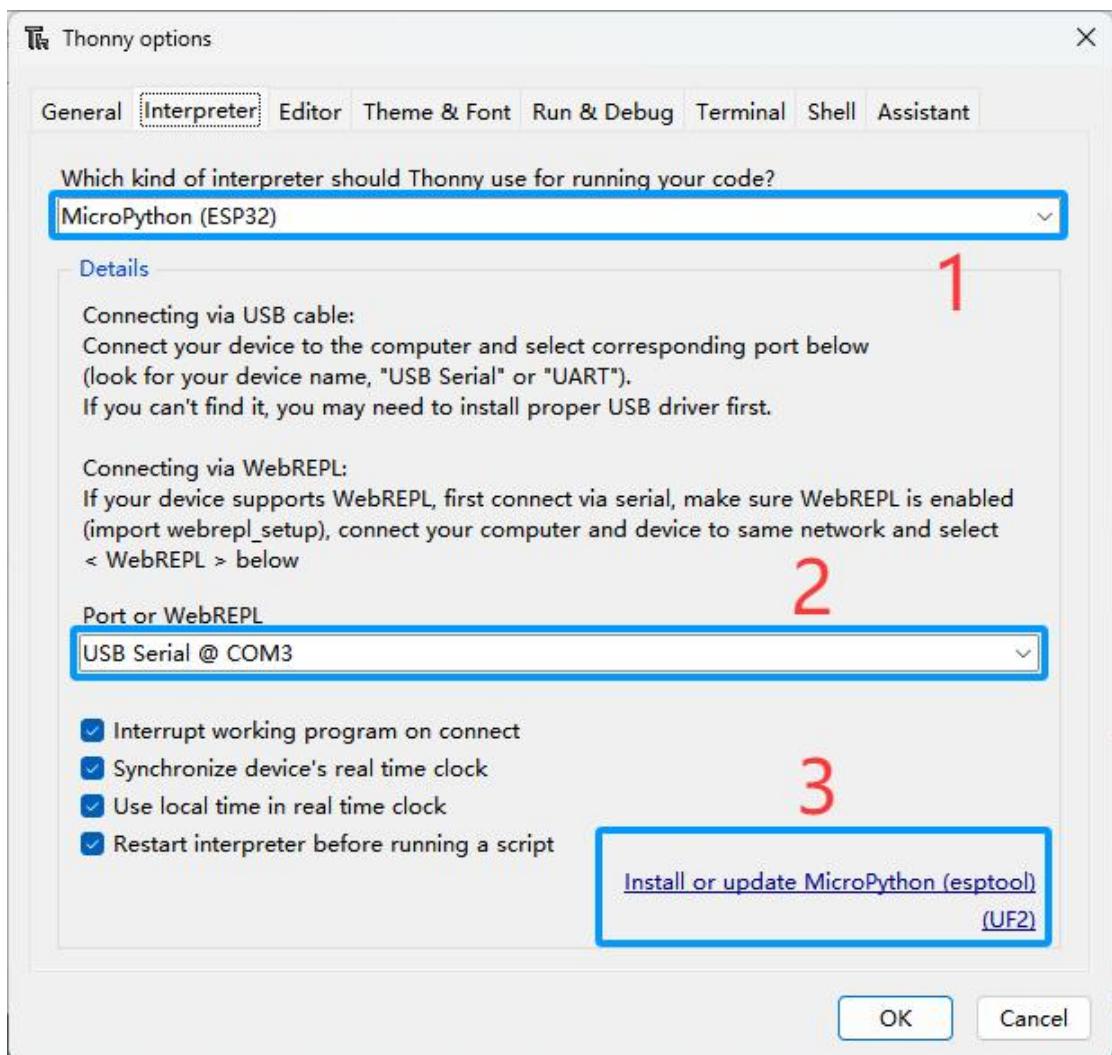


Note: Driver display symbol is “**USB-SERIAL CH340 (COMx)**”, “x” can be any number, it depends on what number your computer assigns to the ESP32 device.

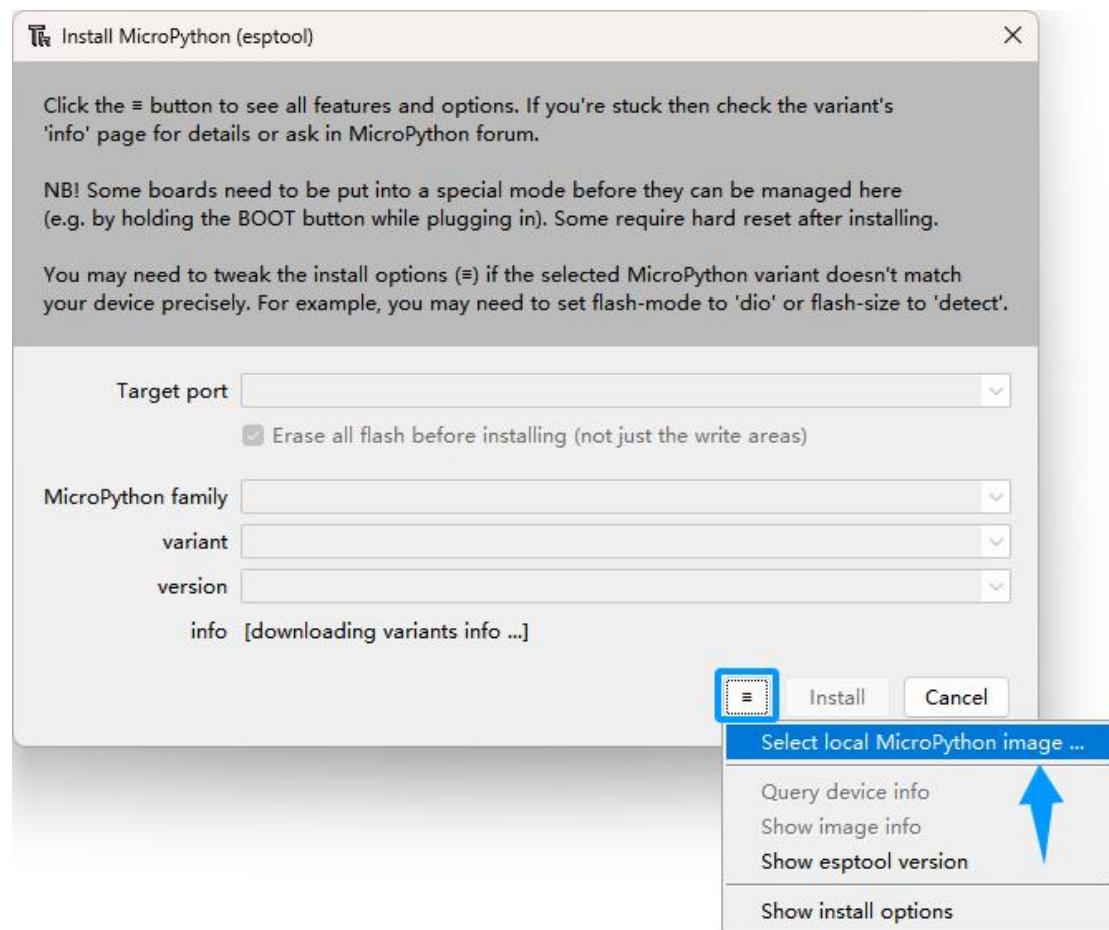
1. Open Thonny, click “Run” and select “Configure interpreter...”



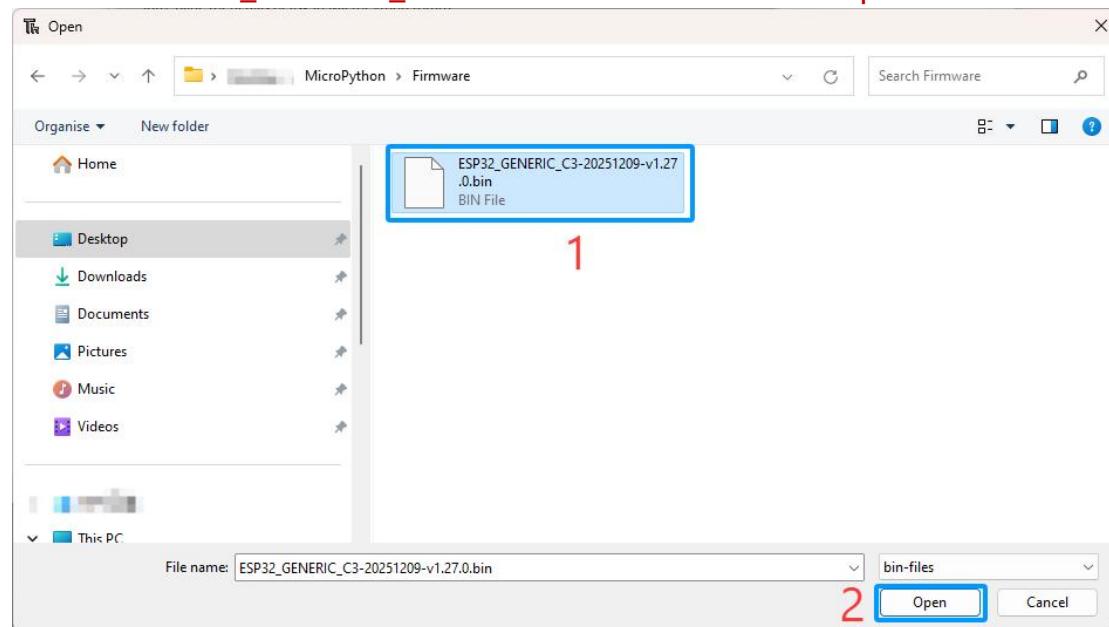
2. Select “Micropython (ESP32)”, select “USB Serial @ (COM3)”, and then click the long button under “Install or update MicroPython(esptool)(UF2)”.



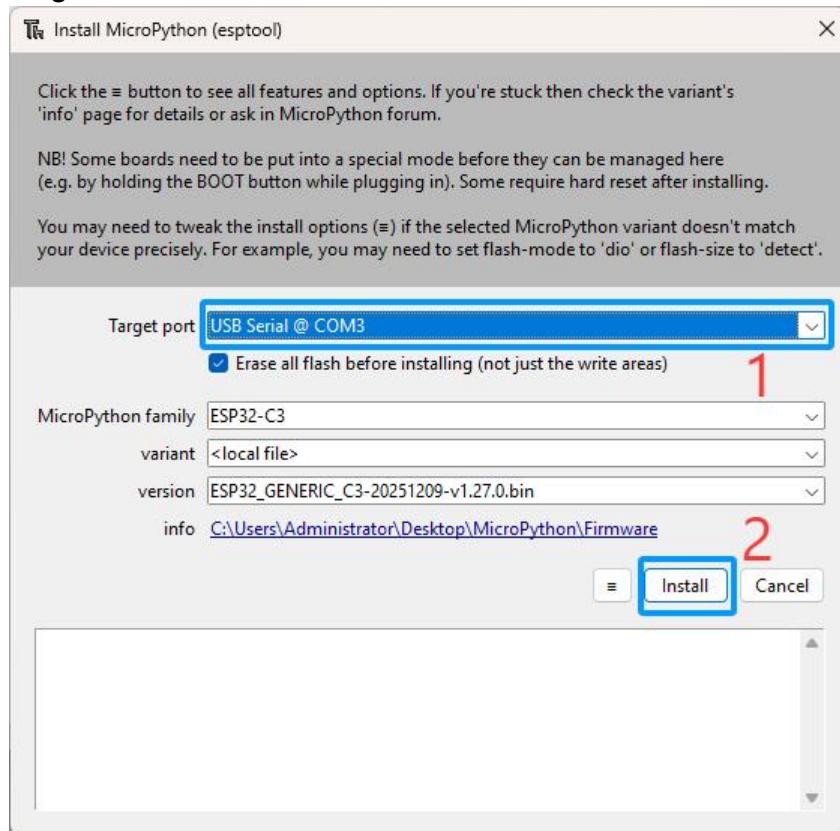
3. The following dialog box pops up. Select “Select local MicroPython image...”.



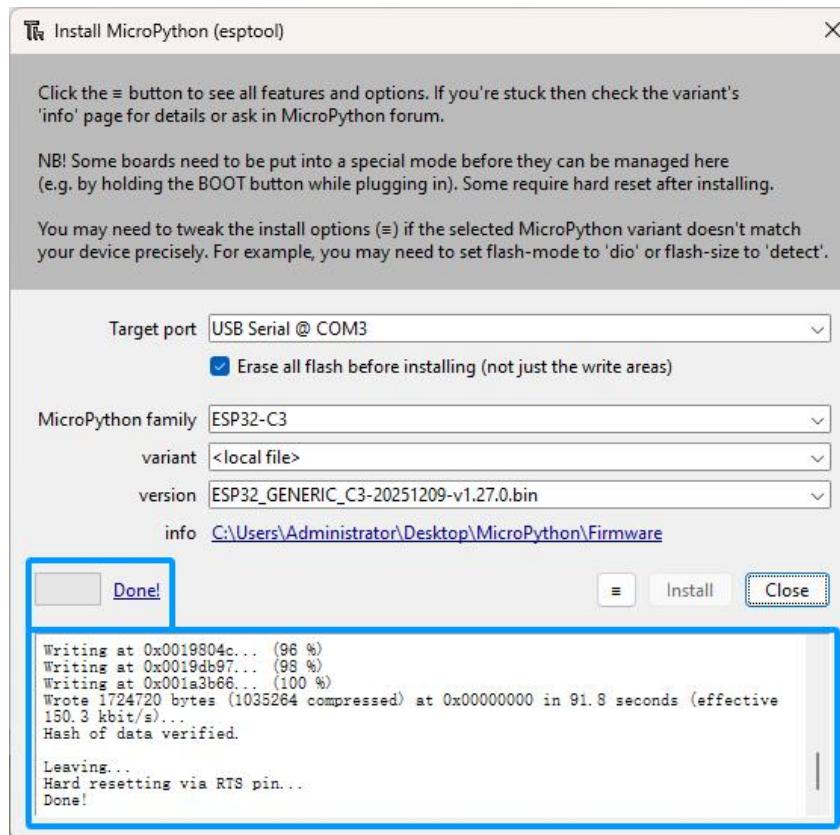
4. The following dialog box pops up. Select the previous prepared microPython firmware “[ESP32_GENERIC_C3-20251209-v1.27.0.bin](#)”. Click “Open”.



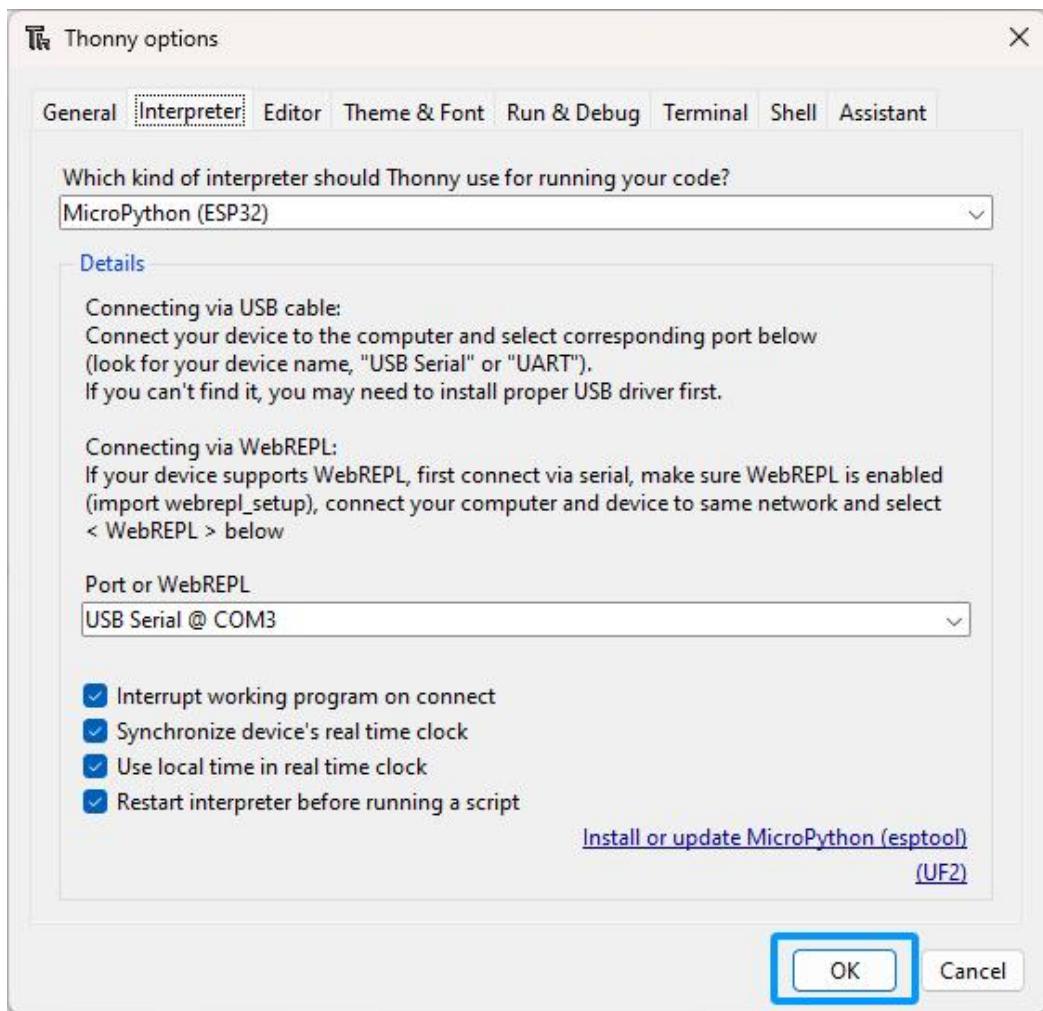
5. Select “**Micropython (ESP32)**”, select “**USB Serial @ (COM3)**”, and then click the long button under “**Install**”.



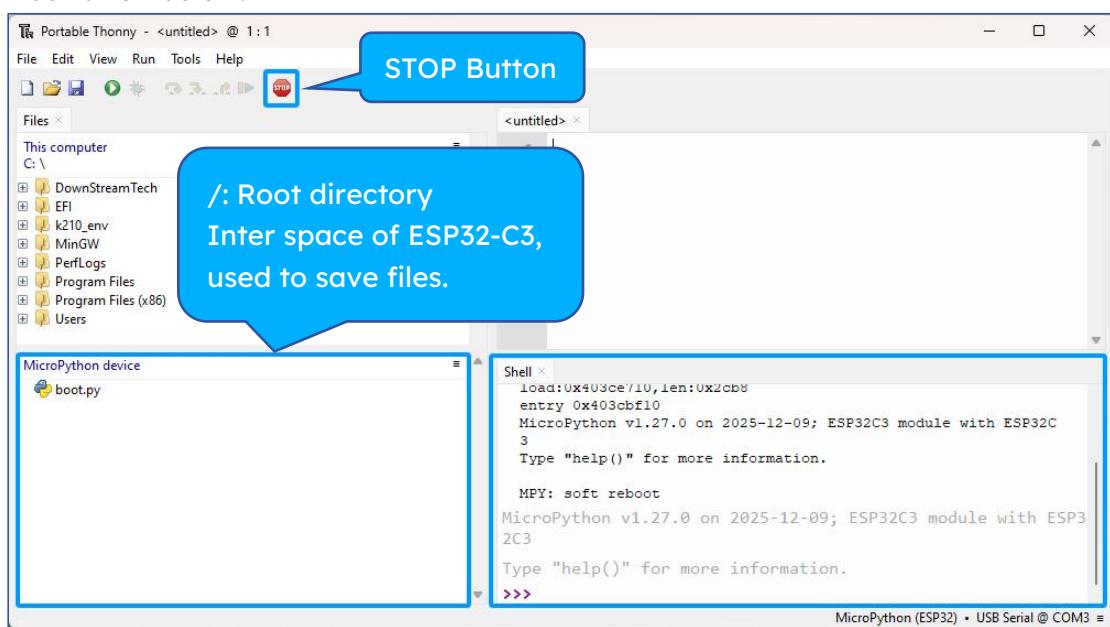
6. Wait for the installation to be done, and then click “**Close**”.



