

Lesson 1 Set Development Environment

1. Install Driver

1) Open python editor under the same directory, or find in folder “Appendix/4. Software Installtion Package/Python Development”, and then click “Help/ Install Driver”.

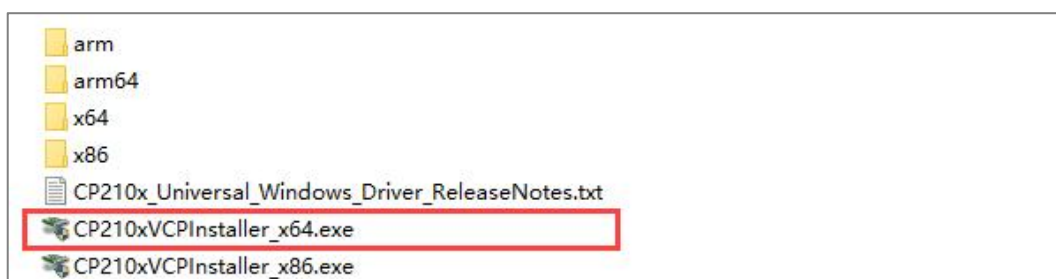


2) Find the following content in pop-up website and select driver file according to your Windows system version. Here take WIN10 for example.

2. Select the correct driver file according to the Windows operating system version:

Windows system version	driver file
10	CP210x_Windows_10_Driver.zip
XP,Vista,7,8	CP210x_VCP_Windows_XP_Vista_7_8.zip

3) Extract the downloaded driver file and double click to open driver program. Take WIN10 64-bit for example.

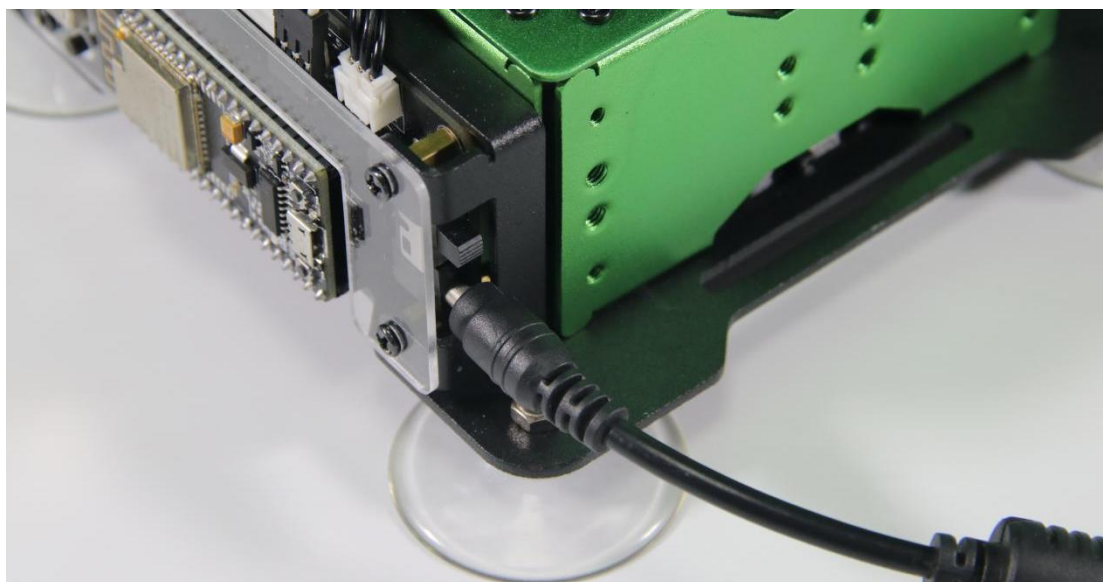


4) Then install the driver program according to the prompt.



2. Device Connection

1) Connect the adapter to MaxArm.





2) Connect the ESP32 main chip to your computer with micro-USB cable, and then turn on MaxArm.



