

Getting Started with MicroPython on ESP32 and ESP8266

Learn how to get started with MicroPython firmware on the ESP32 and ESP8266. We'll introduce you to MicroPython, show you the differences between MicroPython and regular Python, and how to program your ESP based boards with MicroPython using uPyCraft IDE. After completing this guide, you'll have your first LED blinking using MicroPython.



What is MicroPython?

MicroPython is a re-implementation of Python 3 targeted for microcontrollers and embedded systems. MicroPython is very similar with regular Python. So, if you already know how to program in Python, you also know how to program in MicroPython.



Python vs MicroPython

Apart from a few exceptions, the language features of Python are also available in MicroPython. The biggest difference between Python and MicroPython is that MicroPython was designed to work under constrained conditions.



Because of that, MicroPython does not come with the full standard library. It only includes a small subset of the Python standard library. However, it does include modules to access low-level hardware – this means that there are libraries to easily access and interact with the GPIOs.

Additionally, devices with Wi-Fi capabilities like the ESP8266 and ESP32 include modules to support network connections.

Why MicroPython?

Python is one of the most widely used, simple and easy-to-learn programming languages around. So, the emergence of MicroPython makes it extremely easy and simple to program digital electronics. If you've never programmed digital electronics before, MicroPython is a good starting point.

MicroPython's goal is to make programming digital electronics as simple as possible, so it can be used by anyone. Currently, MicroPython is used by hobbyists, researchers, teachers, educators, and even in commercial products. The code for blinking an LED on a ESP32 or ESP8266 is as simple as follows:

```
# Complete project details at https://RandomNerdTutorials.com
from machine import Pin
from time import sleep

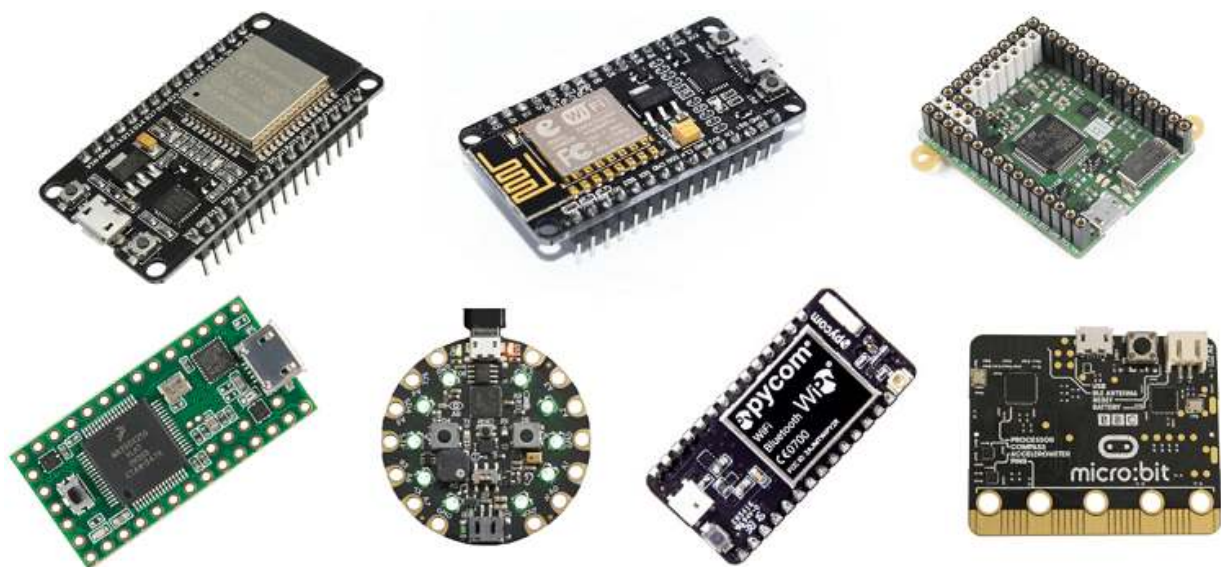
led

= Pin(2, Pin.OUT)
while True:
    led.value(not led.value())
    sleep(0.5)
```

One great feature of MicroPython is that it comes with an interactive REPL (Read-Evaluate-Print Loop). The REPL allows you to connect to a board and execute code quickly without the need to compile or upload code.

MicroPython – Boards support

MicroPython runs on many different devices and boards, such as:



- [ESP32](#)
- [ESP8266](#)
- PyBoard
- Micro:Bit
- Teensy 3.X
- WiPy – Pycom
- Adafruit Circuit Playground Express
- Other ESP32/ESP8266 based boards

For more information about other boards that support MicroPython, take a look at the following links:

- [Boards running MicroPython – MicroPython Forum](#)
- [Boards summary – MicroPython Github](#)

In our projects, we'll use MicroPython with the ESP32 and ESP8266 boards.