7. Click "OK".



8. Close all dialog boxes, turn to main interface and click "STOP". As shown in the illustration below:

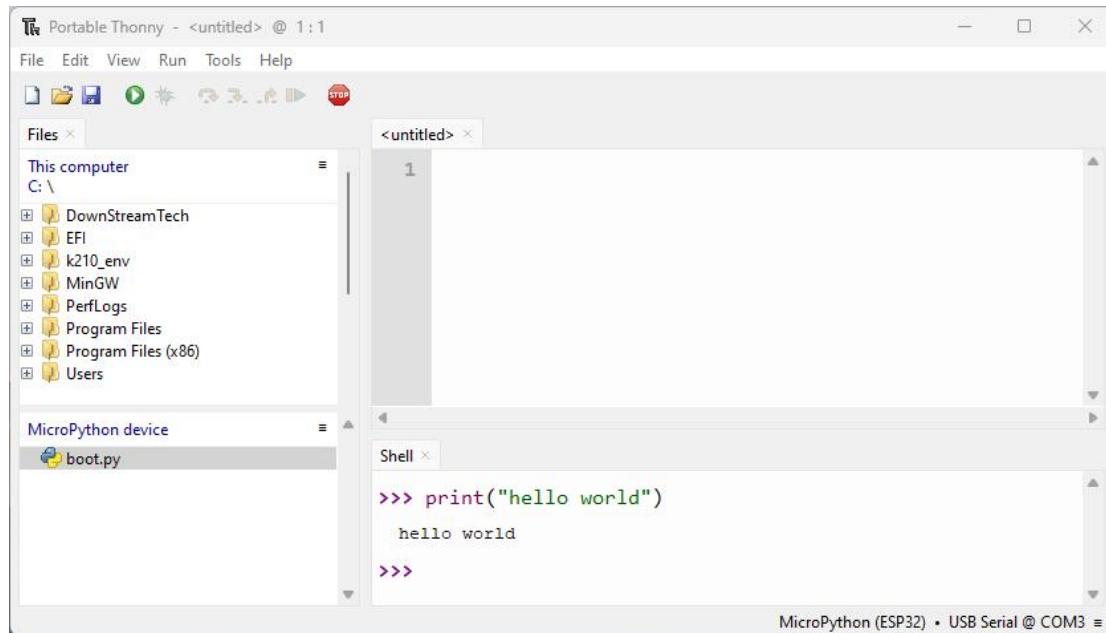


9. So far, all the preparations have been made.

4.5 Basic Usage of Thonny

4.5.1 Test Shell commands

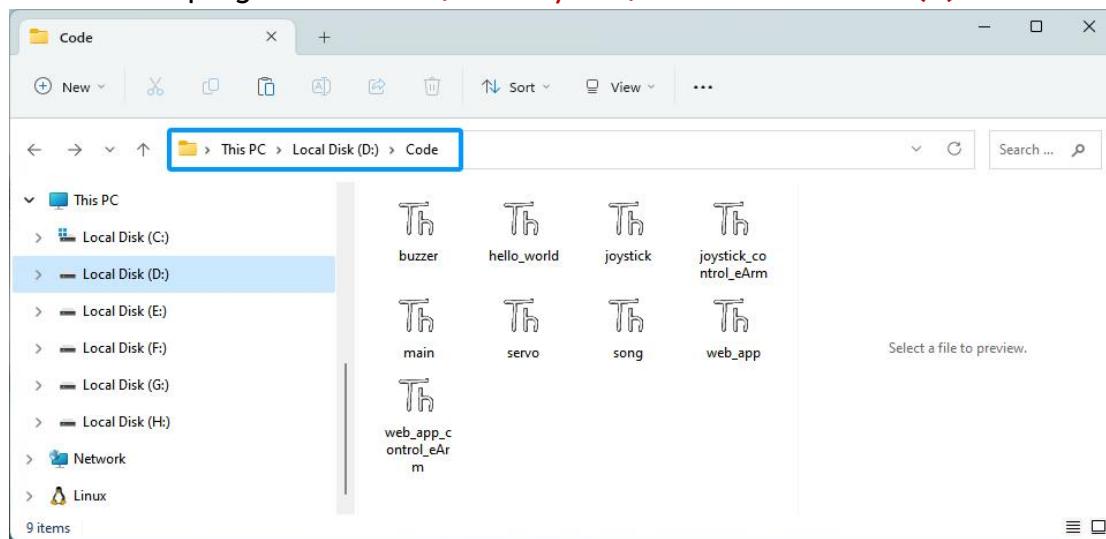
Enter “`print("hello world")`” in “Shell” and press Enter.



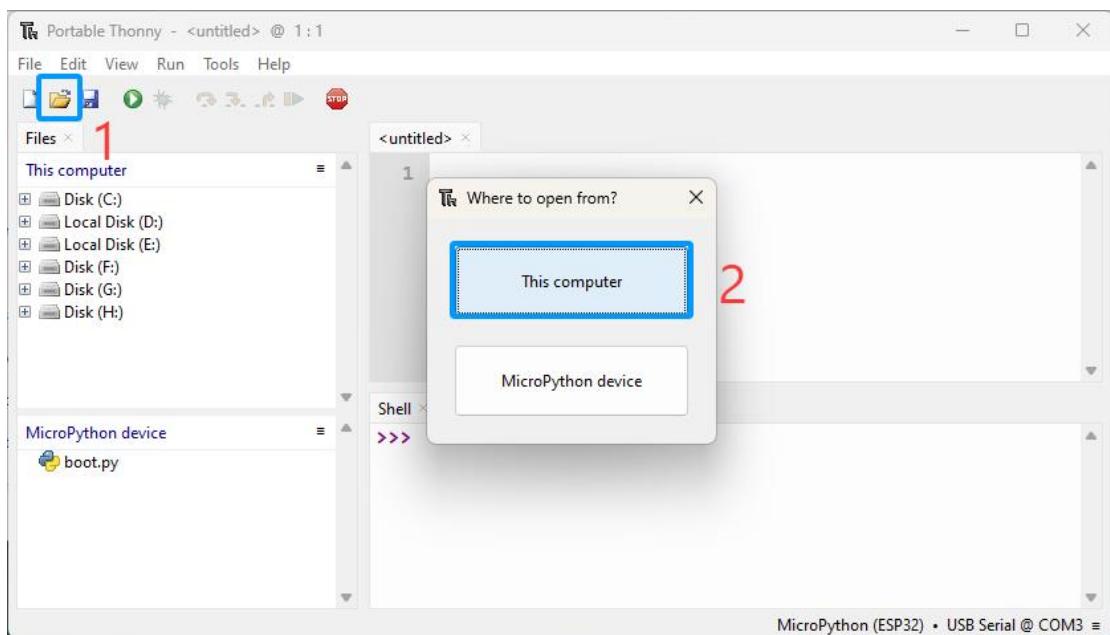
4.5.2 Run Online

ESP32 needs to be connected to a computer when it is run online. Users can use Thonny to writer and debug programs.

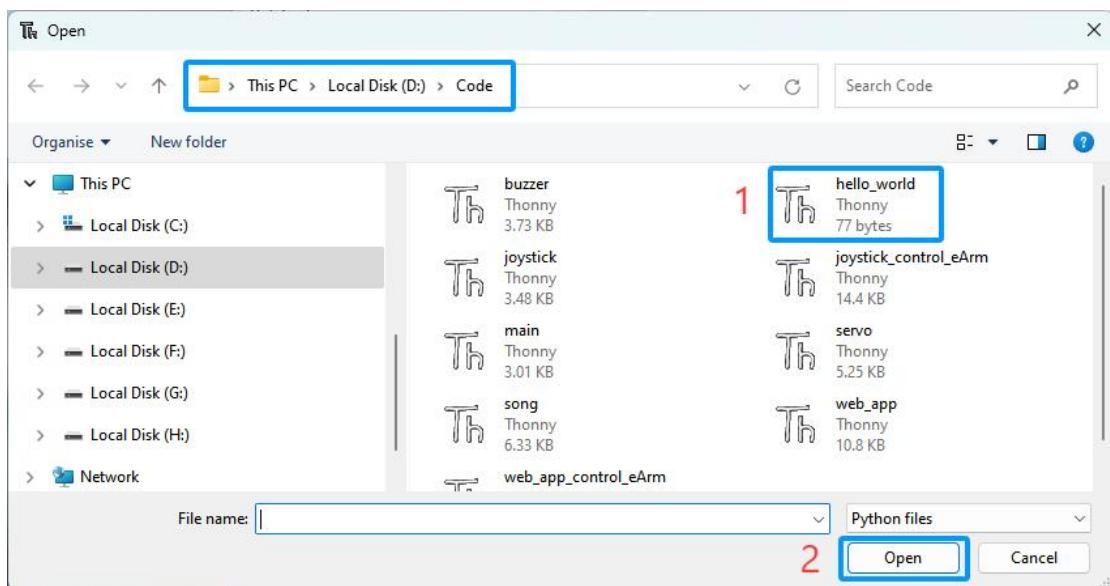
Advance the program folder “`... /MicroPython/Code`” Move to disk (D).



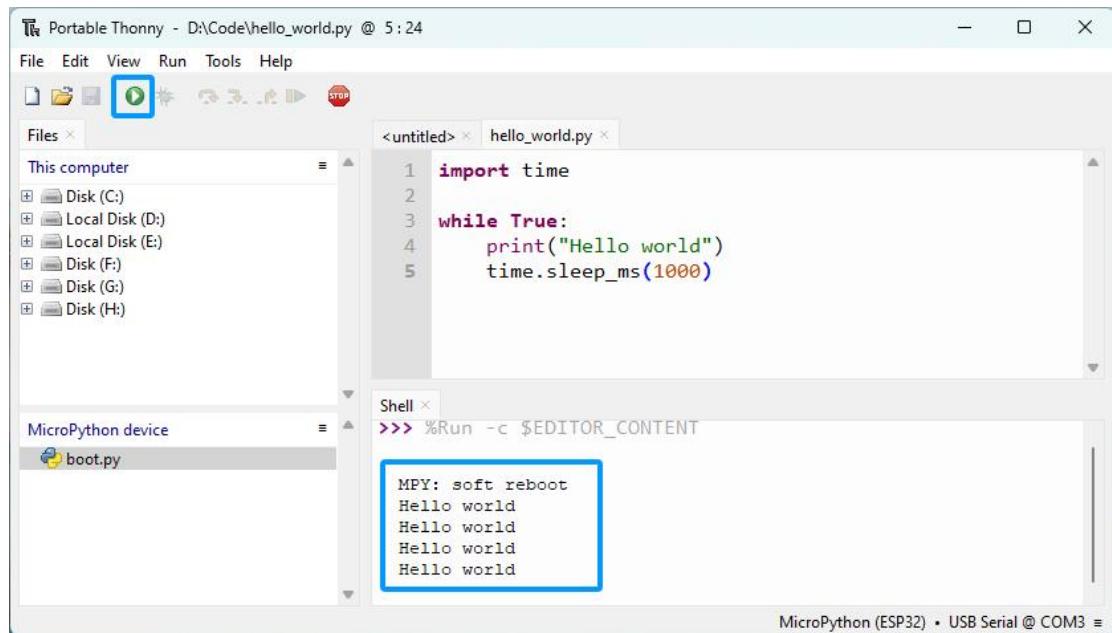
Open “Thonny”, click “Open...”, then click “This computer”.



In the new dialog box, select “hello_world.py” in “D:\Code” folder.



Click “Run current script” to execute the program and “hello world” will be printed in “Shell” .



4.5.3 Run Offline

After ESP32 is reset, it runs the file **boot.py** in root directory first and then runs file **main.py**, and finally, it enters “Shell”. Therefore, to make ESP32 execute user’s programs after resetting, we need to add a guiding program in **main.py** to execute user’s code.

main.py

```
"""
This is a Python script executor that will:
1. Scan all files in the current directory
2. Skip the boot.py and main.py files
3. Only execute files with .py extensions
4. Ensure only files are executed (not directories)
"""

#!/opt/bin/lv_micropython
# Import MicroPython specific modules
import uos as os          # MicroPython's os module
import uerrno as errno      # MicroPython's errno module
```

```

# Get directory iterator for listing files in current directory
# os.ilistdir() returns an iterator of directory entries
iter = os.ilistdir()

# File type constants for directory entries
# These constants come from stat
# module but MicroPython uses different approach
IS_DIR = 0x4000      # Flag indicating the entry is a directory
IS_REGULAR = 0x8000   # Flag indicating the entry is a regular file

# Main loop: Process each entry in the directory
while True:
    try:
        # Get next directory entry using iterator
        # Each entry is a tuple: (name, type, inode[, size])
        entry = next(iter)

        # Extract filename (first element of tuple)
        filename = entry[0]

        # Extract file type (second element of tuple)
        # This indicates if it's a file, directory, etc.
        file_type = entry[1]

        # Skip system files that should not be executed
        # boot.py: Runs on boot, should not be executed manually
        # main.py: Main application file, should be run separately
        if filename == 'boot.py' or filename == 'main.py':
            continue # Skip to next file

    else:
        # Print separator for visual clarity in output
        print("=====")

        # Print filename without newline (end="")
        print(filename, end="")

        # Check if current entry is a directory
        if file_type == IS_DIR:
            print(", File is a directory")
            print("=====")
            # Note: Directories are only identified, not processed further

    else:

```

```

# For regular files, just close the filename line
print("\n=====")

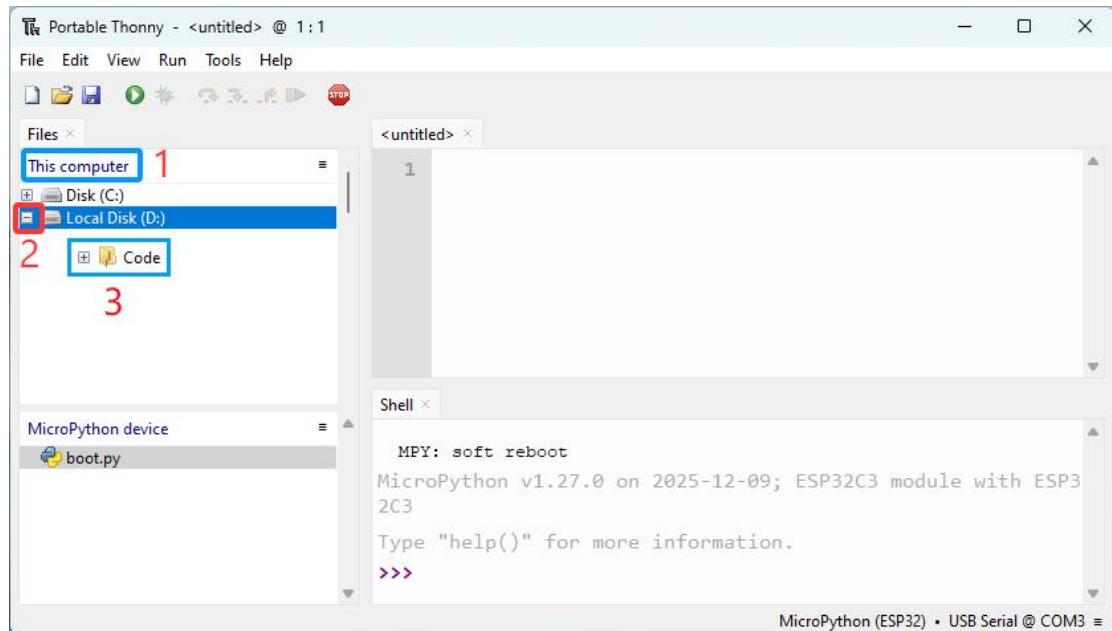
# Originally commented code that would print file contents:
# print("Contents:")
# with open(filename) as f:
#     for line in enumerate(f):
#         print("{}{}".format(line[1]),end="")
#     print("")

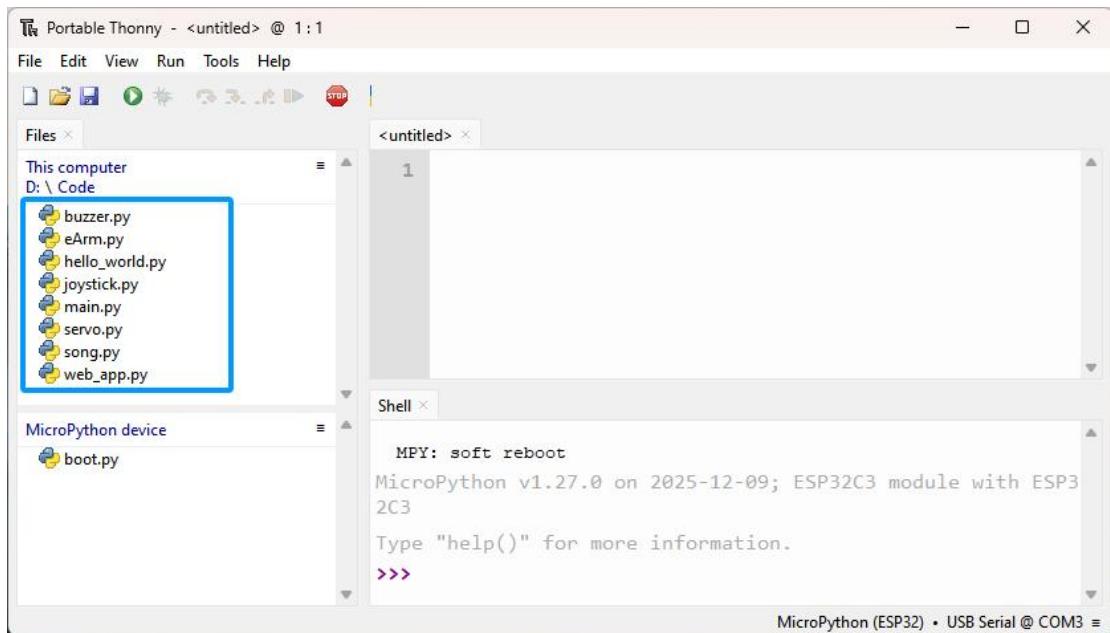
# Actually execute the Python file
# exec() runs the code in the file with global context
# open(filename).read() reads the entire file content
# globals() provides the current global
# namespace to the executed code
exec(open(filename).read(), globals())

# StopIteration exception is raised when iterator has no more items
except StopIteration:
    break # Exit the infinite while loop

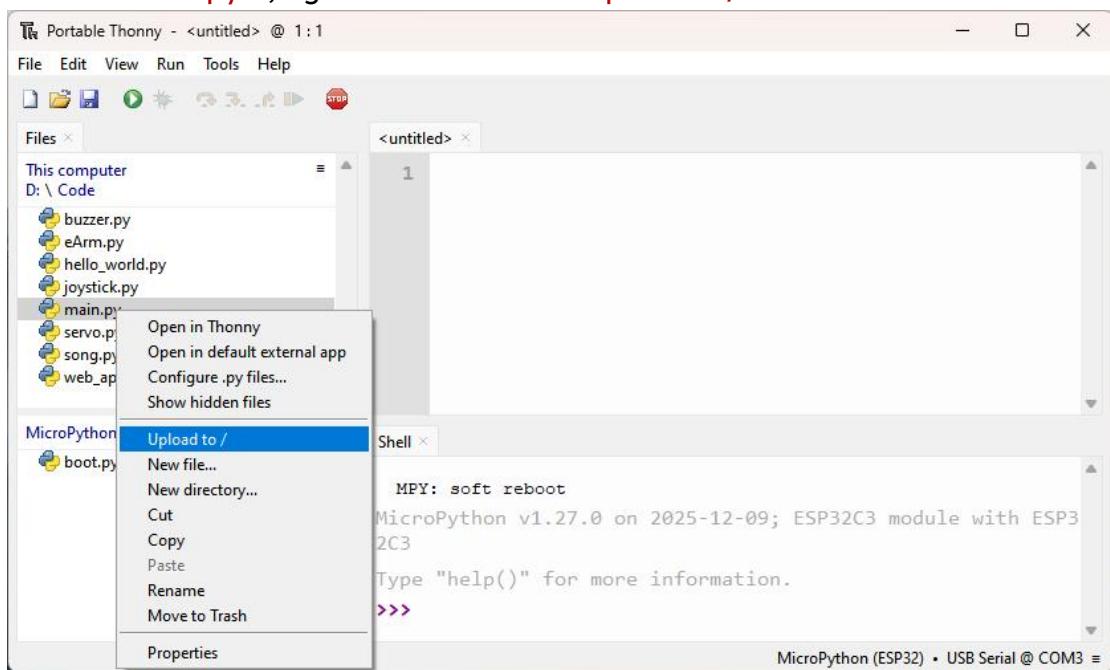
```

Expand “Code” folder in the directory of disk(D), and then double-click the “Code” folder to display all the sample codes.



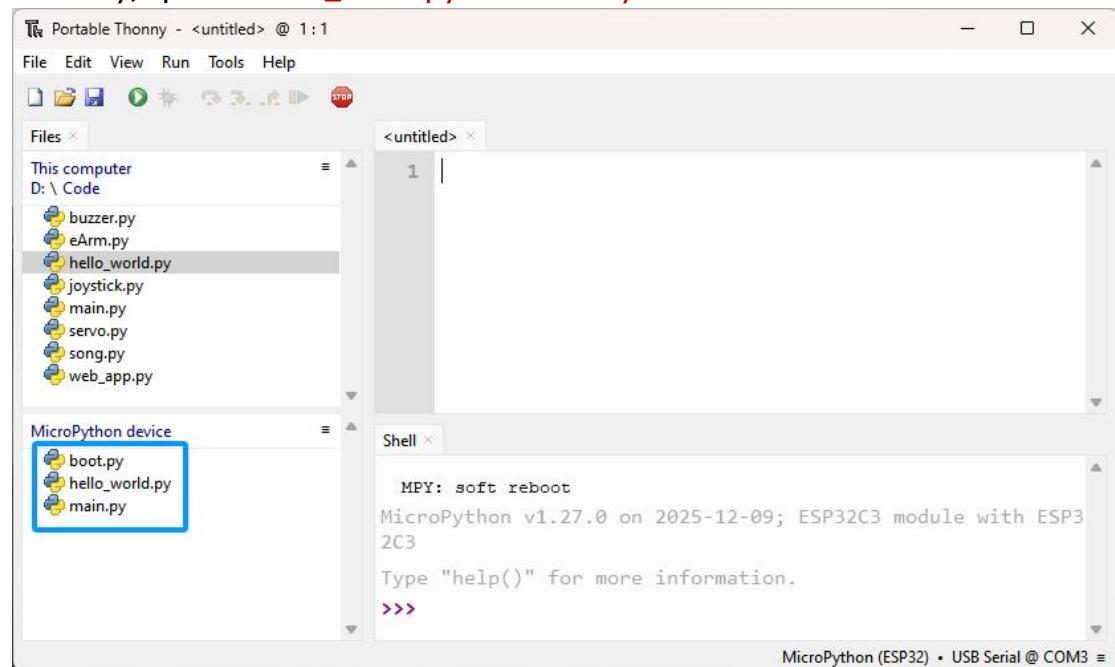


If you want your written programs to run offline, you need to upload **main.py** we provided and all your codes to “**MicroPython device**”, and then press the **reset button** on the control board. Here we use programs “**hello_world.py**” as examples. Select “**main.py**”, right-click to select “**Upload to /**”.