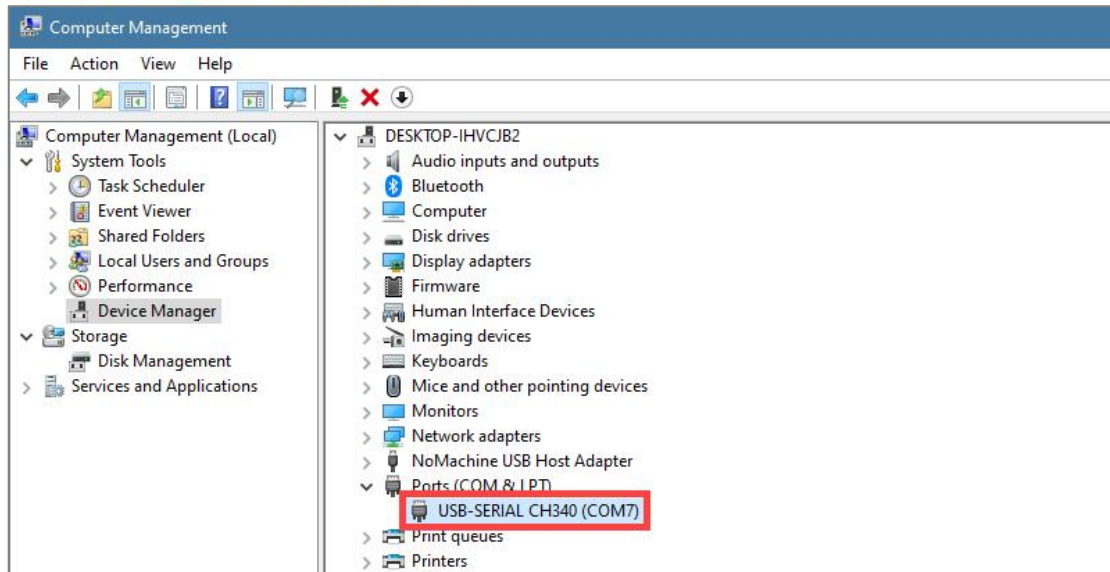
3) Open the Python editor.



4) Click on  icon to connect device. After connecting,  icon will turn green and the prompt will be showed on the lower right corner, as shown in the image below.

```
Port:COM7 Board:esp32,connect successfully !
>>>
```

Note: If multiple devices are connected to your computer through serial ports, and the failure connection is prompted, please go to "Control Panel-> Device Management" on your computer to check the connection port of ESP32. The name with "CH340" is our device.

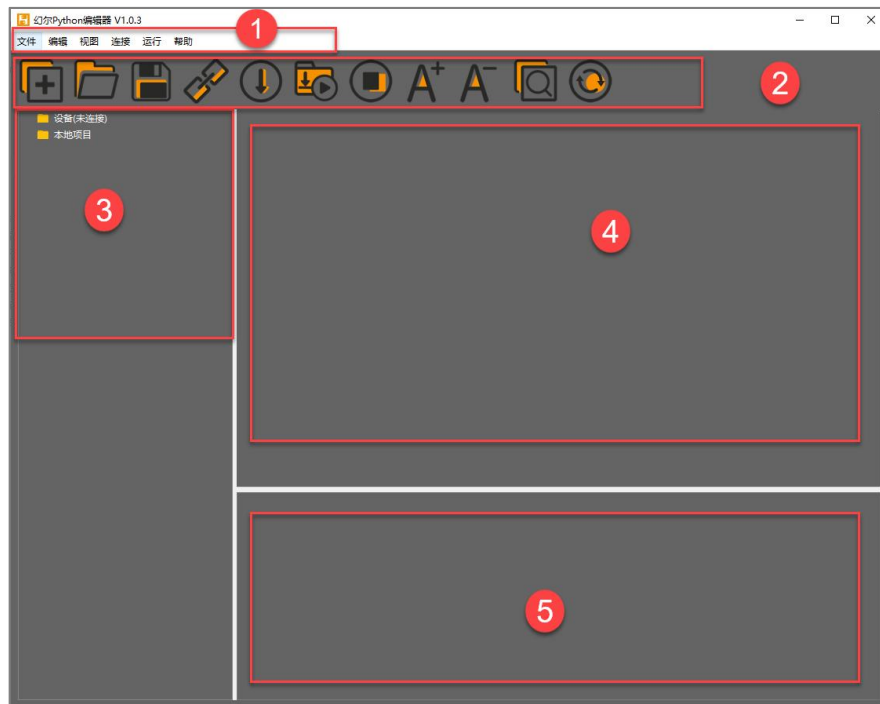


5) Then, back to the editor, and click “connect” in menu bar, and then select ESP32 port. In addition. If COM1 appears, do not select it. (COM1 is the system communication port in general).