ESP32 is the successor of the ESP8266. So, at the moment, not all features are available in MicroPython to take the most out of the ESP32 – it's still an ongoing project. However, it's very usable and you can make a lot of projects with it.

ESP32 and ESP8266 boards are similar, and you won't feel almost any difference programming them using MicroPython. This means that anything you write for the ESP8266 should also run with no changes or minimal changes on the ESP32 (mainly changing the pin assignment).

Installing uPyCraft IDE

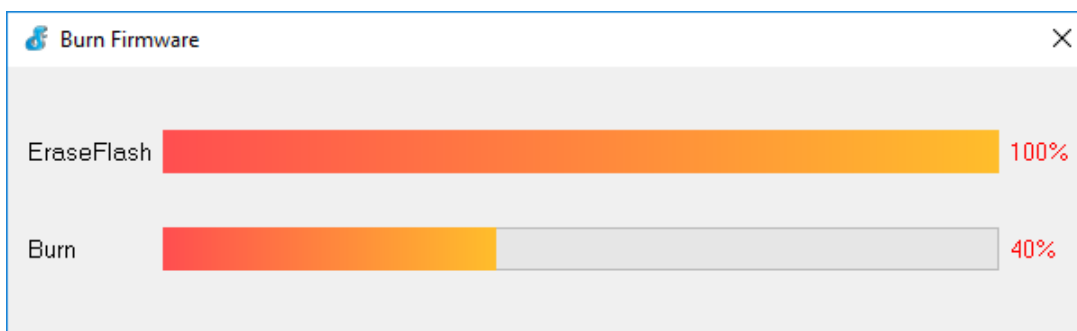
Before continuing with this tutorial, you should install uPyCraft IDE in your computer. Follow one of the next tutorials to install uPyCraft IDE:

- [Install uPyCraft IDE – Windows PC](#)
- [Install uPyCraft IDE – Mac OS X](#)
- [Install uPyCraft IDE – Linux Ubuntu](#)

Flashing MicroPython Firmware to ESP32/ESP8266

Unlike other boards, MicroPython isn't flashed onto the ESP32 or ESP8266 by default. That's the first thing you need to do to start programming your boards with MicroPython: flash/upload the firmware. Follow the next tutorial to flash MicroPython firmware on your board:

- [Flash/Upload MicroPython Firmware to ESP32 and ESP8266](#)



Getting Started with uPyCraft IDE

In this section we'll give you an overview of the uPyCraft IDE software, so that you can start programming the ESP32/ESP8266 with MicroPython.

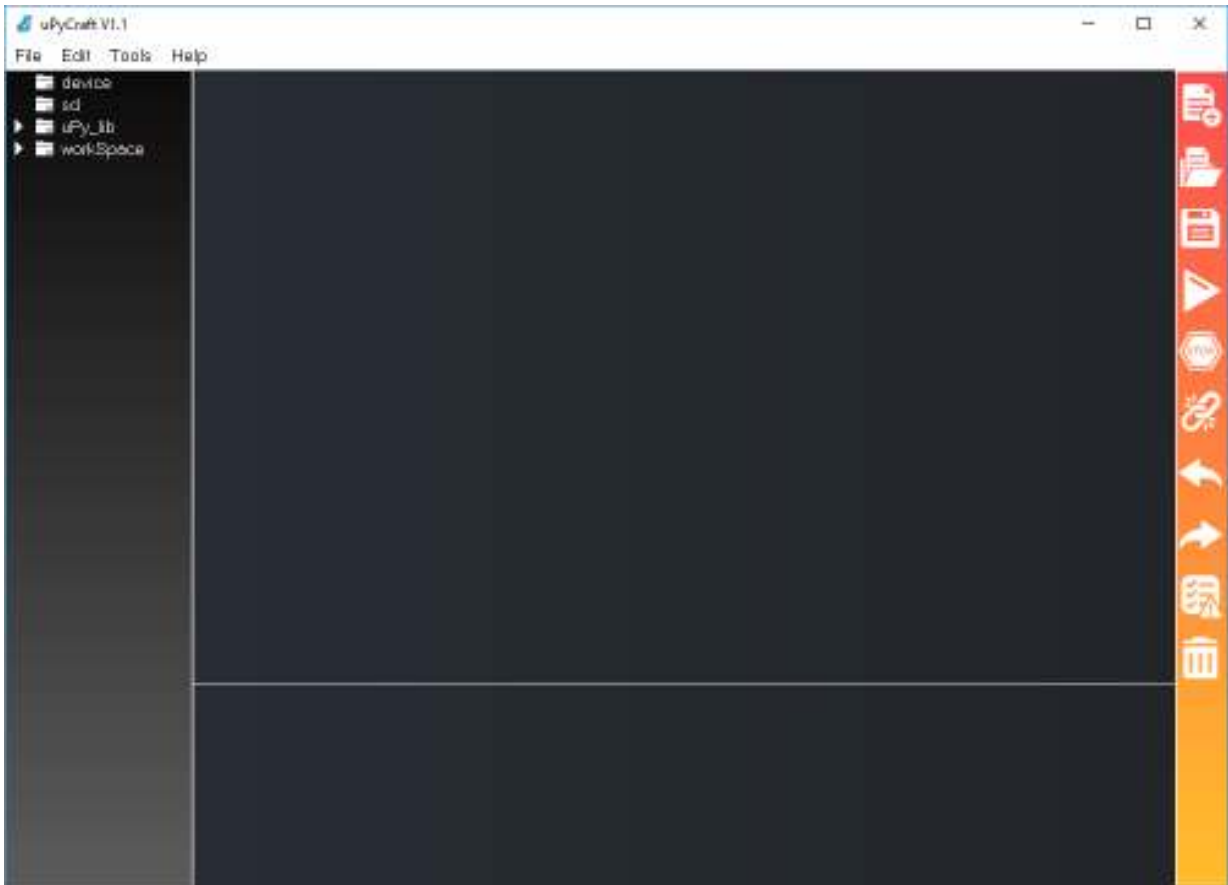
The IDE is a software that contains tools to make the process of development, debugging and upload code easier. There are many ways to program your ESP board with MicroPython. We've chosen uPyCraft IDE because it is simple and intuitive to use and works great with the ESP boards.