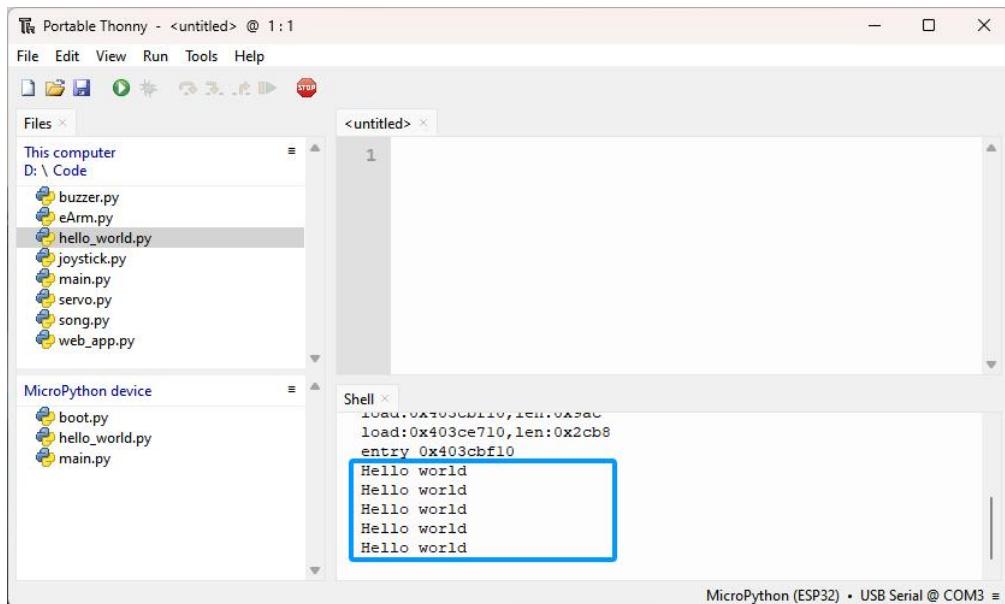
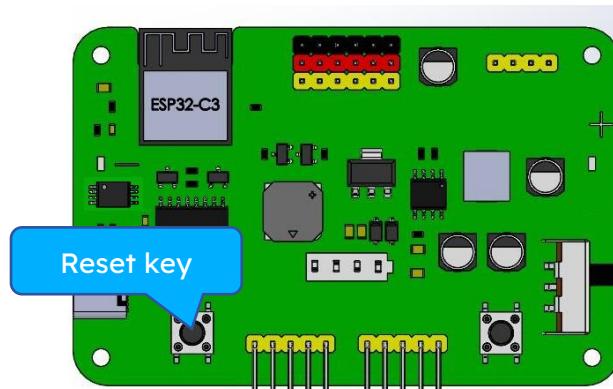


Note: You must stop running online before uploading files to "MicroPython device"! (Click the "**STOP**" menu in Thonny)

Similarly, upload "hello_world.py" to "MicroPython device".

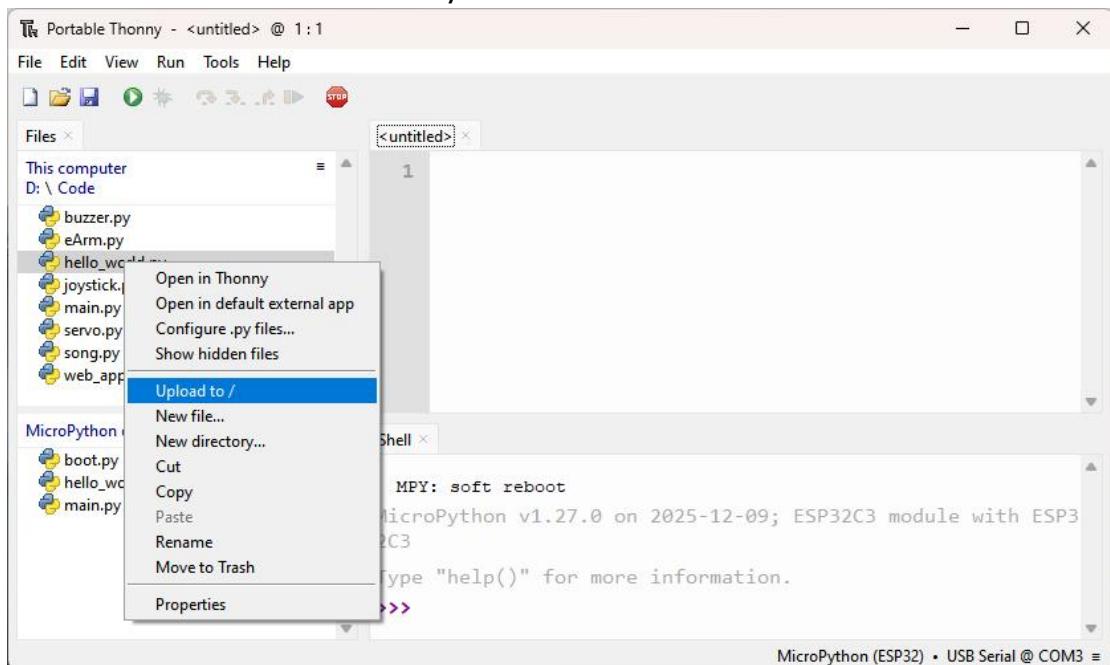


Press the reset key and in the box of the illustration below, you can see the code is executed.

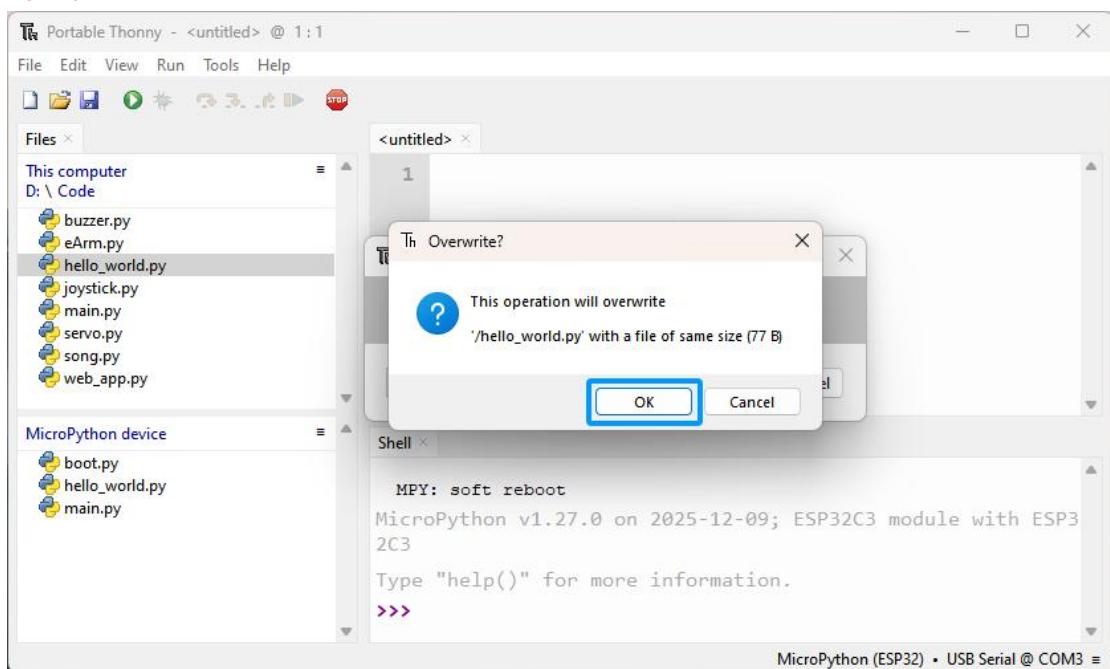


4.5.4 Upload code from computer to ESP32-C3

Select “hello_world.py”, right-click your mouse and select “Upload to /” to upload code to ESP32-C3’s root directory.

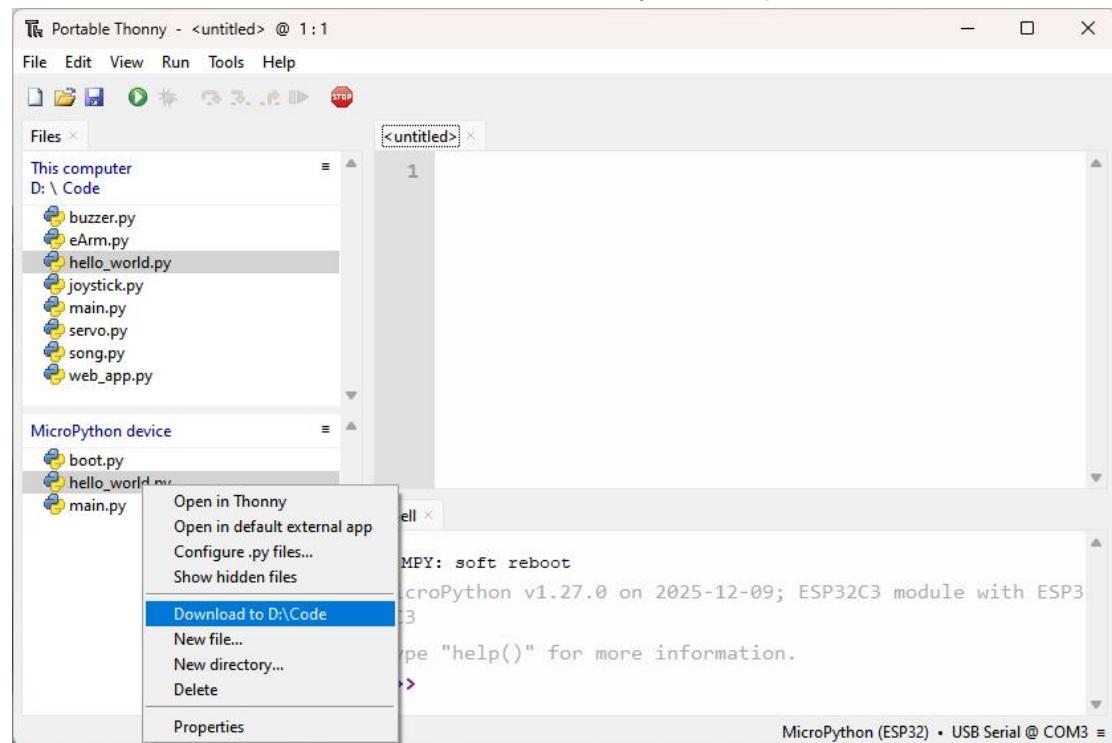


If the pop-up window prompts whether to overwrite the "boot.py" file, please click "OK".



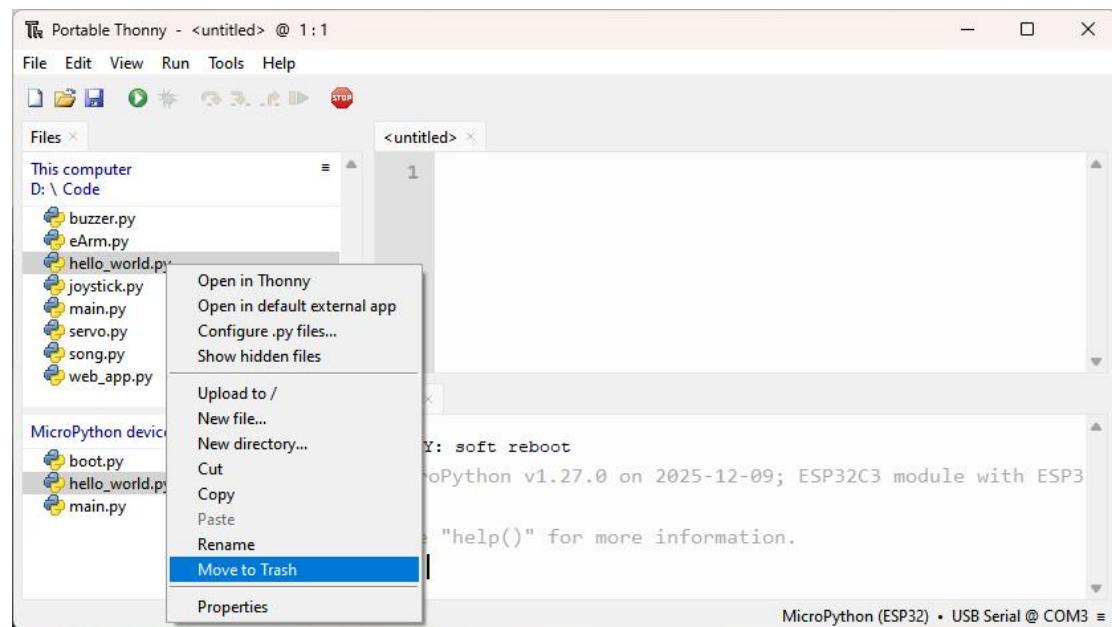
4.5.5 Download the code from ESP32-C3 to computer

Select “hello_world.py” in “MicroPython device”, right-click to select “Download to …” to download the code to your computer.



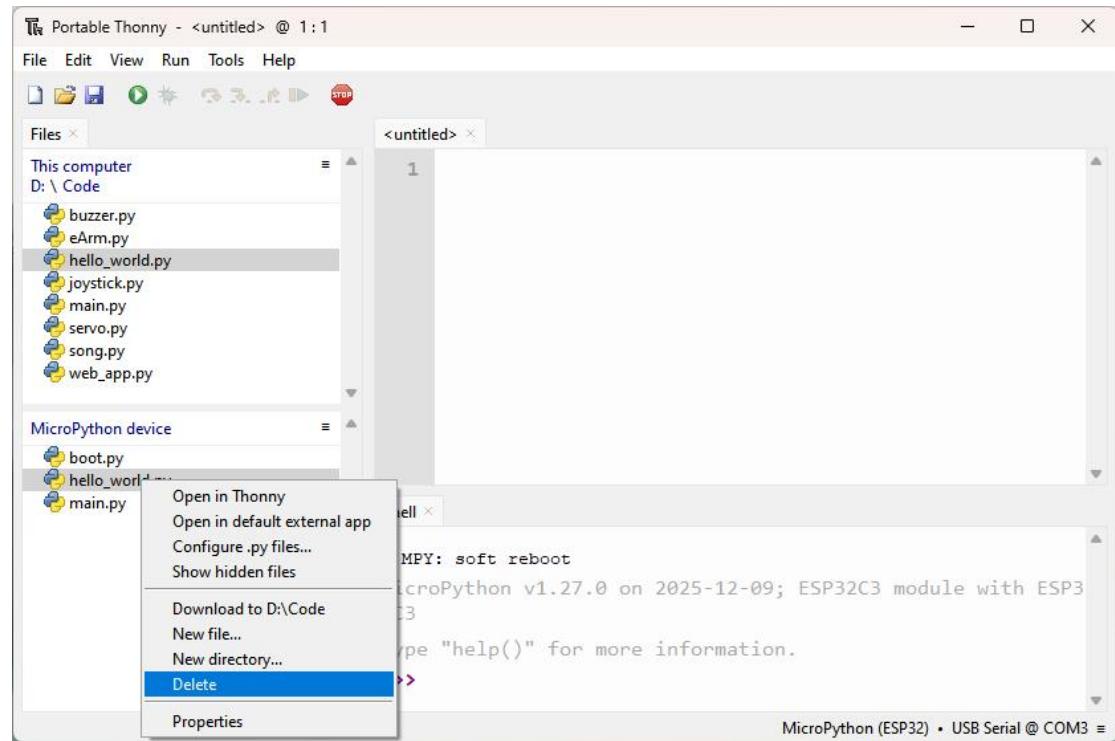
4.5.6 Deleting Files from your Computer Directory

Select “hello_world.py” in “Code”, right-click it and select “Move to Recycle Bin” to delete it from “Code”.



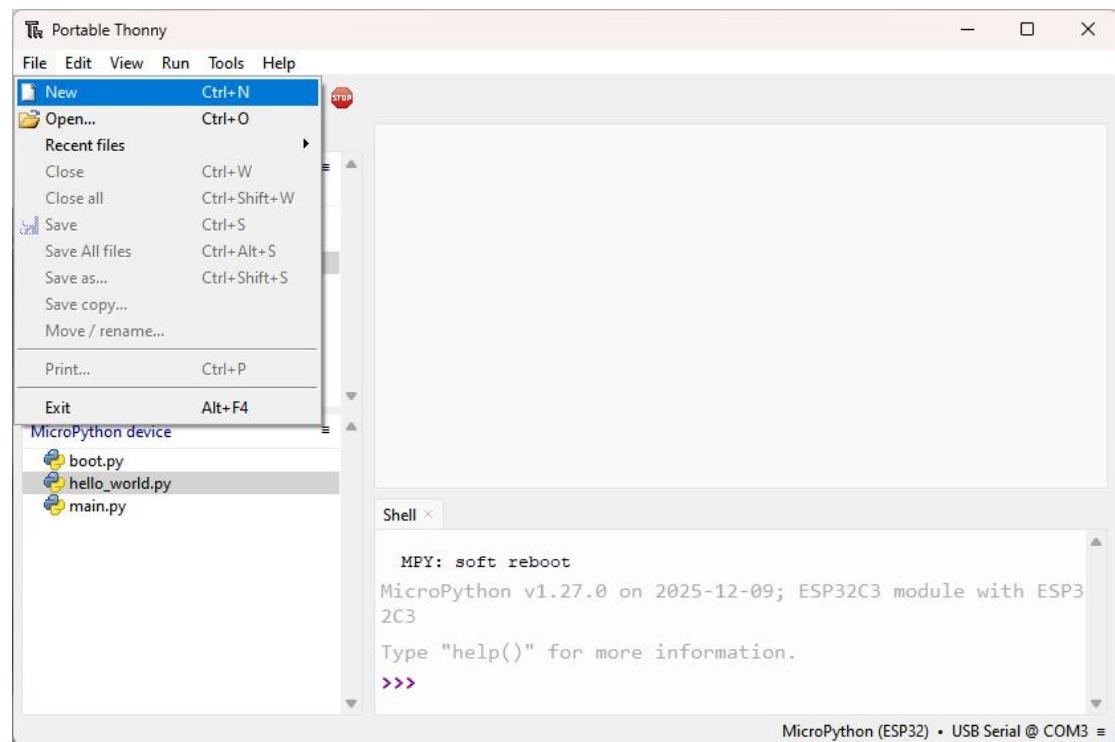
4.5.7 Deleting Files from ESP32-C3's Root Directory

Select “hello_world.py” in “MicroPython device”, right-click it and select “Delete” to delete “hello_world.py” from ESP32-C3’s root directory.



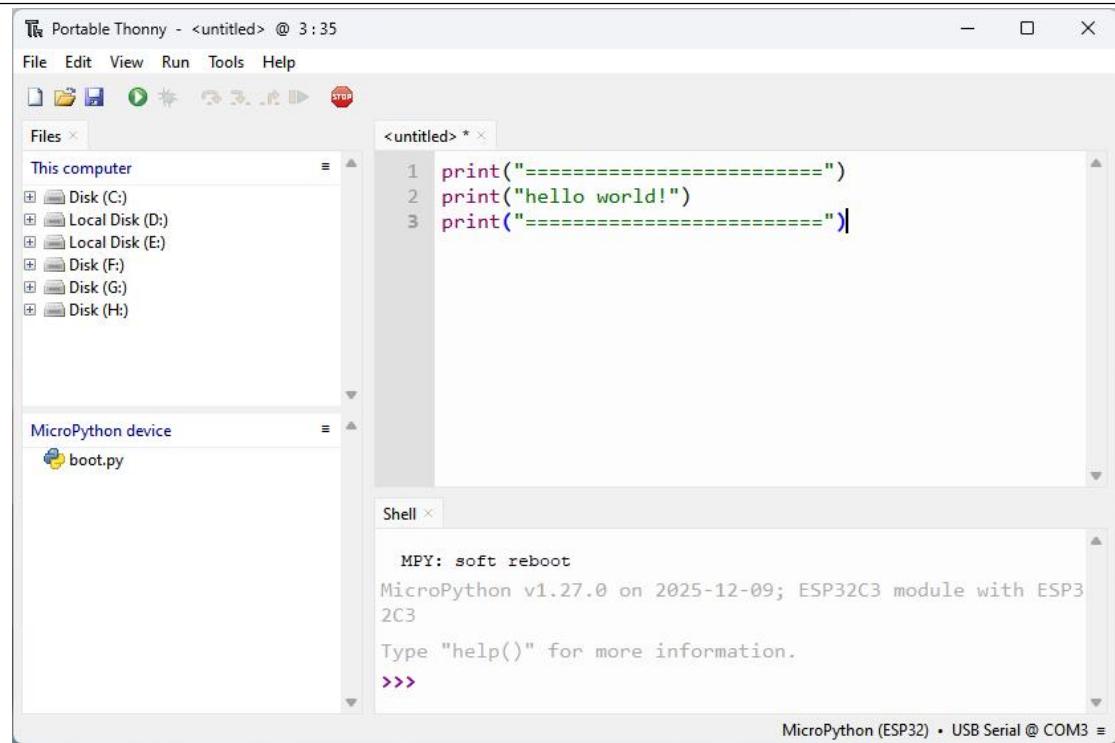
4.5.8 Creating and Saving the code

Click “File” -> “New” to create and write codes.