6) If fail to connect, please refer to the method in tutorial “Appendix/5.Restore ESP32 Board Firmware” to download the firmware again. After downloading, connect the device.

3. Function Instruction



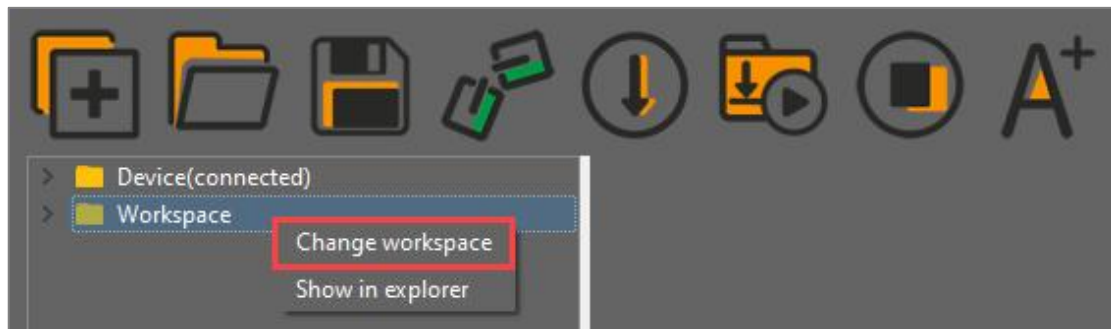
No.	Name	Function
1	Menu bar	It includes File, Edit, View, Connect, Run, Help
2	Tool bar	It includes some shortcut buttons.
3	File list	The files includes device file and local file. The contend of the project file (folder, source code, etc) can be displayed and operated.
B 4	Coding area	Display source code and the coding area.
5	Terminal	Used for message log display and debugging functions. When the device is not connected, it can only be used to view message log, and cannot be debugged.

4. Operation Instruction

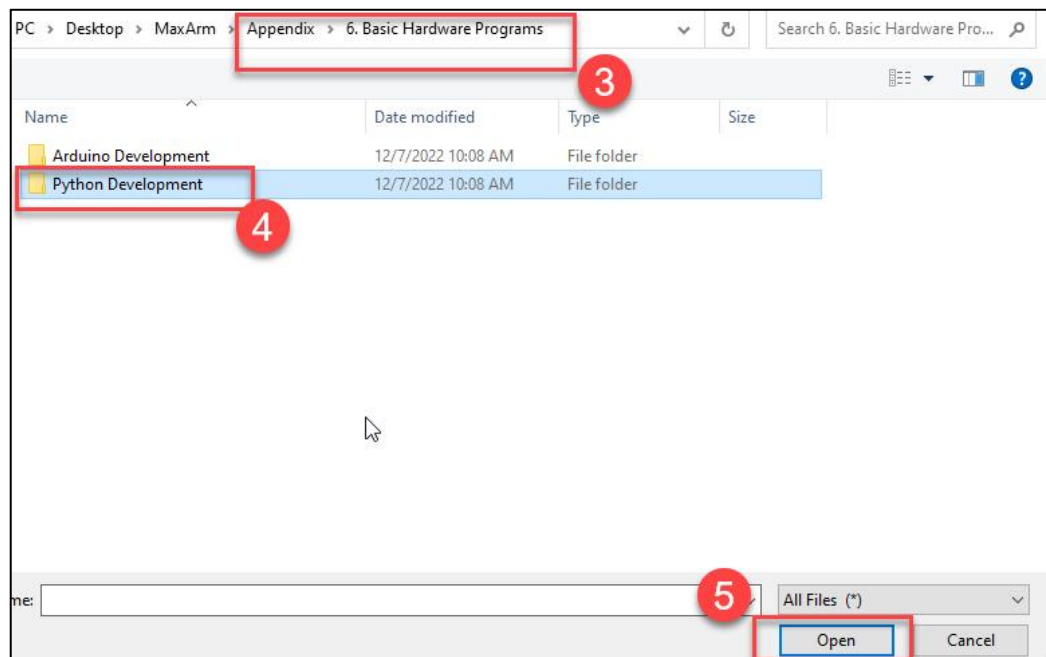
4.1 Import Local Project

Step 1: When importing the local project at the very first time, right click “Workspace” to view the file selection list.