At this point, we assumed that you have:

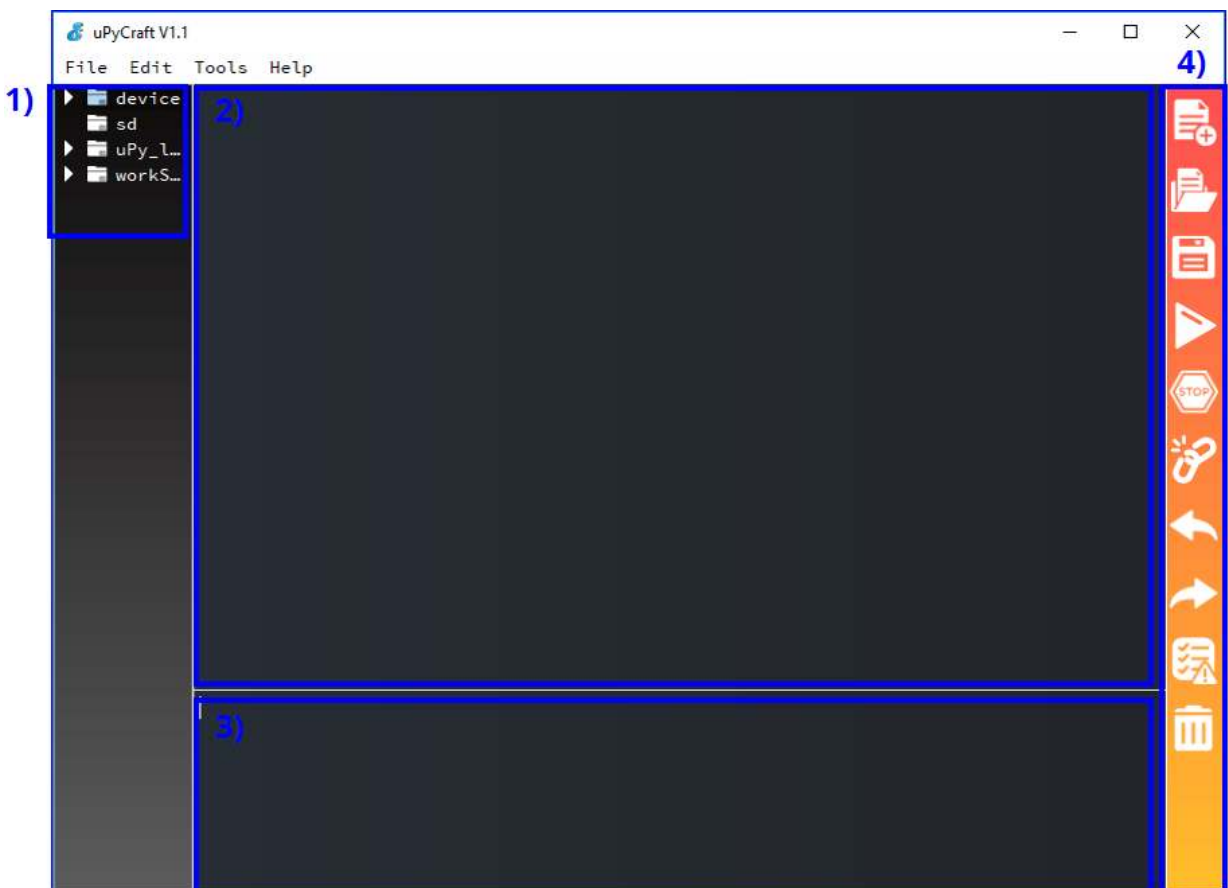
- uPyCraft IDE installed on your computer
- ESP32/ESP8266 flashed with MicroPython firmware