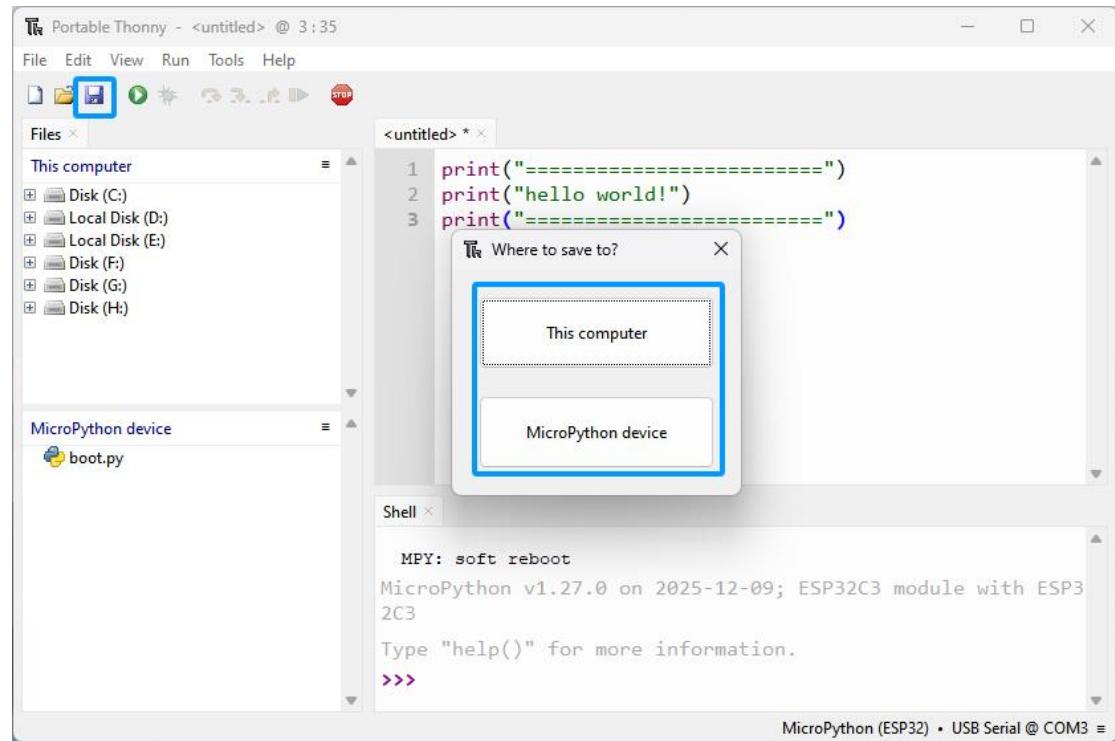


Enter codes in the newly opened file.

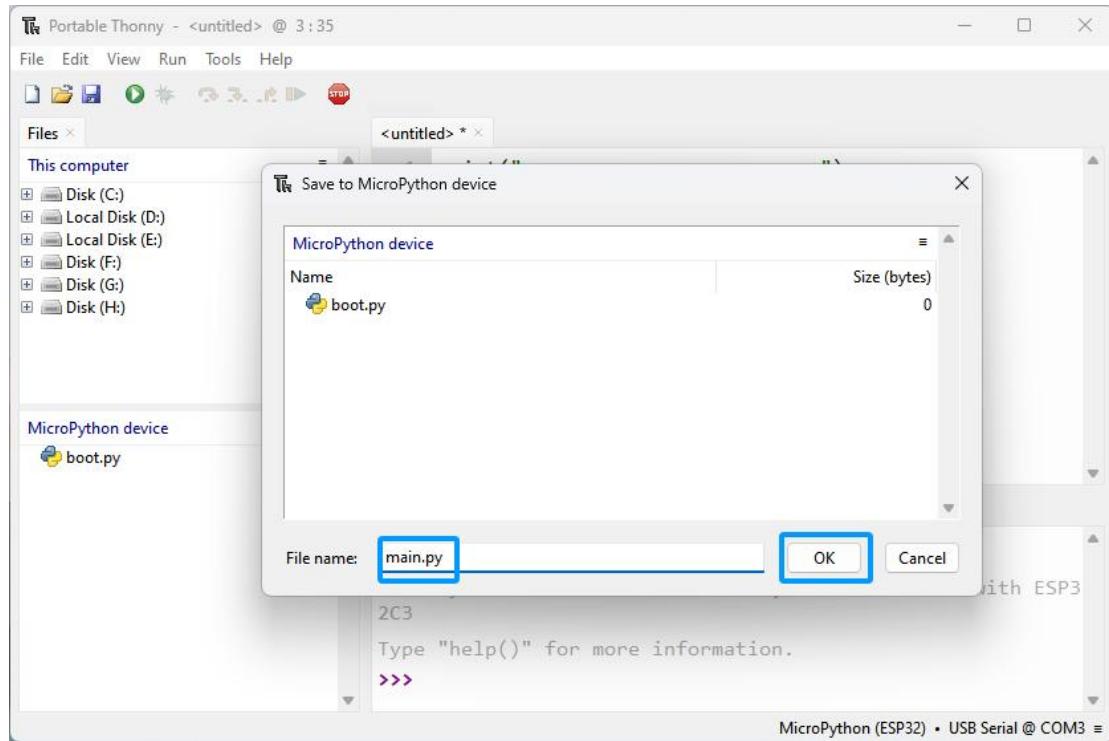
```
print("=====")
print("hello world!")
print("=====")
```



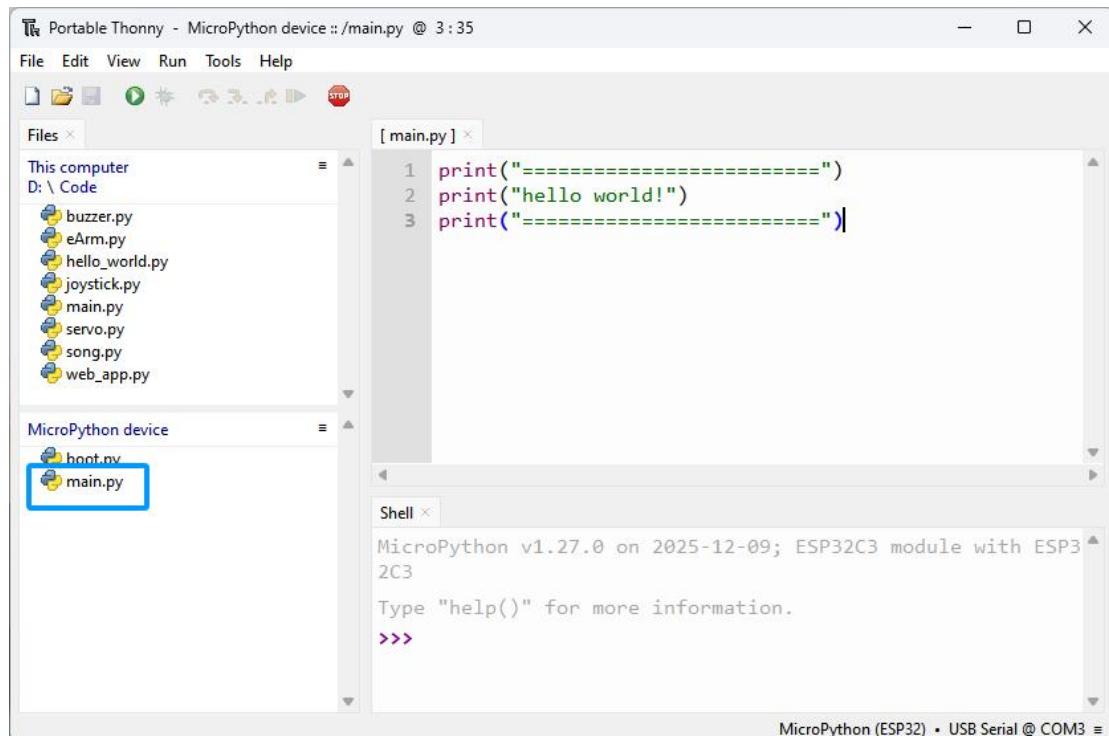
Click “Save” on the menu bar. You can save the codes either to your computer or to ESP32-C3.



Select “MicroPython device”, enter “main.py” in the newly pop-up window and click “OK”.

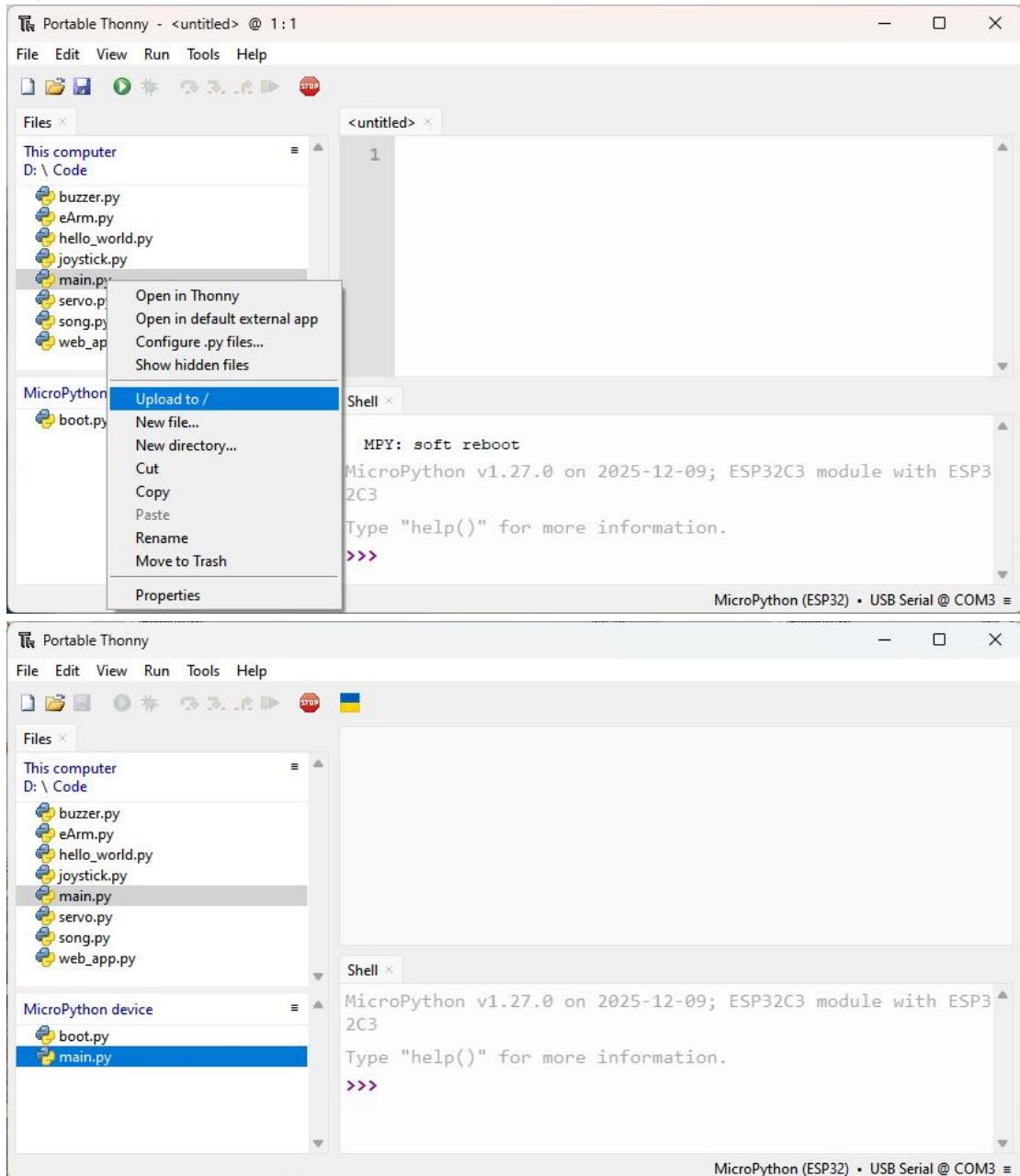


You can see that codes have been uploaded to ESP32-C3.



4.6 Code Examples

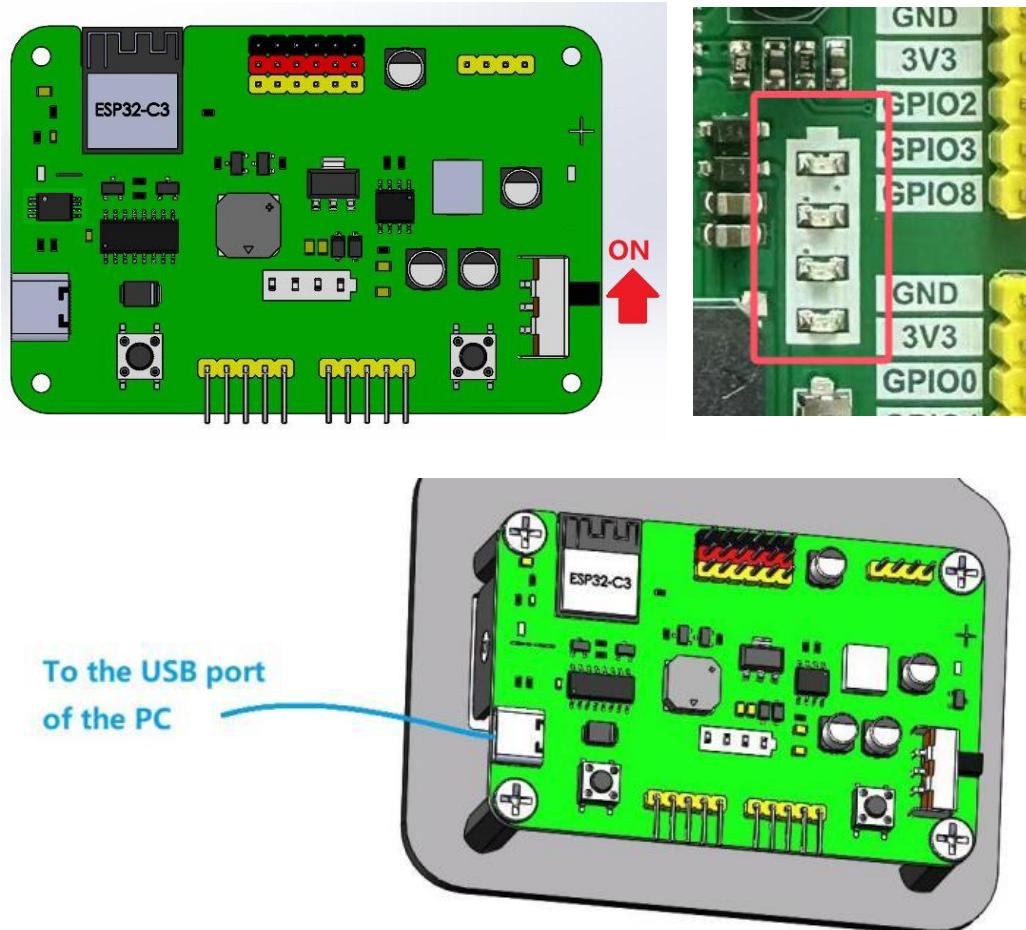
To enable the program to run offline, you need to upload the **main.py** file we provided to the "MicroPython Device". Select "**main.py**", right-click and select "**Upload to /**".



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

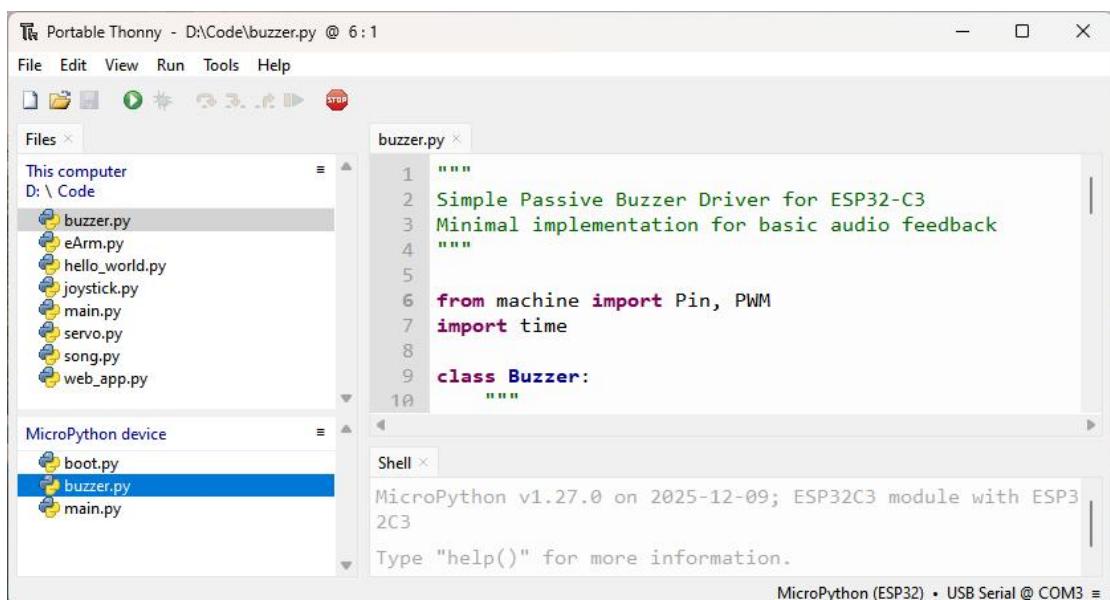
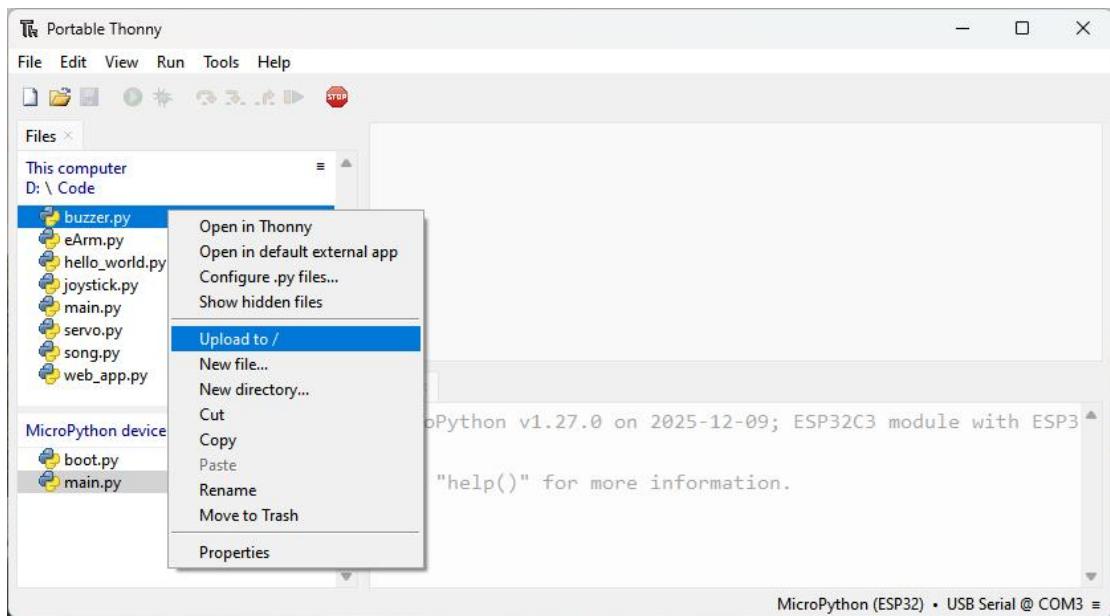
Turn the eArm's power switch to the **ON** position.

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

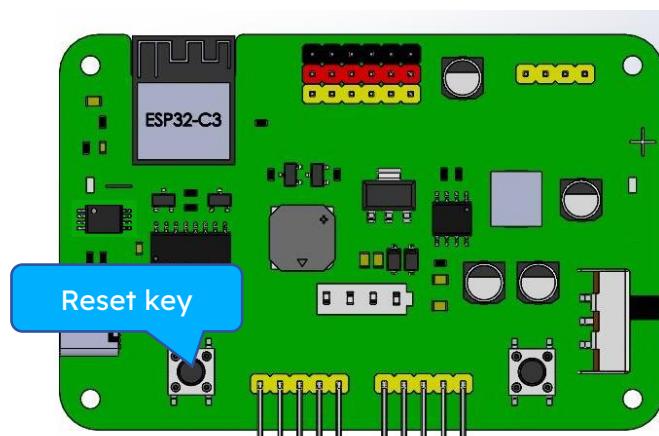


4.5.1 Onboard buzzer

Select “**buzzer.py**” , right-click your mouse and select “**Upload to /**” to upload code to ESP32-C3’s root directory.



Press the reset key and in the box of the illustration below, you can see the code is executed.