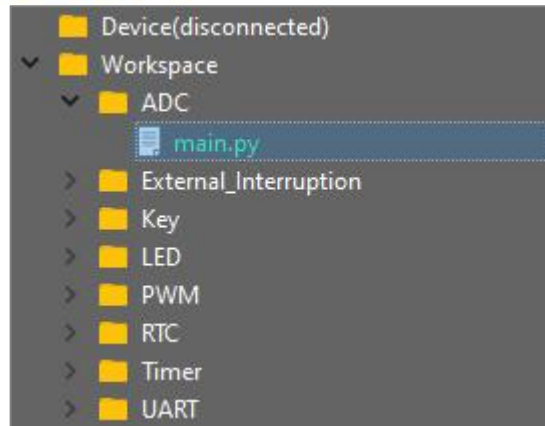
(You can directly right click “Workspace->Change Workspace” at the next import.)



Step 2: Select “Underlying Program” in “5.Appendix”, and click “select Folder”.



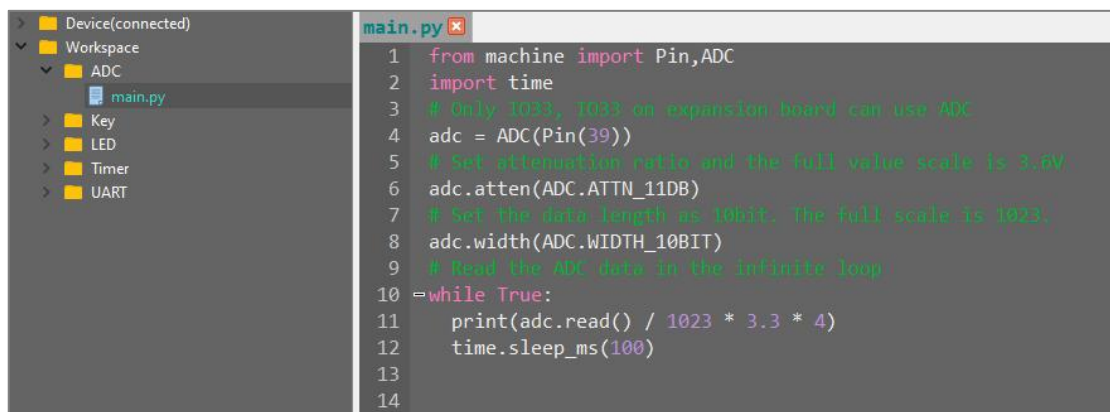
Step 3: The files in folder will be automatically added into the workspace. You can view in “Workspace”.



Note: File import is to import the file from your computer to editor, not ESP32 controller.


4.2 View the Imported File/ Program

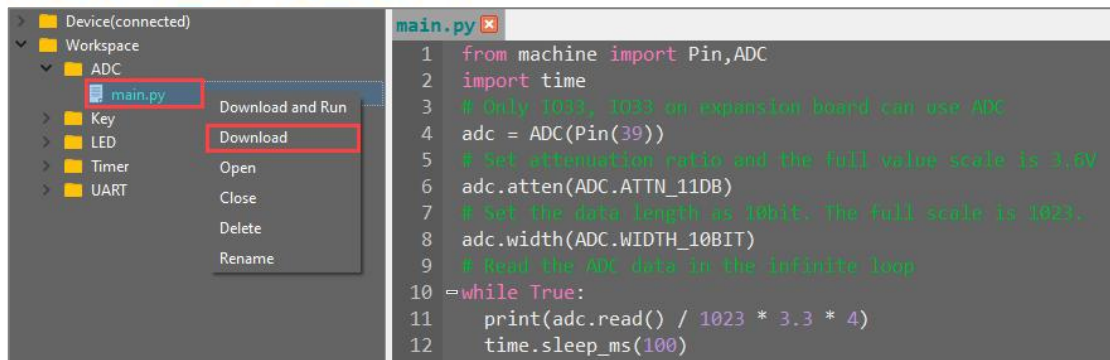
Double click the program file in file list to view the detailed code. Take file “ADC\main.py” as an example:



4.3 Download and Run Program

Program download is the interaction between editor and device. Take “ADC/main.py” as example.