Portable Thonny - D:\Code\buzzer.py @ 6:1

File Edit View Run Tools Help

Files x

This computer

D:\Code

- buzzer.py
- eArm.py
- hello_world.py
- joystick.py
- main.py
- servo.py
- song.py
- web_app.py

MicroPython device

boot.py

buzzer.py

main.py

buzzer.py

Simple Passive Buzzer Driver for ESP32-C3

Minimal implementation for basic audio feedback

```
1 """
2 Simple Passive Buzzer Driver for ESP32-C3
3 Minimal implementation for basic audio feedback
4 """
5
6 from machine import Pin, PWM
7 import time
8
9 class Buzzer:
10     """
11
12     rst:0x1 (POWERON),boot:0xc (SPI_FAST_FLASH_BOOT)
13     SPIWP:0xee
14     mode:DIO, clock div:1
15     load:0x3fcfd5820,len:0xddc
16     load:0x403cbf10,len:0x9ac
17     load:0x403ce710,len:0x2cb8
18     entry 0x403cbf10
19
20     Buzzer initialized on GPIO9
21 
```

Shell x

rst:0x1 (POWERON),boot:0xc (SPI_FAST_FLASH_BOOT)

SPIWP:0xee

mode:DIO, clock div:1

load:0x3fcfd5820,len:0xddc

load:0x403cbf10,len:0x9ac

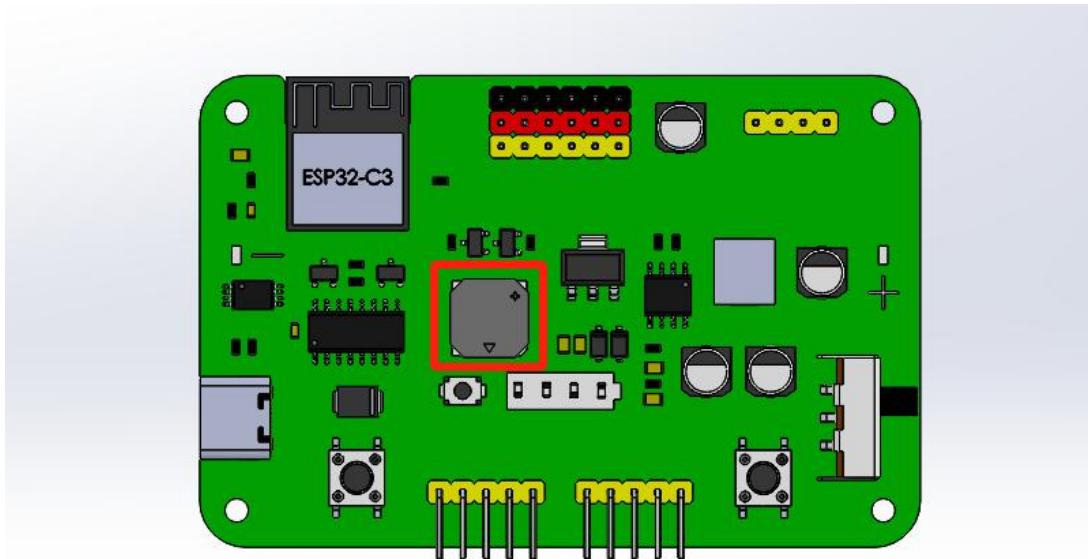
load:0x403ce710,len:0x2cb8

entry 0x403cbf10

Buzzer initialized on GPIO9

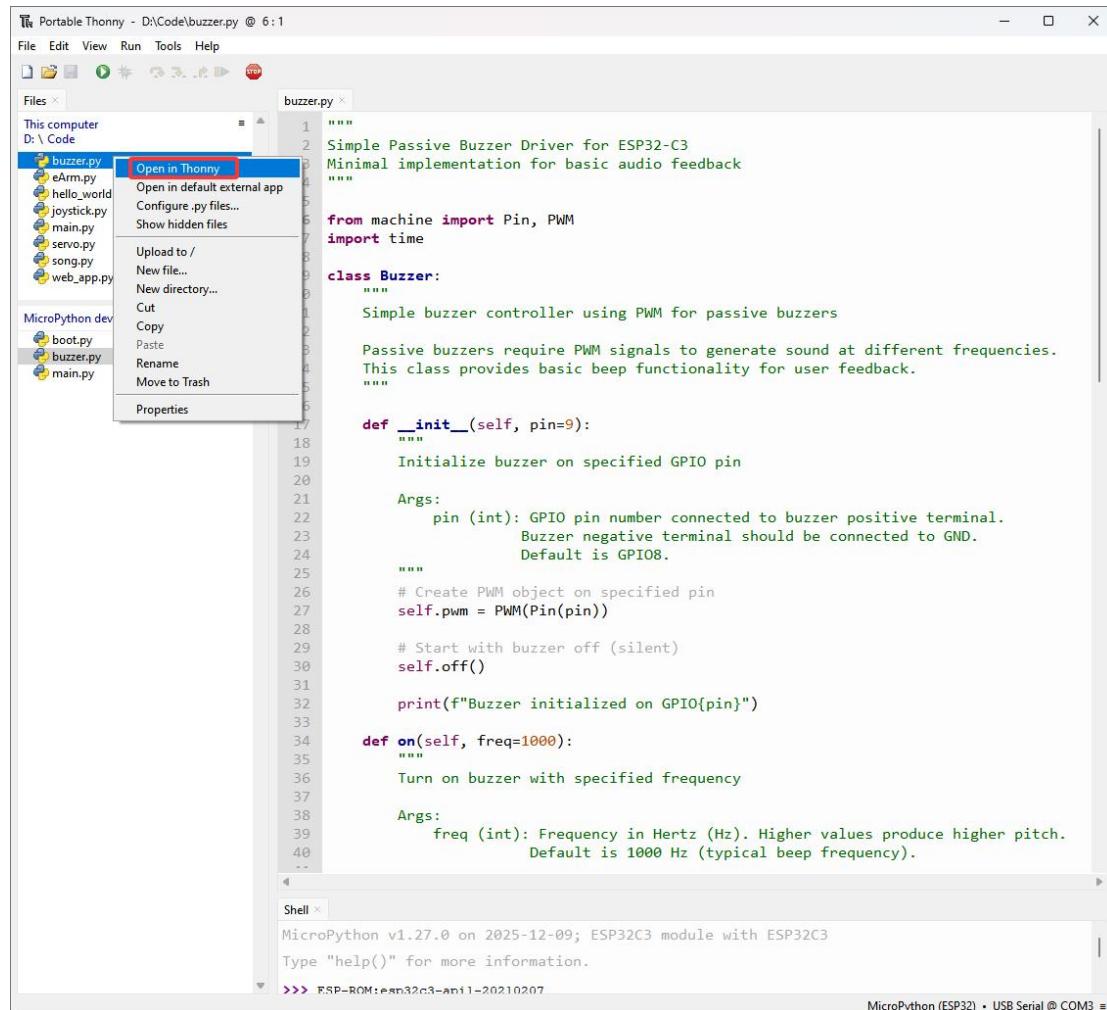
MicroPython (ESP32) • USB Serial @ COM3 =

After the code is uploaded successfully, the buzzer on the ESP32-C3 board will keep making sounds:

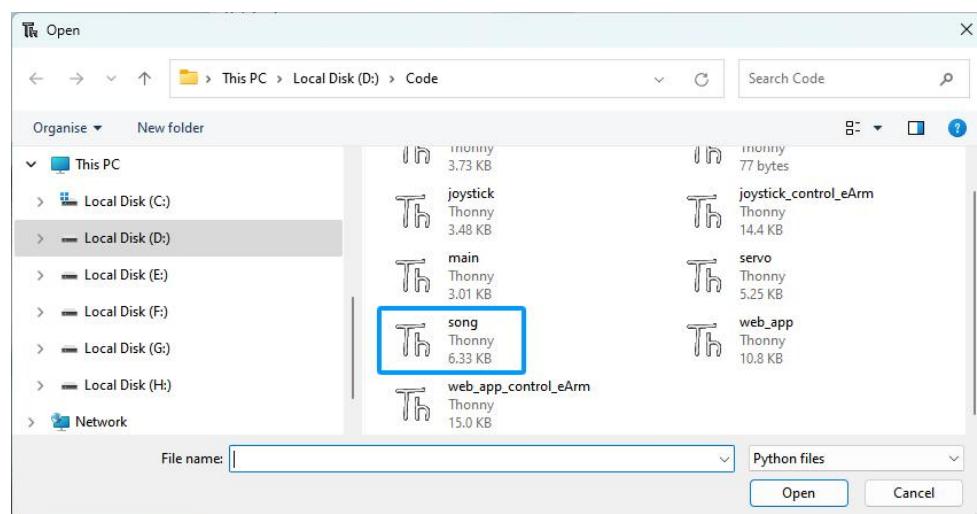


Code Explanation:

The detailed analysis of the code is in the ".py" file. Please open and read it with Thonny IDE.

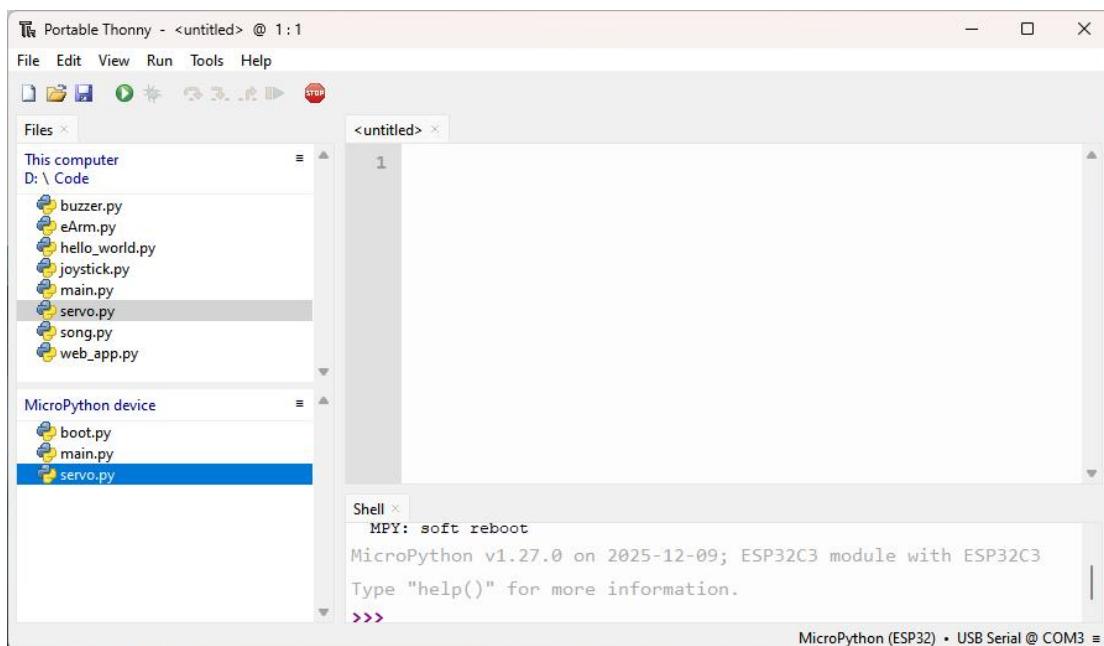
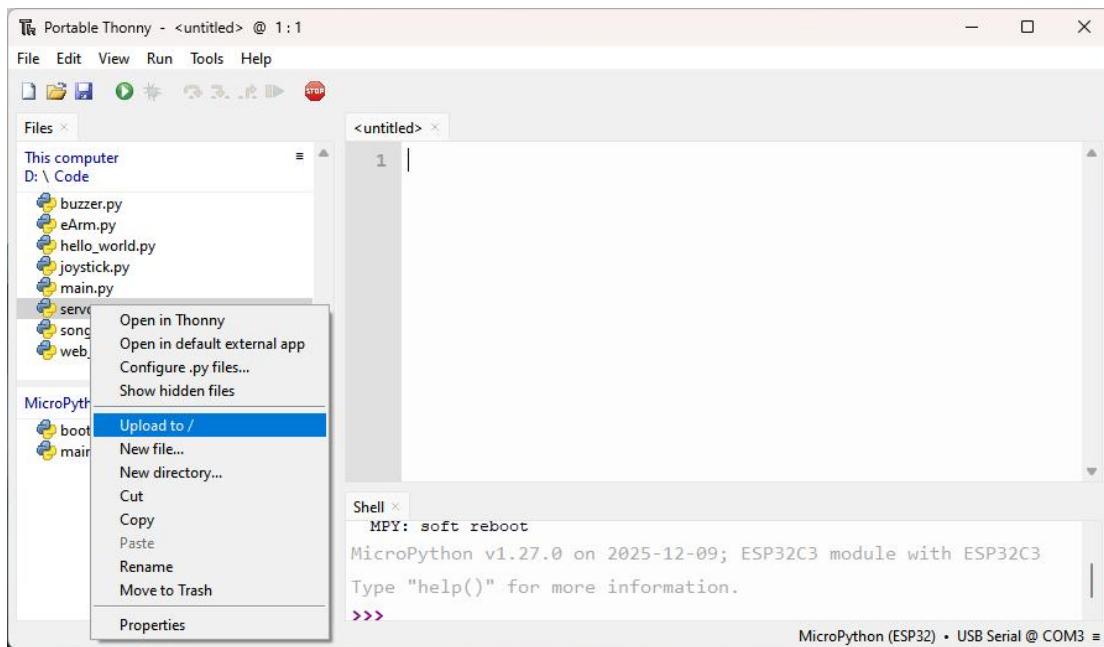


We also provide a sample code for a passive buzzer playing songs. If you are interested, you can follow the steps above to implement it.

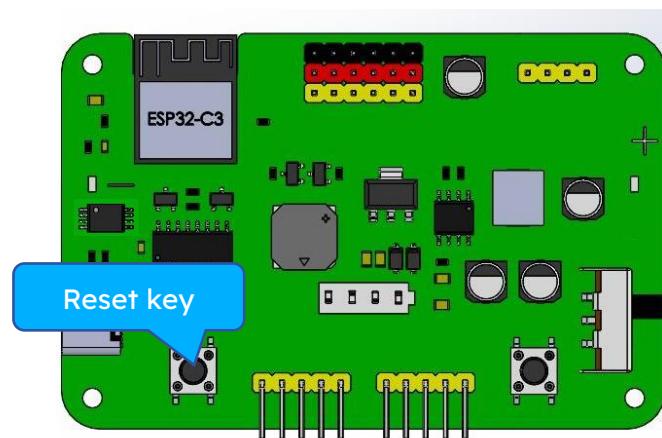


4.5.2 To Drive a Servo

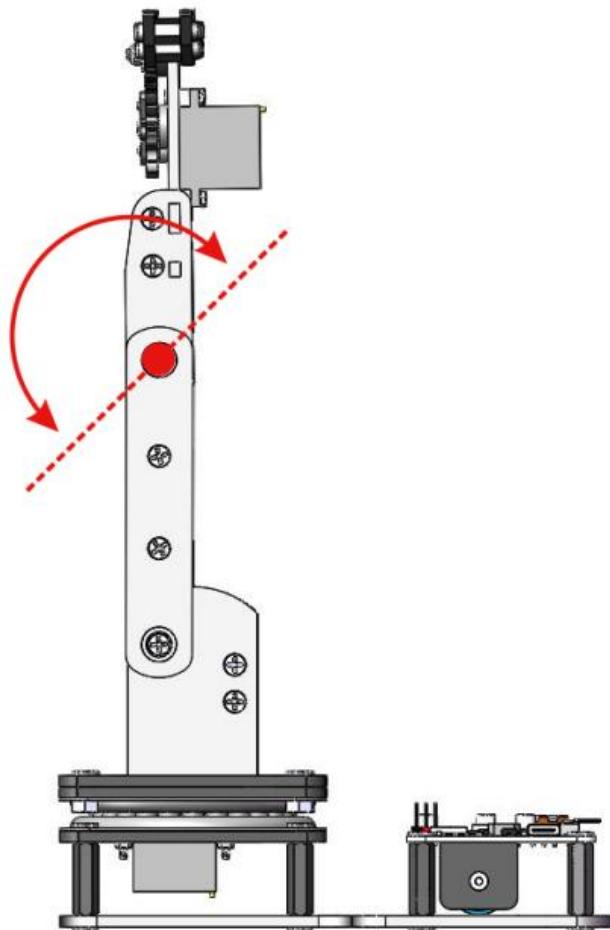
Select “servo.py”, right-click your mouse and select “Upload to /” to upload code to ESP32-C3’s root directory.



Press the **reset key** and in the box of the illustration below, you can see the code is executed.



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



Code Explanation:

The detailed analysis of the code is in the ".py" file. Please open and read it with Thonny IDE.

The screenshot shows the Thonny IDE interface. The main window displays the Python code for `servo.py`. The code defines a `Servo` class for controlling an ESP32-C3 MicroPython servo. The `__init__` method initializes the servo with a specified pin number, frequency, and angle range. It includes validation for the angle parameters and initializes the pin and PWM. The code is annotated with docstrings and type hints for the parameters. In the bottom right corner, the status bar indicates "MicroPython (ESP32) • USB Serial @ COM3". On the left, the file browser shows other files like `buzzer.py`, `eArm.py`, etc. A context menu is open over the `servo.py` file, with the option "Open in Thonny" highlighted.

```
from machine import Pin, PWM
import time

class Servo:
    """
    Servo control class
    For ESP32-C3 MicroPython servo control
    """

    def __init__(self, pin_num, freq=50, min_angle=0, max_angle=180)
        """
        Initialize servo

        Parameters:
            pin_num: GPIO pin number (e.g., 1, 2, 3...)
            freq: PWM frequency, default 50Hz (standard servo frequency)
            min_angle: Minimum angle, default 0 degrees
            max_angle: Maximum angle, default 180 degrees
        """

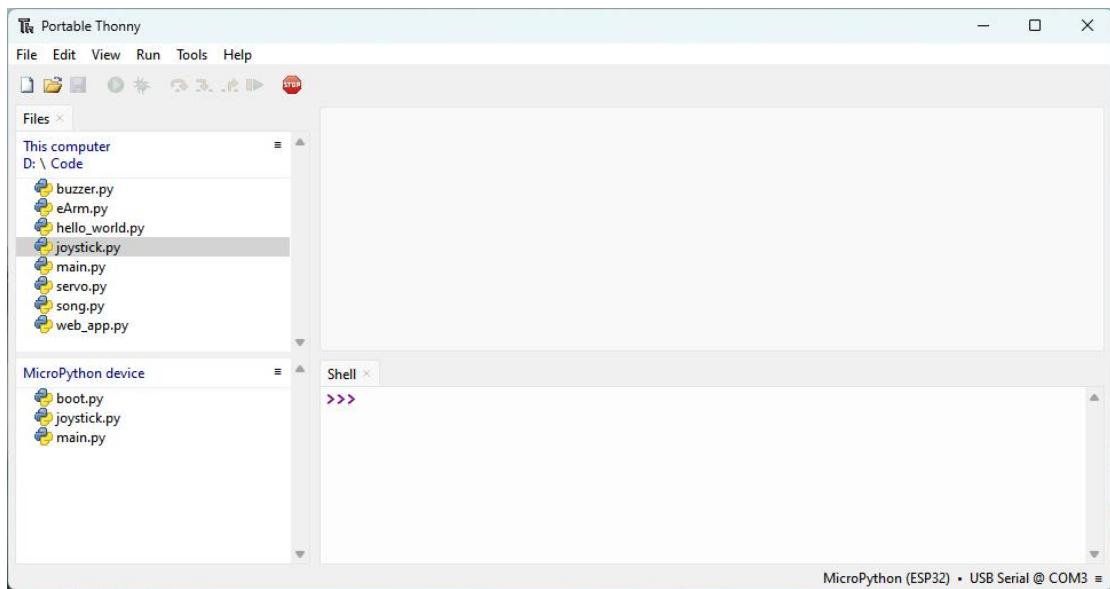
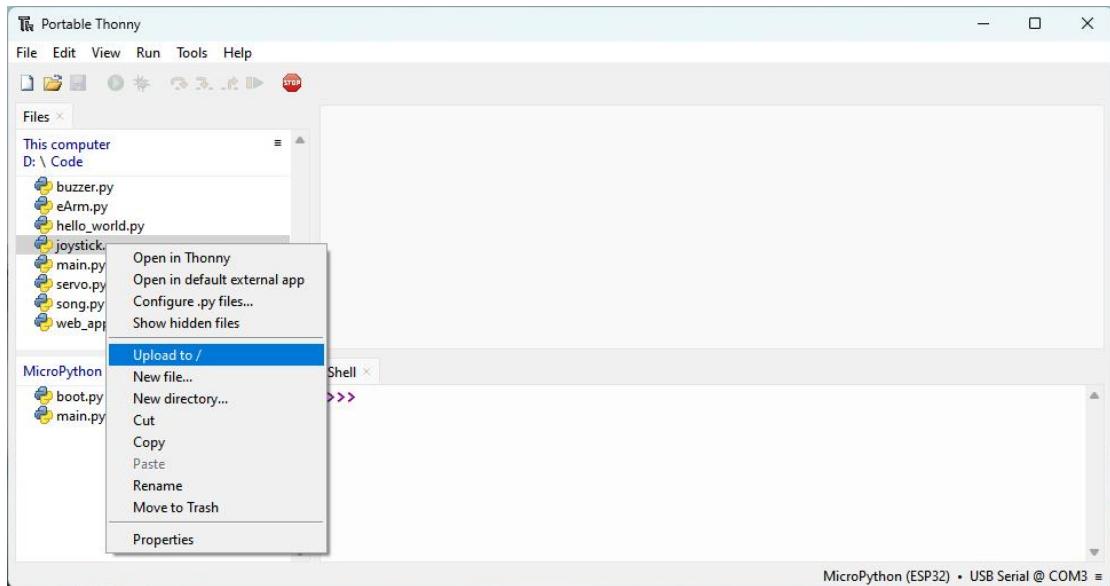
        # Validate parameters
        if min_angle < 0 or max_angle > 180:
            raise ValueError("Angle range should be 0-180 degrees")
        if min_angle >= max_angle:
            raise ValueError("Minimum angle must be less than maximum")

        # Initialize pin and PWM
        self.pin = Pin(pin_num, Pin.OUT)
        self.pwm = PWM(self.pin, freq=freq)

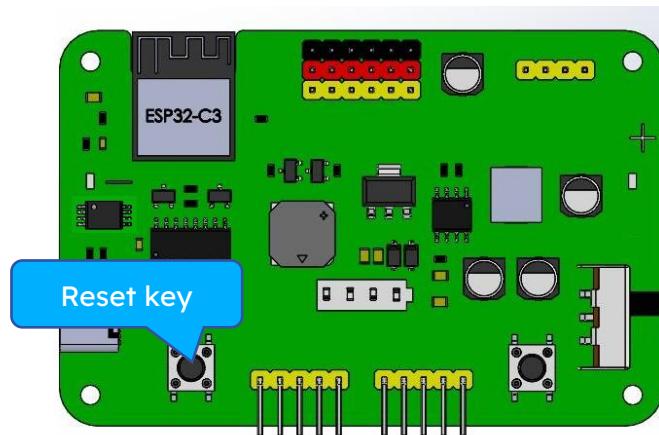
        # Servo parameters
        self.freq = freq
        self.min_angle = min_angle
        self.max_angle = max_angle
```

4.5.3 Read the Value of the Joystick

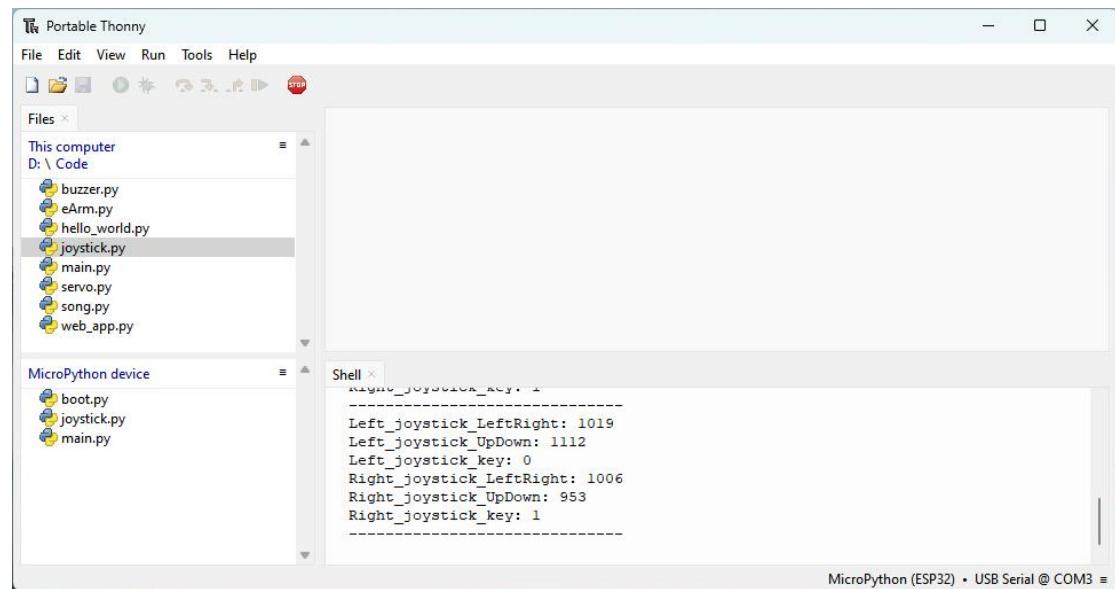
Select “**joystick.py**” , right-click your mouse and select “**Upload to /**” to upload code to ESP32-C3’s root directory.



Press the reset key and in the box of the illustration below, you can see the code is executed.



Shake the handle left, right, up, down, or press the joystick down, and the shell will print the joystick value:



Code Explanation:

The detailed analysis of the code is in the ".py" file. Please open and read it with Thonny IDE.

The screenshot shows the Thonny IDE interface. The main window displays the code for `joystick.py`. The code reads analog and digital inputs from two joysticks connected to an ESP32-C3 module. It uses the `ADC` and `Pin` classes from the `machine` module. The code is well-structured with comments explaining the hardware pin configurations and ADC settings. A context menu is open over the code editor, showing options like "Open in Thonny", "Open in default external app", "Configure .py files...", and "Upload to /". Below the code editor is a "Shell" window showing the MicroPython prompt and the output of a soft reboot command. The bottom status bar indicates "MicroPython (ESP32) • USB Serial @ COM3 =".

```
1 """
2 Simple eArm Joystick Reader for ESP32-C3
3 Reads analog and digital inputs from two joysticks
4 """
5
6 from machine import Pin, ADC
7 import time
8
9 # ===== Hardware Pin Configuration =====
10 # Configure pins according to eArm hardware layout
11
12 # Left Joystick Configuration:
13 # Horizontal (Left-Right) axis connected to GPIO1 as analog input
14 left_lr = ADC(Pin(1)) # GPIO1: Left/Right movement of left joystick
15 # Vertical (Up-Down) axis connected to GPIO0 as analog input
16 left_ud = ADC(Pin(0)) # GPIO0: Up/Down movement of left joystick
17 # Button/Key press connected to GPIO8 as digital input
18 left_key = Pin(10, Pin.IN) # GPIO8: Button on left joystick
19
20 # Right Joystick Configuration:
21 # Horizontal (Left-Right) axis connected to GPIO3 as analog input
22 right_lr = ADC(Pin(3)) # GPIO3: Left/Right movement of right joystick
23 # Vertical (Up-Down) axis connected to GPIO2 as analog input
24 right_ud = ADC(Pin(2)) # GPIO2: Up/Down movement of right joystick
25 # Button/Key press connected to GPIO10 as digital input with internal pull-up
26 right_key = Pin(8, Pin.IN, Pin.PULL_UP) # GPIO10: Button on right joystick
27
28 # ===== ADC Configuration =====
29 # Configure ADC attenuation for 0-3.3V full range input
30 # ATTN_11DB allows measuring voltages from 0V to approximately 3.6V
31
32 # Left joystick ADC configuration
33 left_lr.atten(ADC.ATTN_11DB) # Set full voltage range for left horizontal axis
34 left_ud.atten(ADC.ATTN_11DB) # Set full voltage range for left vertical axis
35
```

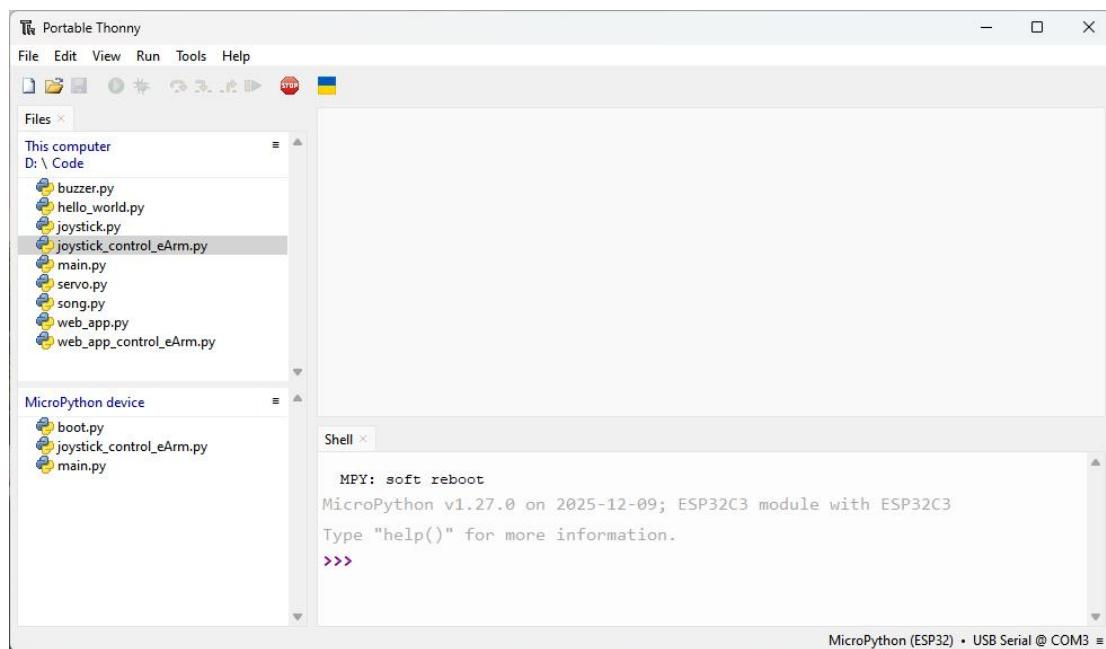
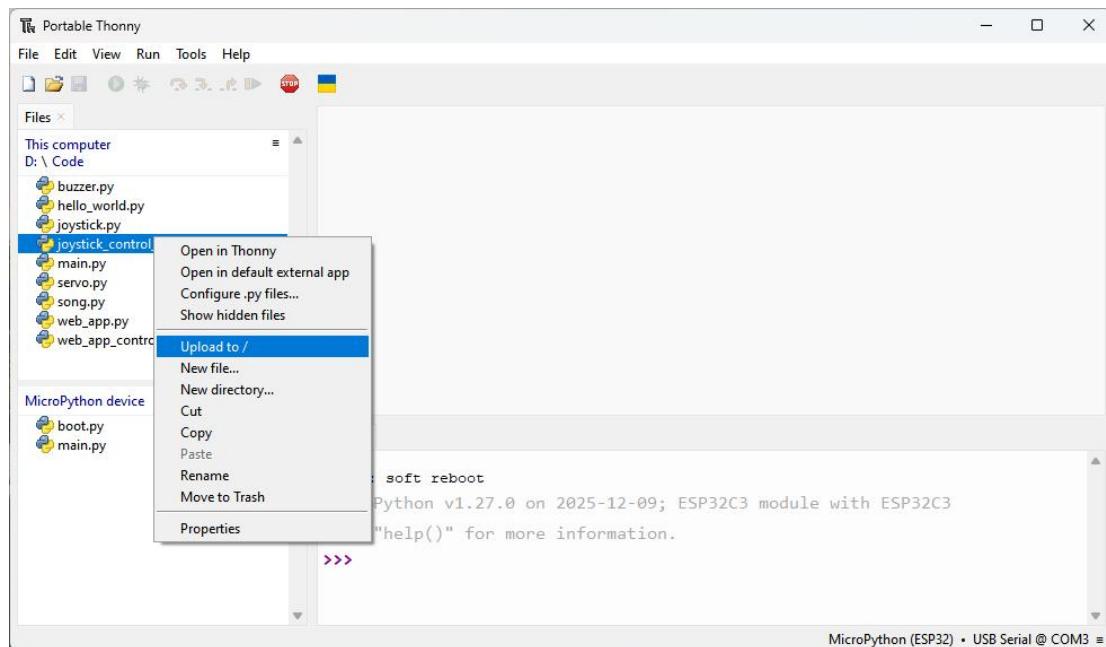
KeyboardInterrupt:
MicroPython v1.27.0 on 2025-12-09; ESP32C3 module with ESP32C3
Type "help()" for more information.

MPY: soft reboot
MicroPython v1.27.0 on 2025-12-09; ESP32C3 module with ESP32C3
Type "help()" for more information.
>>>

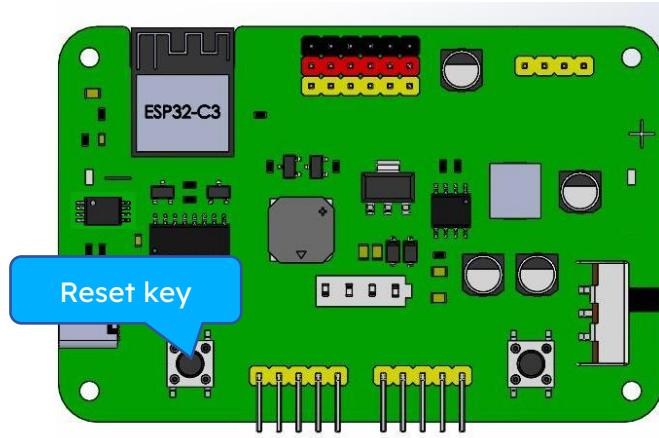
MicroPython (ESP32) • USB Serial @ COM3 =

4.5.4 Joystick Control eArm

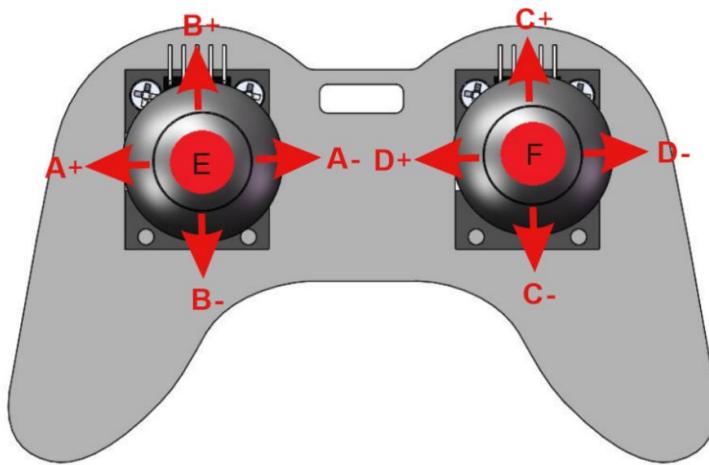
Select “**joystick_control_eArm.py**” , right-click your mouse and select “**Upload to /**” to upload code to ESP32-C3’s root directory.



Press the **reset key** and in the box of the illustration below, you can see the code is executed.



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



A+: Rotate arm left (Servo A)

B-: Raise rear arm (Servo B)

C-: Raise the forearm (Servo C)

D+: Open the gripper (Servo D)

A+: Rotate arm right (Servo A)

B+: Lower rear arm (Servo B)

C+: Lower the forearm (Servo C)

D-: Close the gripper (Servo D)

F: Reserve

E: Enable/disable the buzzer

Code Explanation:

The detailed analysis of the code is in the ".py" file. Please open and read it with Thonny IDE.

The screenshot shows the Thonny IDE interface with the following details:

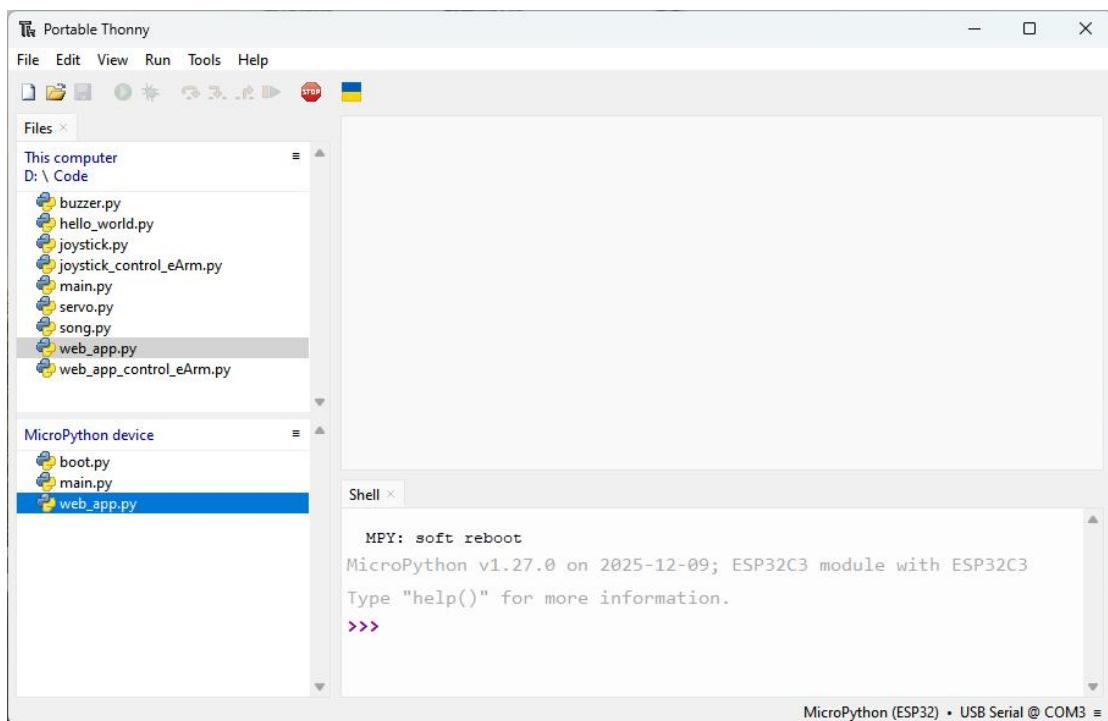
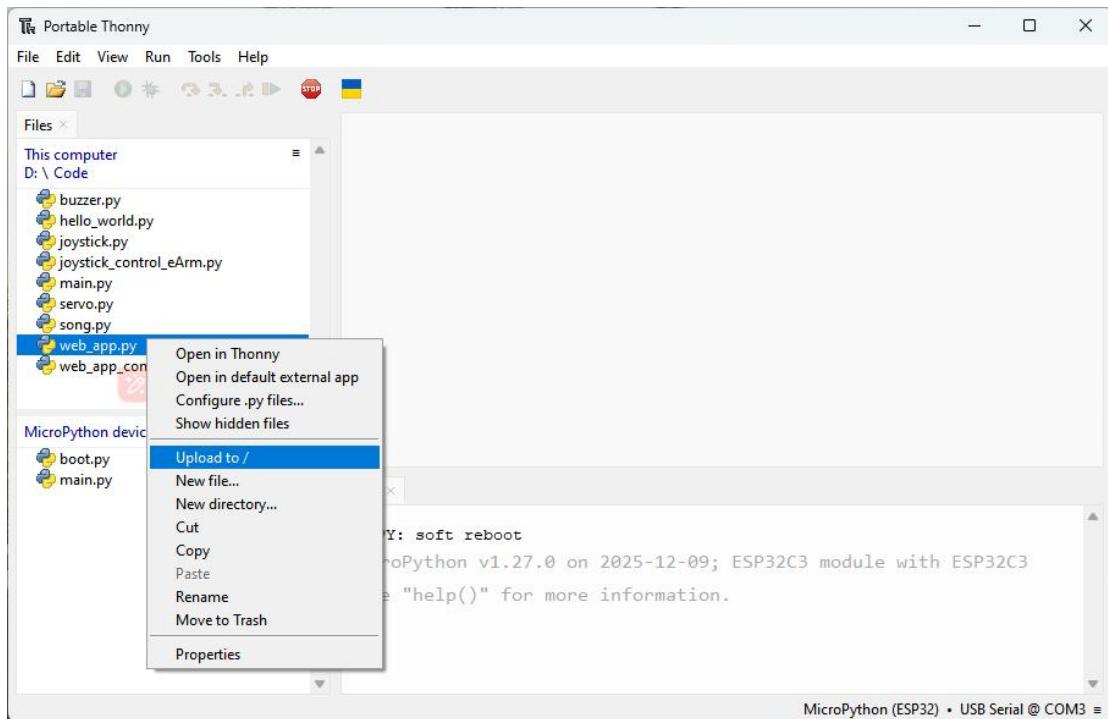
- Title Bar:** Portable Thonny - D:\Code\joystick_control_eArm.py @ 409 : 9
- File Menu:** File Edit View Run Tools Help
- Toolbar:** Standard icons for file operations.
- Left Sidebar:**
 - Files:** This computer (D:\Code) contains: buzzer.py, hello_world.py, joystick.py, joystick_control_eArm.py (selected), main.py, servo.py, song.py, web_app.py, web_app_control_eArm.py.
 - MicroPython device:** boot.py, main.py
- Code Editor:** The "joystick_control_eArm.py" tab is active, displaying the following Python code:

```
1 """
2 eArm Robotic Arm Web Control System
3 Real-time button control with automatic servo adjustment
4 """
5
6 from machine import Pin, PWM, ADC
7 import time
8 import _thread
9
10 # ======Joystick Hardware Pin Configuration ======
11
12 # Left Joystick Configuration:
13 # Horizontal (Left-Right) axis connected to GPIO1 as analog input
14 left_lr = ADC(Pin(1)) # GPIO1: Left/Right movement of left joystick
15 # Vertical (Up-Down) axis connected to GPIO0 as analog input
16 left_ud = ADC(Pin(0)) # GPIO0: Up/Down movement of left joystick
17 # Button/Key press connected to GPIO8 as digital input
18 left_key = Pin(10, Pin.IN, Pin.PULL_UP) # GPIO8: Button on left joystick
19
20 # Right Joystick Configuration:
21 # Horizontal (Left-Right) axis connected to GPIO3 as analog input
22 right_lr = ADC(Pin(3)) # GPIO3: Left/Right movement of right joystick
23 # Vertical (Up-Down) axis connected to GPIO2 as analog input
24 right_ud = ADC(Pin(2)) # GPIO2: Up/Down movement of right joystick
25 # Button/Key press connected to GPIO10 as digital input with internal pull-up
26 right_key = Pin(8, Pin.IN, Pin.PULL_UP) # GPIO10: Button on right joystick
27
28 # ====== ADC Configuration ======
29 # Configure ADC attenuation for 0-3.3V full range input
30 # ATTN_11DB allows measuring voltages from 0V to approximately 3.6V
31
32 # Left joystick ADC configuration
33 left_lr.atten(ADC.ATTN_11DB) # Set full voltage range for left horizontal axis
34 left_ud.atten(ADC.ATTN_11DB) # Set full voltage range for left vertical axis
35
```
- Shell:** Shows the MicroPython REPL output:

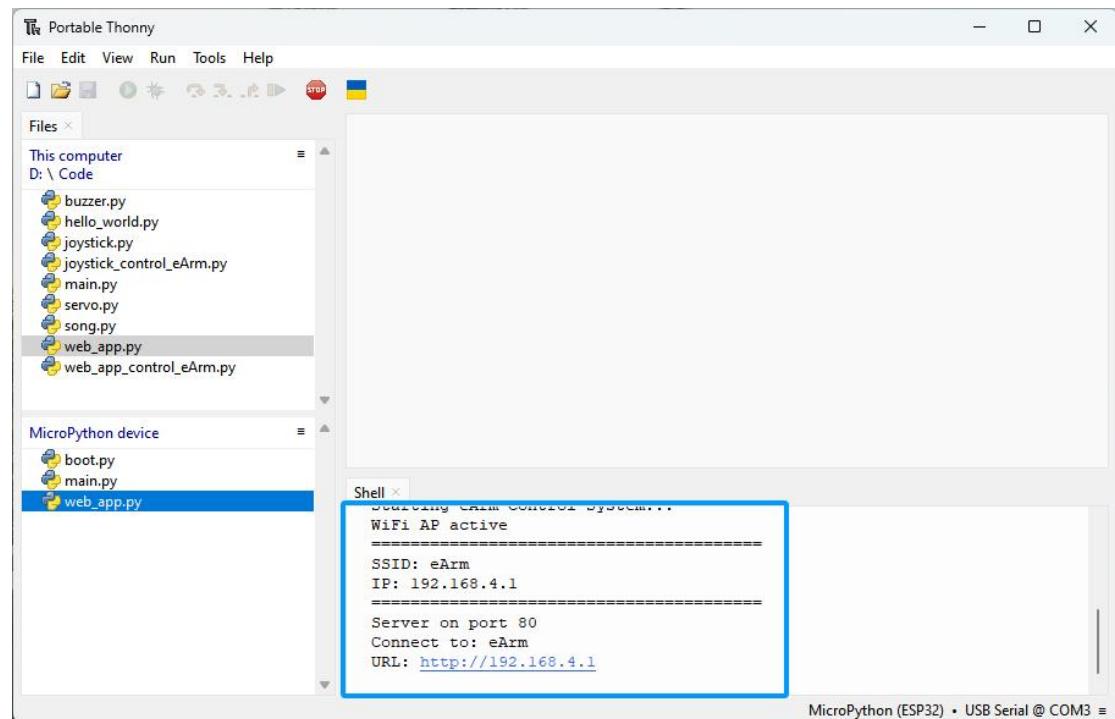
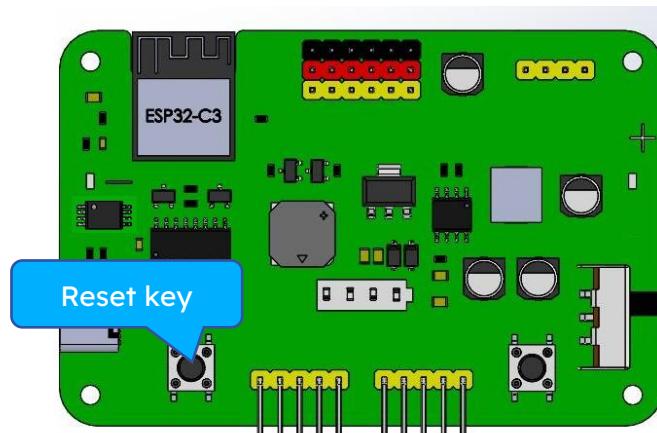
```
MPY: soft reboot
MicroPython v1.27.0 on 2025-12-09; ESP32C3 module with ESP32C3
Type "help()" for more information.
>>>
```
- Status Bar:** MicroPython (ESP32) • USB Serial @ COM3 =

4.5.5 Read the Value of the Web APP

Select “**web_app.py**” , right-click your mouse and select “**Upload to /**” to upload code to ESP32-C3’s root directory.