- 1) Open the program “ADC/main.py” in Workspace. Click on  icon in menu bar, or directly right click and select “download” the program file.

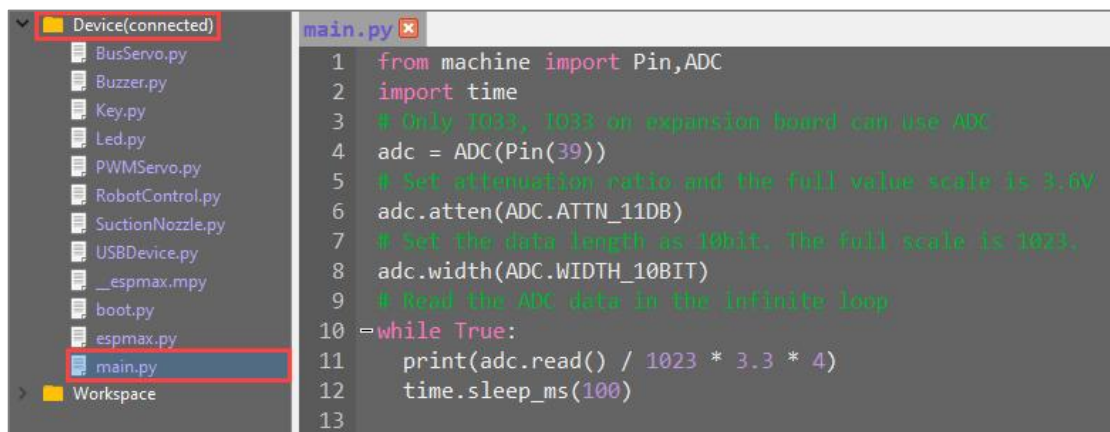


2) The download process and complete status can be viewed in terminal interface.

```

Downloading....
main.py Download ok !
>>>
    
```

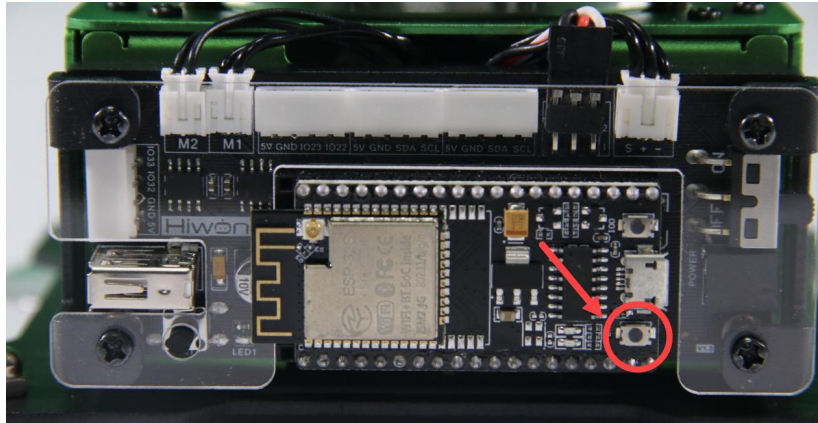
3) After downloading, the program is saved in the list of "device".



Note: You can not directly create file in "device", and the files in the "device" can only be downloaded to save the modified content.

4) Click on the reset icon or press the reset button on ESP32 controller.





- 5) If want to stop the program, click on icon. Then the terminal will print the information.



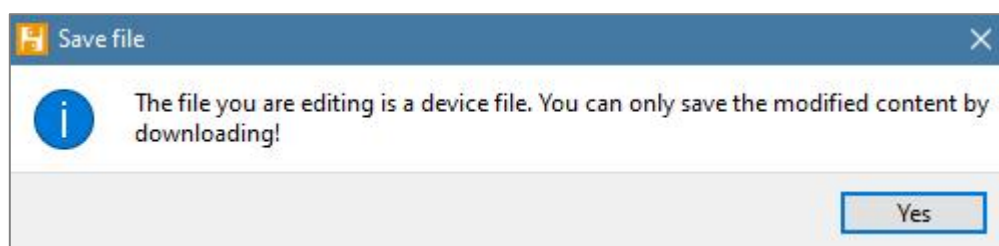
```
Traceback (most recent call last):
  File "main.py", line 12, in <module>
KeyboardInterrupt:
MicroPython v1.12-654-ge3e18b722-dirty on 2020-10-20; ESP32 module with ESP32
Type "help()" for more information.
>>> import gc
>>> gc.collect()
>>>
```

4.4 Write and Save Code

4.4.1 Notices

The coding area on the left side of interface supports the functions, such as code create, view, edit, modify and save. There are some tips for you before coding:

- 1) You cannot create file in "device" directly. After editing the file in "device", the modified contend can only be saved by downloading. If need to back up, it is recommended to copy the file to the workshop first.



2) Please do not modify the action group file with “rob” suffix in editor to avoid unknown format errors. It is recommended to do it in PC software.

4.4.2 Create Program

This section takes writing “Hello World” as example to instruct some basic operations:



1) click in the upper left corner or press Ctrl+N to create new file.

2) Input the code “print(‘Hello World’)”, as the figure shown below:

```
untitled* x
1 print('Hello World')
```

Note: Please use English punctuation.

3) Input the following code, as the figure shown below.

```
untitled* x
1 print('Hello World')
2 num = 3
3 if num == 5:
4     print('user')
```

Note: The editor supports automatic indentation of code. In the displayed code, the editor also supports associative reminders, which can be completed by pressing "Tab" or “Enter”.

At the same time, you can find that the key words, such as if, elif, are highlighted for quick visit.

4) If need to comment the code, you can select all the codes and then press “Ctrl+/" to comment all. Then select all and press “Ctrl+/" again to uncomment all.


```
untitled* x
1 # print('Hello world')
2 # num = 3
3 # if num == 5:
4 #     print('user')
5 # elif num == 2:
6 #     print('manager')
```

- 5) Step 5: Press “Ctrl” with releasing and scroll mouse wheel to zoom out or zoom in the code.

```
untitled* x
1 print('Hello World')
2 num = 3
3 if num == 5:
4     print('user')
5 elif num == 2:
6     print('manager')
```

- 6) If need to delete part of code, select and press “Delete” to delete it.



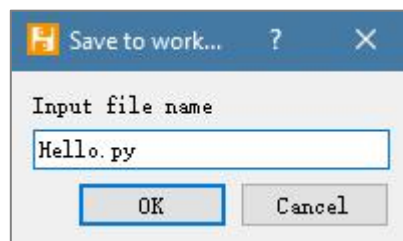
- 7) After coding, you can click  to check grammar. Then the checking result will be displayed in terminal interface.

```
untitled* x
1 print('Hello World')
2 num = 3
3 if num == 5:
4     print('user')
5 elif num == 2:
6     print('manager')
7
```

Syntax check completed,no errors

Reminder:It is recommended to check grammar after coding to examine mistake!

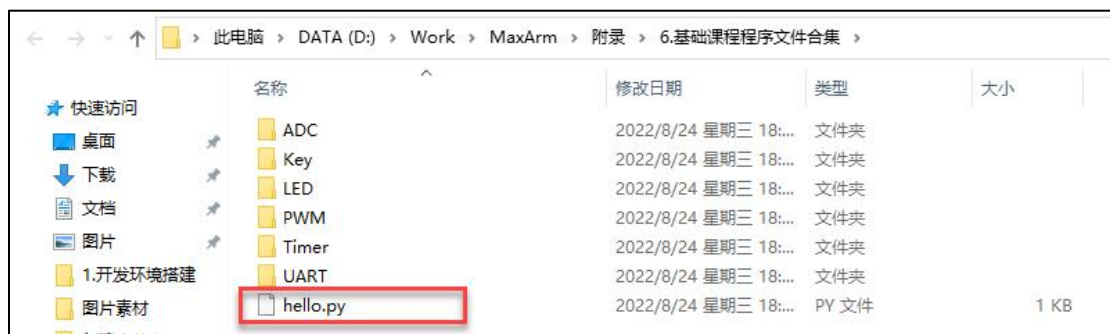
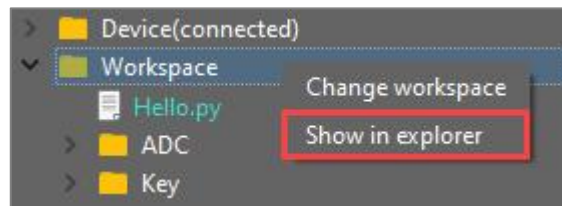
- 8) If need to save code, press “Ctrl+S” shortcut key. Then fill in file name in pop-up window and click “Ok”. It is recommended to name the file according to the specification of variable naming. Do not use Chinese and spaces to avoid parsing issues after downloading to the device.