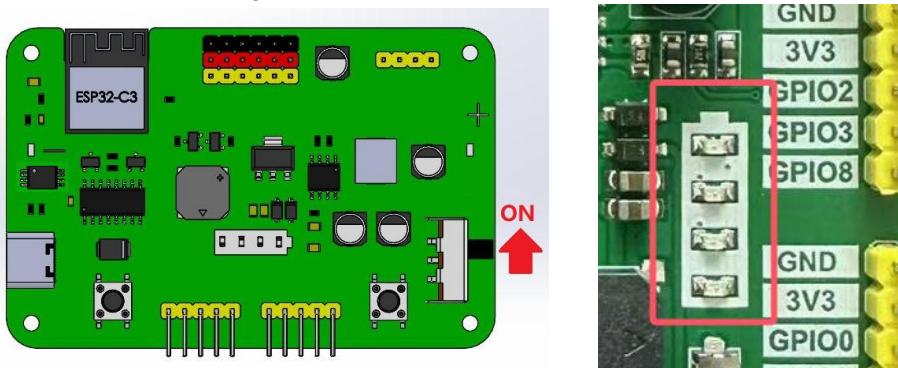


Press the reset key and in the box of the illustration below, you can see the code is executed.



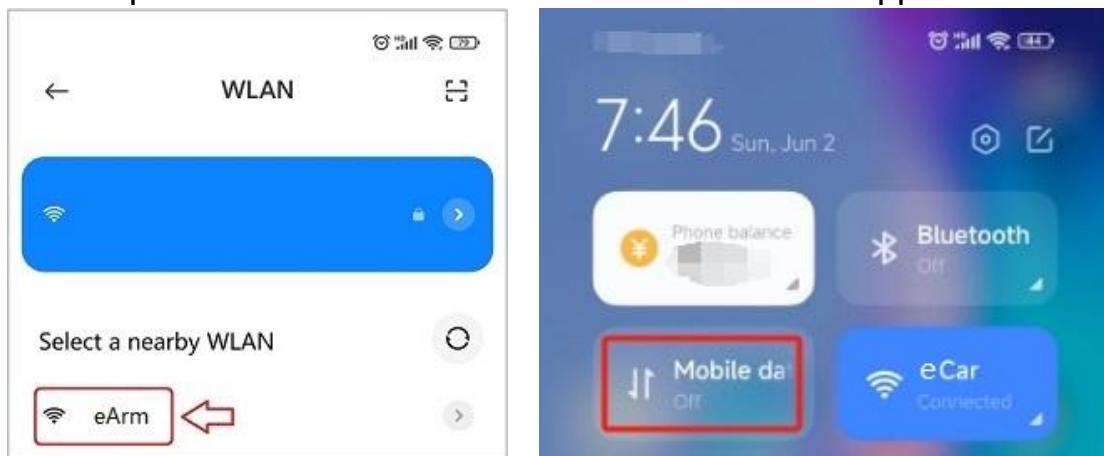
Turn the eArm's power switch to the ON position.

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

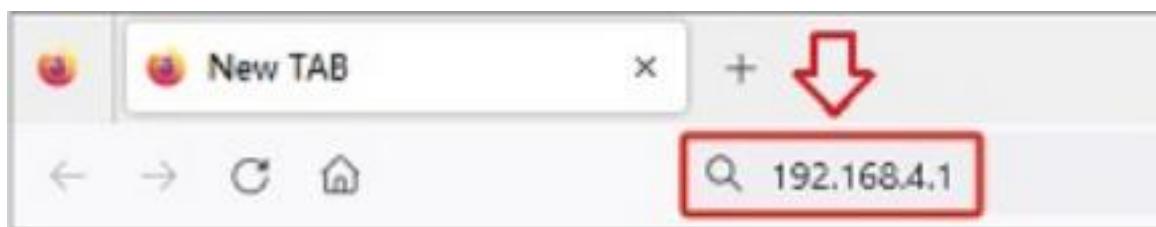


Turn on the phone's Wi-Fi and connect to the network named 'eArm'

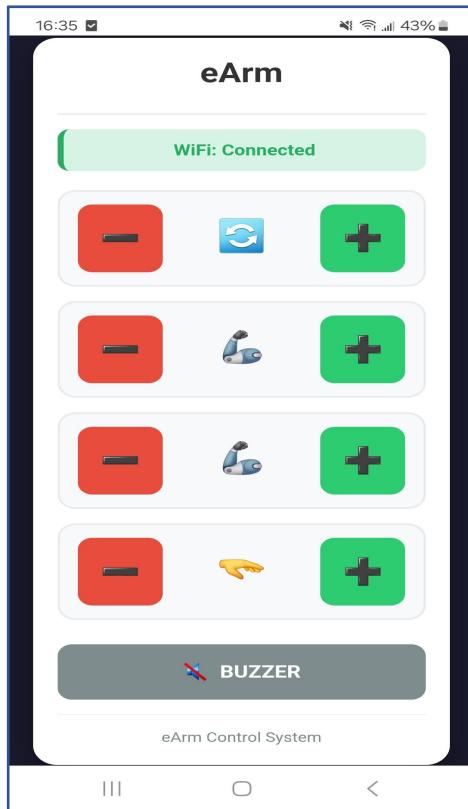
- Once connected to the Wi-Fi, you may see a 'Network connection failed' message; just disregard it.
- Step 1: Turn off mobile data in your phone settings
- Step 2: This ensures stable connection to the Web App



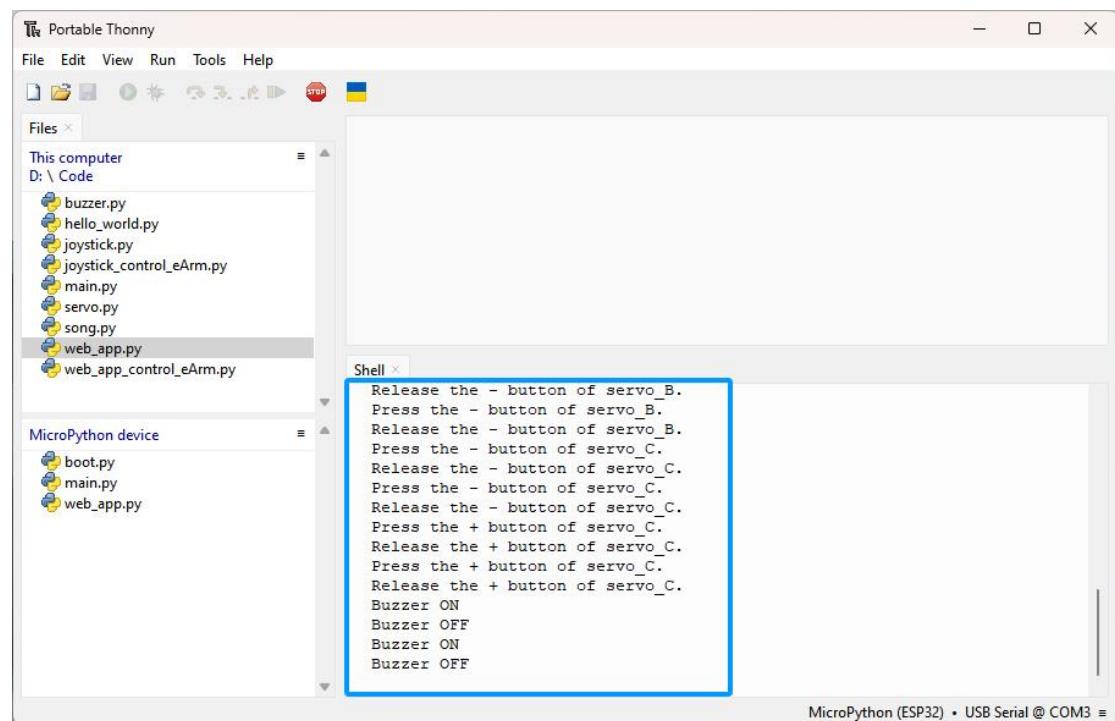
1. Open the web browser on your phone
2. Type 192.168.4.1 into the address bar
3. Press Enter to connect



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



Click the button and the shell will print the key information.



Code Explanation:

The detailed analysis of the code is in the ".py" file. Please open and read it with Thonny IDE.

The screenshot shows the Portable Thonny IDE interface with the following details:

- Title Bar:** Portable Thonny - D:\Code\web_app.py @ 249:1
- File Menu:** File, Edit, View, Run, Tools, Help
- Toolbar:** Includes icons for file operations like Open, Save, Run, Stop, and a refresh symbol.
- Left Sidebar:**
 - Files:** Shows files in "This computer" (D:\Code) and "MicroPython device".
 - This computer:** Contains files: buzzer.py, hello_world.py, joystick.py, joystick_control_eArm.py, main.py, servo.py, song.py, web_app.py (selected), and web_app_control_eArm.py.
 - MicroPython device:** Contains files: boot.py, main.py, and web_app.py.
- Code Editor:** The active tab is "web_app.py". The code is as follows:

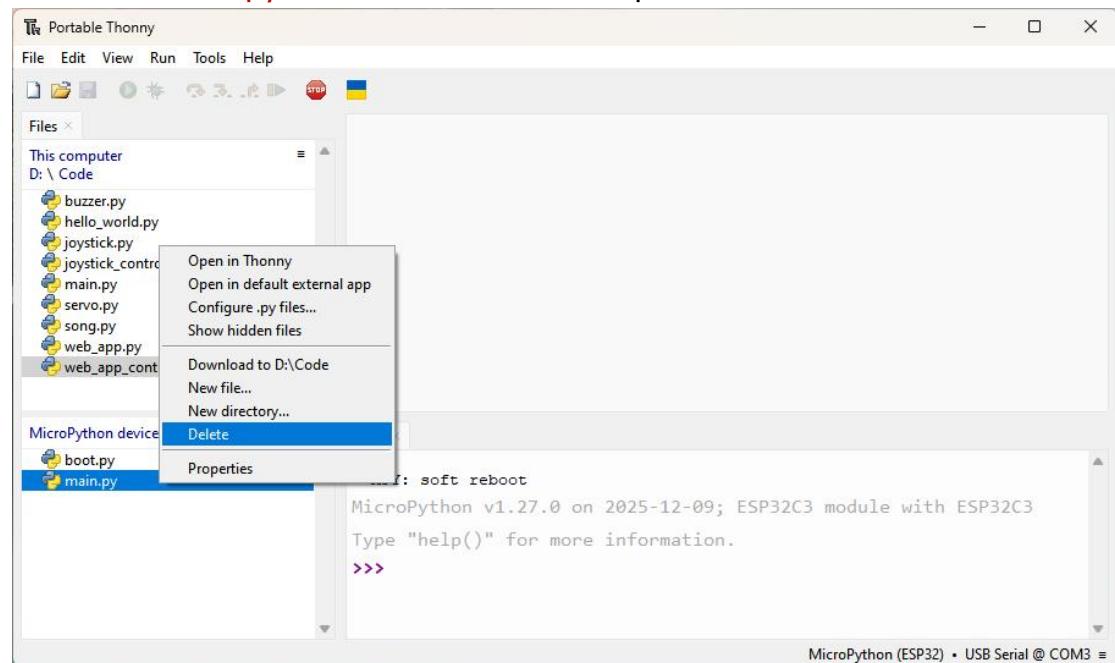
```
1 """
2 eArm Robotic Arm Web Control System
3 """
4
5 from machine import Pin
6 import time
7 import network
8 import socket
9 import gc
10 import _thread
11
12 # ===== WiFi Setup =====
13 WIFI_SSID = "eArm"
14 AP_IP = "192.168.4.1"
15
16 buzzer_state = False
17
18 def setup_wifi():
19     """Configure ESP32 as WiFi Access Point"""
20     ap = network.WLAN(network.AP_IF)
21     ap.active(True)
22     ap.config(essid=WIFI_SSID, authmode=network.AUTH_OPEN)
23     ap.ifconfig((AP_IP, '255.255.255.0', AP_IP, AP_IP))
24
25     for i in range(10):
26         if ap.active():
27             print("WiFi AP active")
28             break
29         time.sleep(0.5)
30
31     print("-" * 40)
32     print("SSID: " + WIFI_SSID)
33     print("IP: " + AP_IP)
34     print("-" * 40)
35     return ap
```

- Shell:** Displays the output of the code execution, showing the state of servos B and C.
- Status Bar:** MicroPython (ESP32) • USB Serial @ COM3 =

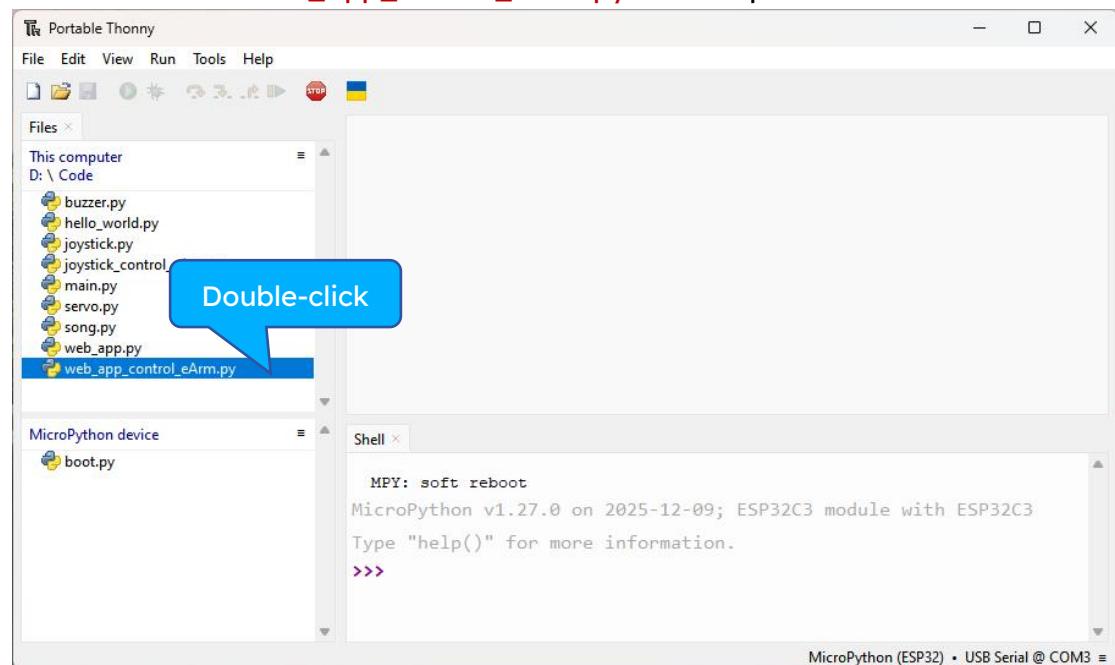
4.5.6 Web App Control eArm

By invoking other ".py" files from the script in "main.py", it is only suitable for simple code. For more powerful code, you need to delete the "main.py" file that contains the script and directly save the file you want to run as "main.py".

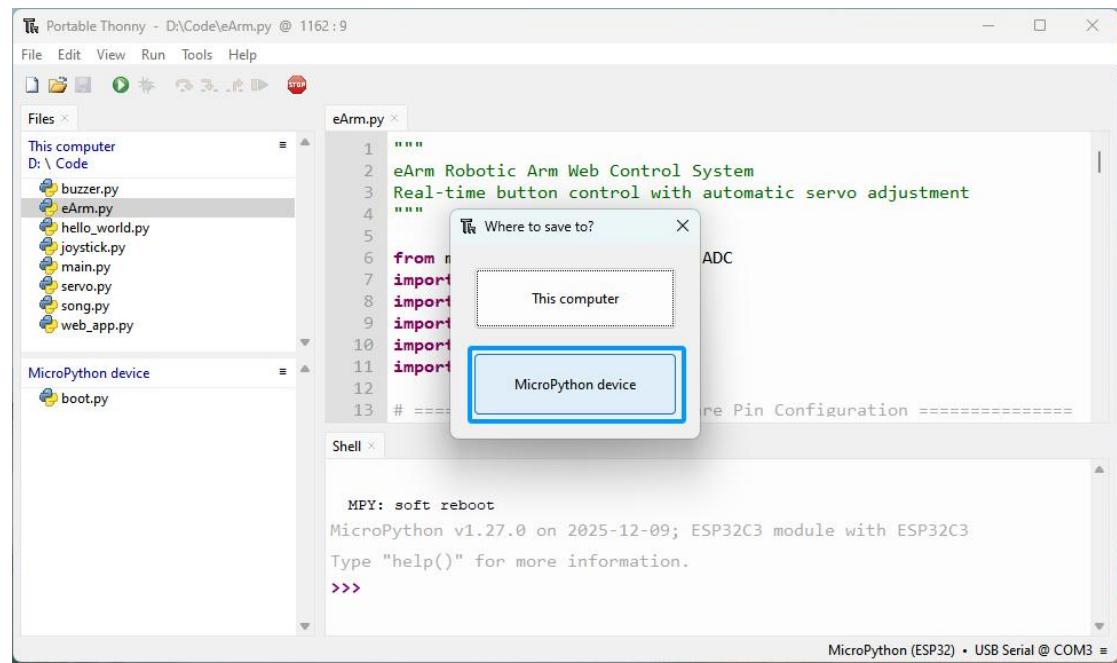
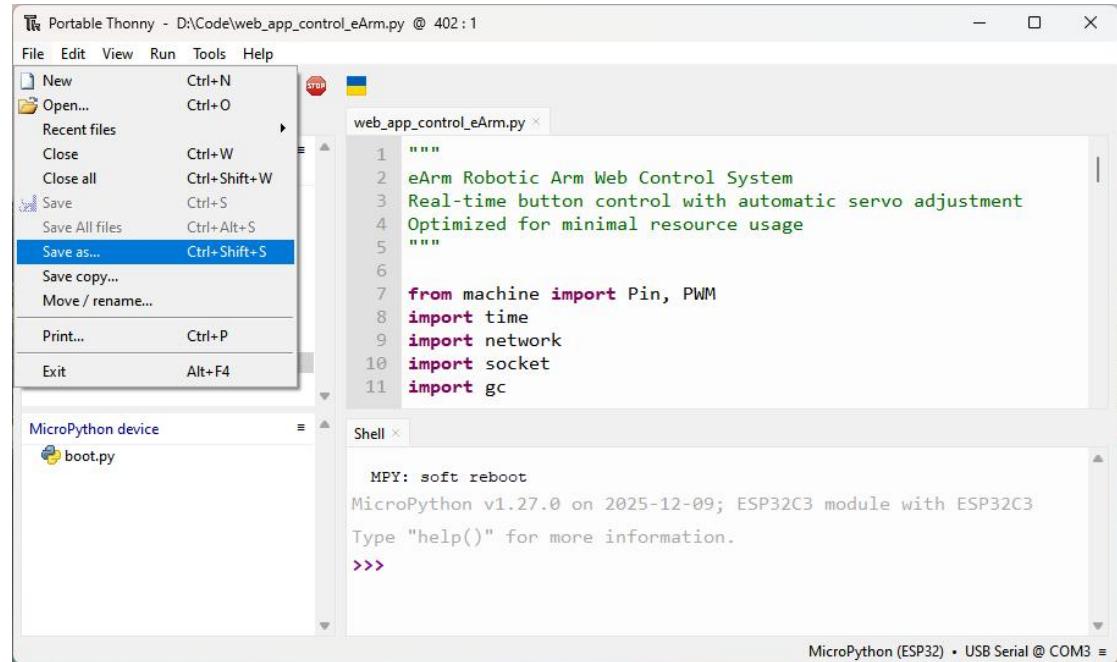
Delete the "main.py" file that contains the script:

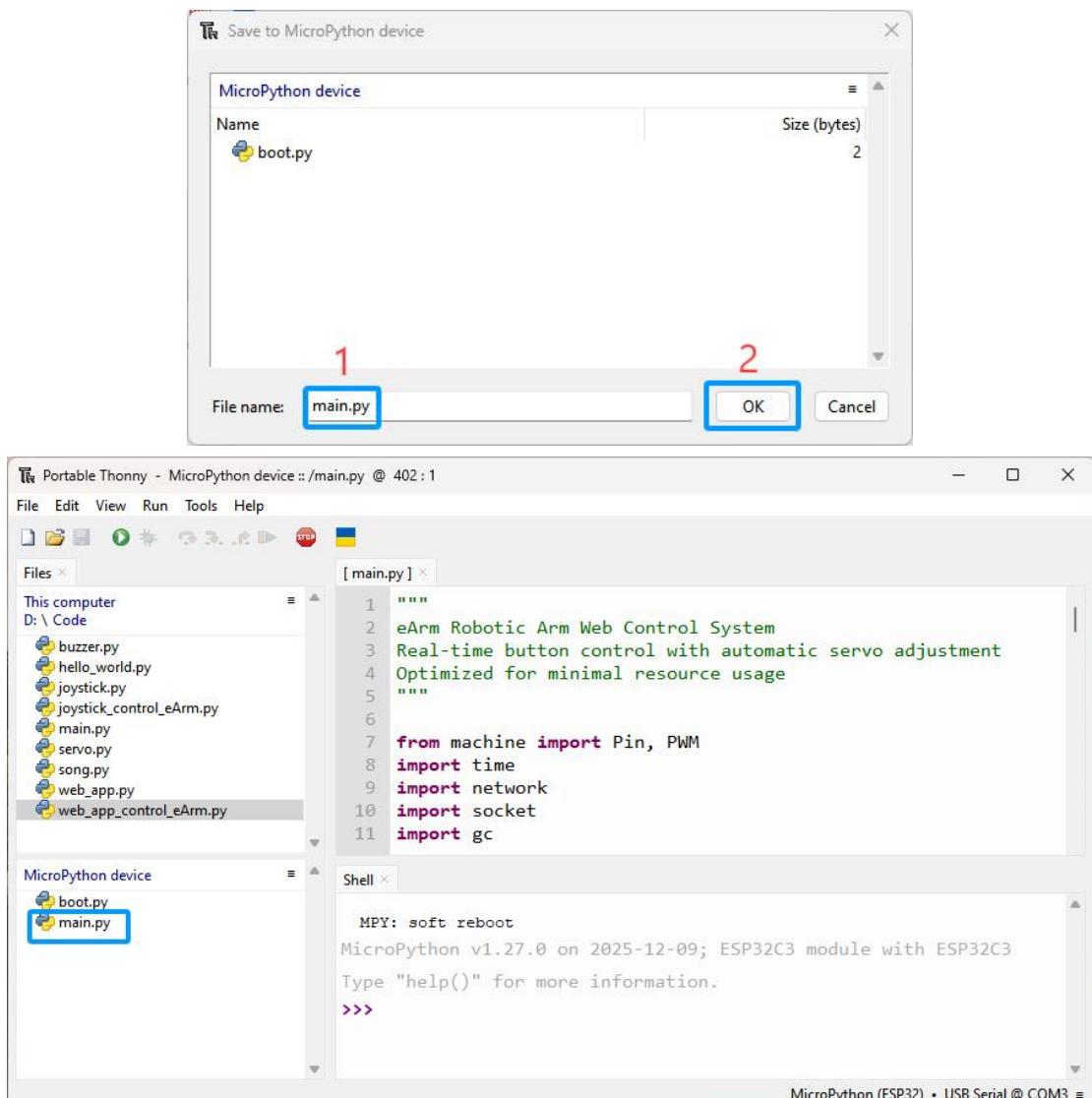


Double-click the "web_app_control_eArm.py" file to open it.

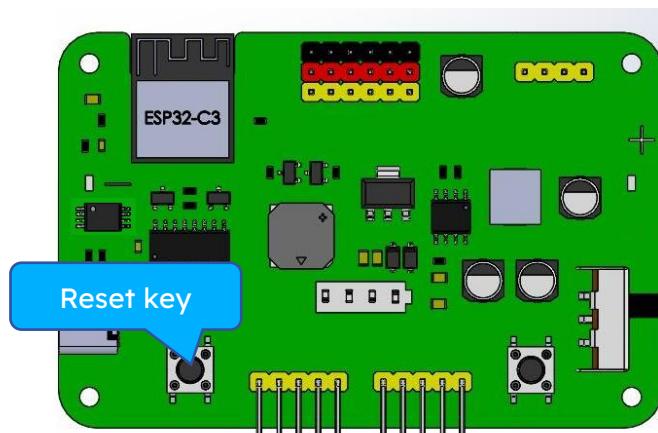


Click "File" > "Save as ...", then save "eArm.py" with the name "main.py" to the root directory of the ESP32-C3 device.



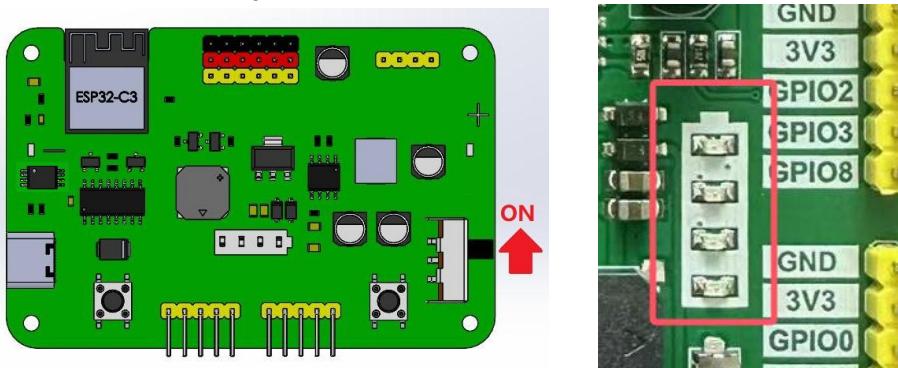


Press the reset key and in the box of the illustration below, you can see the code is executed.



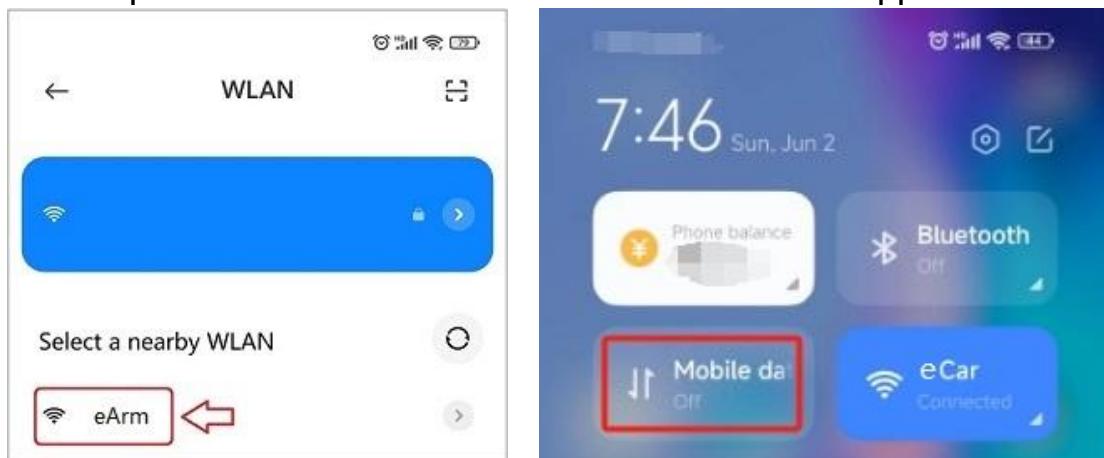
Turn the eArm's power switch to the ON position.

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

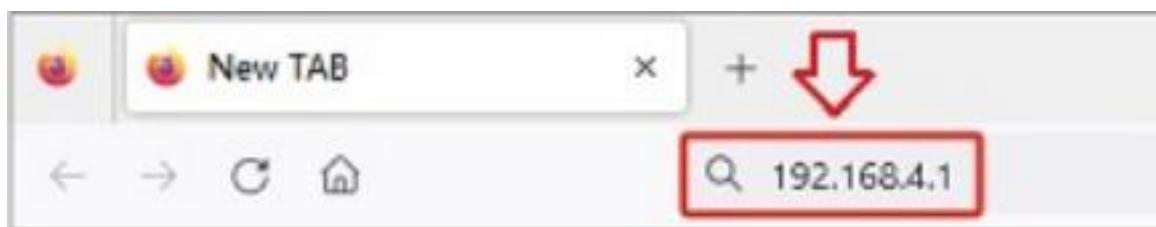


Turn on the phone's Wi-Fi and connect to the network named 'eArm'

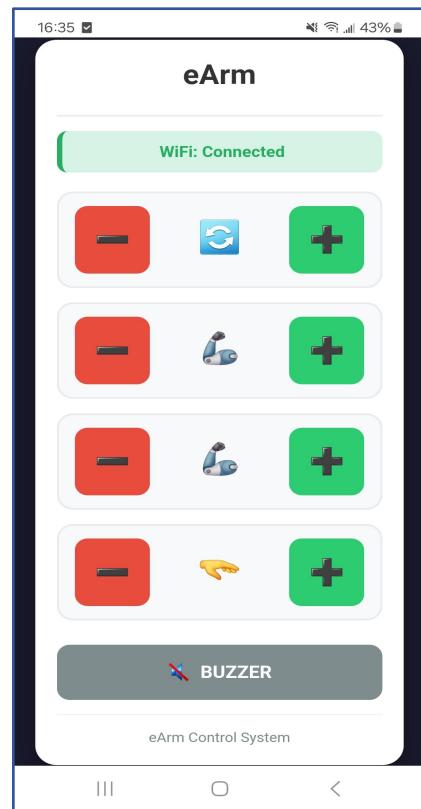
- Once connected to the Wi-Fi, you may see a 'Network connection failed' message; just disregard it.
- Step 1: Turn off mobile data in your phone settings
- Step 2: This ensures stable connection to the Web App



1. Open the web browser on your phone
2. Type 192.168.4.1 into the address bar
3. Press Enter to connect



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



		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

Code Explanation:

The detailed analysis of the code is in the ".py" file. Please open and read it with Thonny IDE.

The screenshot shows the Portable Thonny IDE interface. The top bar displays "Portable Thonny - D:\Code\web_app_control_eArm.py @ 402 : 1". The menu bar includes File, Edit, View, Run, Tools, and Help. Below the menu is a toolbar with icons for file operations and a stop button. The left sidebar has two sections: "This computer" containing files like buzzer.py, hello_world.py, joystick.py, joystick_control_eArm.py, main.py, servo.py, song.py, web_app.py, and web_app_control_eArm.py; and "MicroPython device" containing boot.py and main.py. The main workspace is titled "web_app_control_eArm.py" and contains the following Python code:

```
1 """
2 eArm Robotic Arm Web Control System
3 Real-time button control with automatic servo adjustment
4 Optimized for minimal resource usage
5 """
6
7 from machine import Pin, PWM
8 import time
9 import network
10 import socket
11 import gc
12 import _thread
13
14 # ====== Buzzer Control Class ======
15 class Buzzer:
16     """
17         Simple buzzer controller using PWM for passive buzzers
18     """
19
20     def __init__(self, pin=8):
21         """Initialize buzzer on specified GPIO pin"""
22         self.pwm = PWM(Pin(pin))
23         self.off()
24
25     def on(self, freq=1000):
26         """Turn on buzzer with specified frequency"""
27         self.pwm.freq(freq)
28         self.pwm.duty(512)
29
30     def off(self):
31         """Turn off buzzer (stop sound)"""
32         self.pwm.duty(0)
33
34     def beep(self, freq=1000, duration=200):
35         """
36             Beep for duration at frequency
37         """
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
229
230
231
232
233
234
235
236
237
238
239
239
240
241
242
243
244
245
246
247
248
249
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
888
889
889
890
891
892
893
894
895
896
897
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
948
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
978
979
980
981
982
983
984
985
986
987
987
988
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1097
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1178
1179
1179
1180
1181
1182
1183
1184
1185
1186
1187
1187
1188
1189
1189
1190
1191
1192
1193
1194
1195
1195
1196
1197
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1248
1249
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1278
1279
1279
1280
1281
1282
1283
1284
1285
1286
1287
1287
1288
1289
1289
1290
1291
1292
1293
1294
1295
1296
1297
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1348
1349
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1378
1379
1379
1380
1381
1382
1383
1384
1385
1386
1387
1387
1388
1389
1389
1390
1391
1392
1393
1394
1395
1396
1397
1397
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1418
1419
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1428
1429
1429
1430
1431
1432
1433
1434
1435
1436
1437
1437
1438
1439
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1448
1449
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1477
1478
1478
1479
1479
1480
1481
1482
1483
1484
1485
1486
1487
1487
1488
1489
1489
1490
1491
1492
1493
1494
1495
1496
1497
1497
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1537
1538
1539
1539
1540
1541
1542
1543
1544
1545
1546
1547
1547
1548
1549
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1567
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1577
1578
1578
1579
1579
1580
1581
1582
1583
1584
1585
1586
1587
1587
1588
1589
1589
1590
1591
1592
1593
1594
1595
1596
1597
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1617
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1627
1627
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1637
1637
1638
1639
1639
1640
1641
1642
1643
1644
1645
1646
1647
1647
1648
1649
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1667
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1677
1678
1679
1679
1680
1681
1682
1683
1684
1685
1686
1687
1687
1688
1689
1689
1690
1691
1692
1693
1694
1695
1696
1697
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1717
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1727
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1737
1738
1739
1739
1740
1741
1742
1743
1744
1745
1746
1747
1747
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1777
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1787
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1797
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1817
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1827
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1837
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1847
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1877
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1887
1888
1889
1889
1890
1891
1892
1893
1894
1895
1896
1897
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1917
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1927
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1937
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1947
1948
1949
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1977
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1987
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1997
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2017
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2047
2048
2049
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2067
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2077
2078
2079
2079
2080
2081
2082
2083
2084
2085
2086
2087
2087
2088
2089
2089
2090
2091
2092
2093
2094
2095
2096
2097
2097
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
210
```

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 development environment, 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:

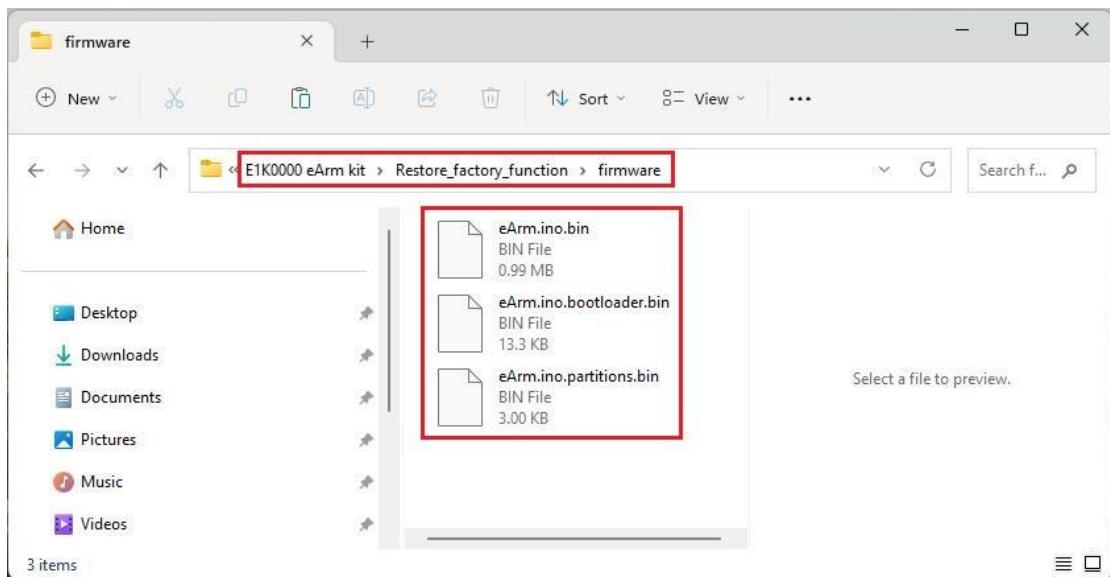
- ▶ Provide a pre-compiled firmware file (e.g., firmware.bin), allowing users to flash it with a simple tool (e.g., a serial flasher) in one step.
- ▶ No need to install Arduino IDE, resolve compilation errors, or manage library conflicts.

Use Cases:

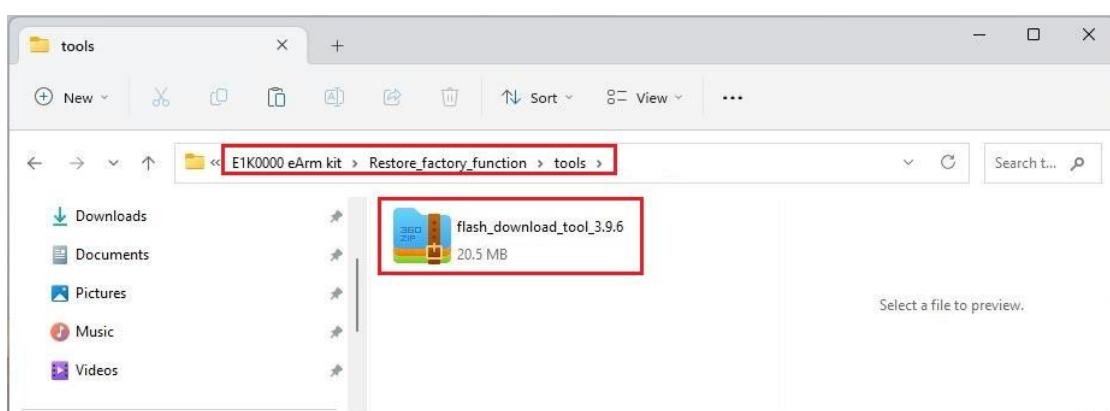
- ▶ Restoring a malfunctioning device to factory settings.
- ▶ Mass-flashing the same firmware to multiple devices (improves efficiency).

3. How to Implement This?

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



5.2 Burning Tool

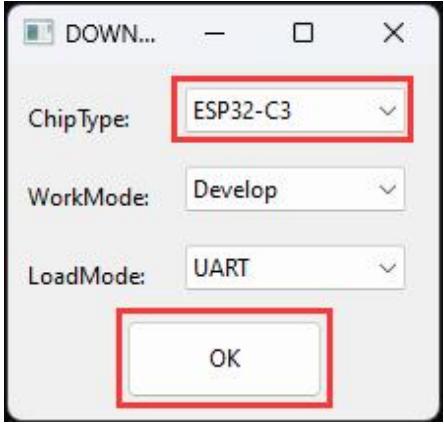


You can also download the latest version of the burning tool from the official website:

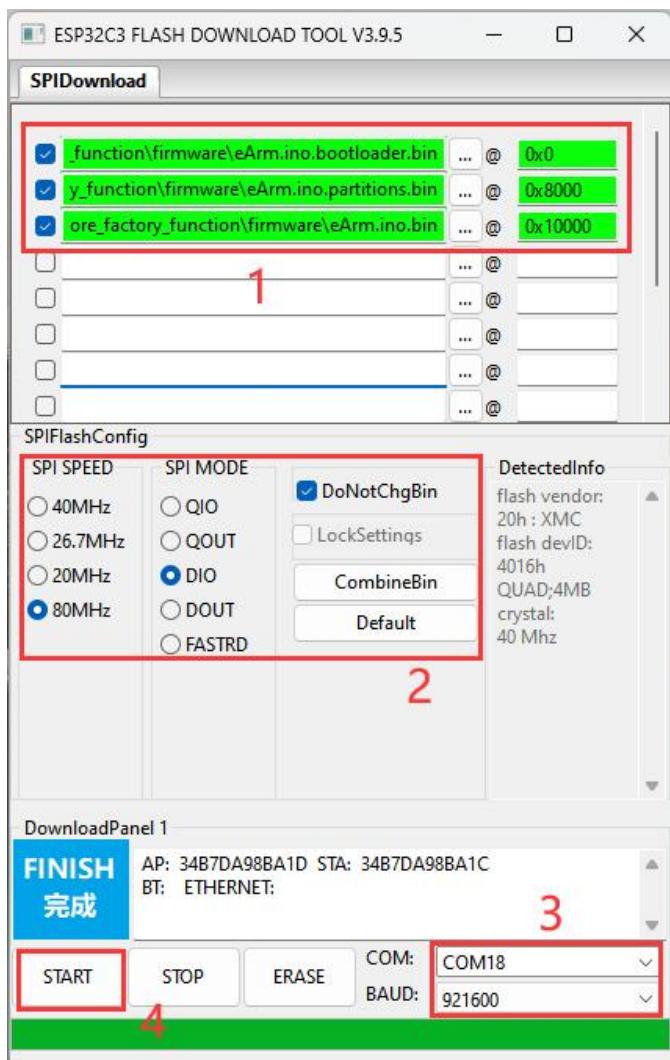
<https://www.espressif.com.cn/zh-hans/support/download/other-tools>

Flash Download Tools				
	Title	Platform	Version	Release Date
<input type="checkbox"/>	+ Flash Download Tools	Windows PC	V3.9.7	2024.06.07

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
QA

6.1 Unable to recognize USB serial ports

- 👉 Do you use a USB cable with data communication function?
- 👉 Does the USB cable connect well?
- 👉 Is the CH340 USB driver installed?

6.2 Robotic arm doesn't work

- 👉 Whether to turn on the power switch during installation to initialize the angle of the servo.
- 👉 Is the battery electricity sufficient?

6.3 Other problems

- 👉 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?

7

Contact Us

If you encounter any technical issues or wish to share feedback with us, please feel free to contact us

 support@siyeeenove.com

 <https://siyeeenove.com>