Note:

If want to save it as main function “main.py”, it is recommended to name it to other name first, such as “test_main.py”. so that if it gets stuck when debugging the program (shortcut keys " Ctrl+C" and "Ctrl+D" are not working), you just need to reset the control board, delete and re-download the required program.

- 9) The program file is saved to workspace by default and do not support other saving path. You can right click “workspace” , then select “show in explorer” to open the directory where the file is located.



- 10) If need to change the file location, you can directly copy the file to the corresponding directory.

- 11) After saving, refer to “4.3 Download and Run Program”.

```
Downloading.
Hello.py Download ok !
>>>
```

4.5 Terminal Usage Method (Debugging)

Terminal is the function area integrating information window and debugging window. But it need to be explained that if do not connect device, terminal is for viewing information, not for editing and debugging.

About viewing information, it has been mentioned above. Here we are going to

introduce you debugging function.

- 1) The terminal supports code input. Input “print(123)” in terminal and press “Enter”. The effect is as follow:

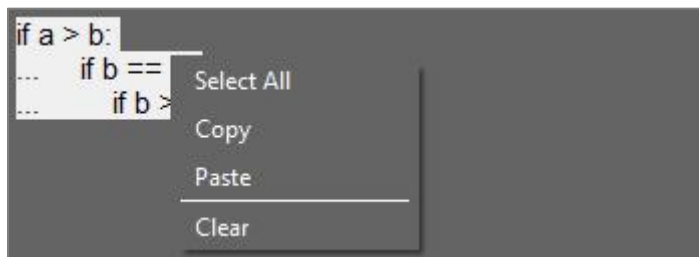
```
print(123)
123
>>>
```

- 2) In addition, the terminal supports auto-indentation as well.

When inputting python statement ending with colon (for example, if, for, while), the prompt will change to “...” and the cursor will be indented 4 spaces. When pressing “Enter”, the next line will continue to indent 4 spaces as a regular statement based on the previous line, or indent at different level. If press backspace, it will undo one level of indentation.

```
if a > b:
...     if b == 0:
...         if b > c:
```

- 3) If need to copy the code, right-click to operate in terminal after selecting the target code.



Please note that, since the terminal has auto-indentation function, you need to press “Ctrl+E” to enter editing mode before pasting code. Otherwise, indentation error will occur during debugging.

As the figure shown below, it is the correct copied indentation format.


```
Hello.py x
1 print('Hello World')
2 num = 3
3 if num == 5:
4     print('user')
5 elif num == 2:
6     print('manager')
7

paste mode; Ctrl-C to cancel, Ctrl-D to finish
=== print('Hello World')
=== num = 3
=== if num == 5:
===     print('user')
=== elif num == 2:
===     print('manager')
```

The following figure shows the wrong indentation format:

```
Hello.py x
1 print('Hello World')
2 num = 3
3 if num == 5:
4     print('user')
5 elif num == 2:
6     print('manager')
7

print('Hello World')
Hello World
>>> num = 3
>>> if num == 5:
...     print('user')
... elif num == 2:
...     print('manager')
```

If need to exit the editing mode, you can press “Ctrl+C”. In addition, if accidentally write an endless loop, you can press “Ctrl+C” to exit.

Reminder: In terminal, the shortcut keys “Ctrl+C” can only be used to interrupt the running program instead of copying, and “Ctrl+V” has no pasting function.

- 4) When entering command in terminal, “Tab” key can be used to automatically fill the code. For example, entering “os” and then pressing “Tab” key. The effect is as follow:

```
>>> os
__class__  __name__  ADC      Pin
gc         time    uos      bdev
adc
>>> os
```

If there are multiple types, they will be listed; if there is only one type, it will be automatically filled; If the situations both do not exist, it will be useless.

- 5) After entering commands in terminal, you can view records through pressing “↑” “↓” on keyboard, which can save the time of entering commands.

More commands and command instruction, please visit
“<http://docs.micropython.org/en/latest/library/uos.html>”.