uPyCraft IDE Overview

Open uPyCraft IDE, a new window opens as follows:



Let's take a closer look at each section of uPyCraft IDE:



1. Folder and files
2. Editor
3. MicroPython Shell/Terminal
4. Tools

1. Folder and files

This section shows several folders and files. The **device** folder shows the files that are currently stored on your ESP board. If you have your ESP32 or ESP8266 connected via serial to uPyCraft IDE, when you expand the **device** folder, all files stored should load. By default, you should only have a *boot.py* file. To run your main code, it is recommended to create a *main.py* file.

- **boot.py**: runs when the device starts and sets up several configuration options;
- **main.py**: this is the main script that contains your code. It is executed immediately after the *boot.py*.

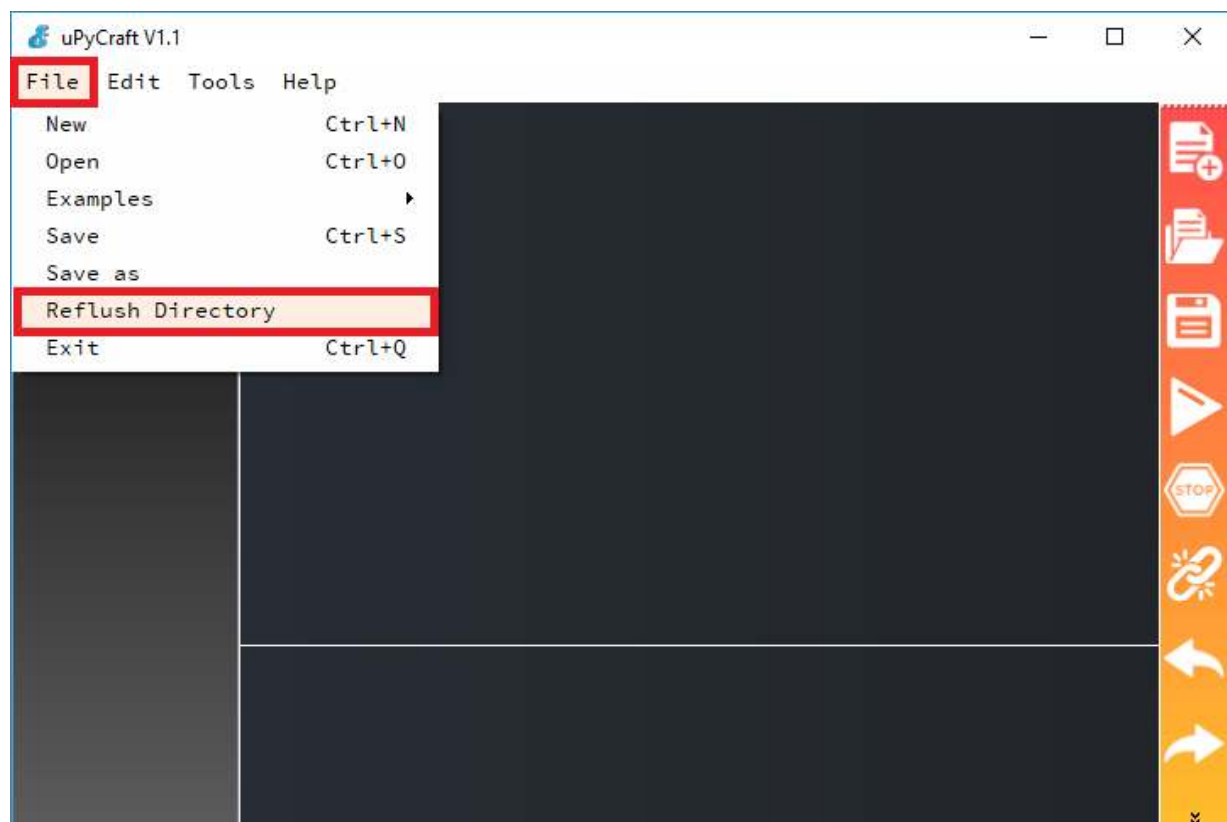
The **sd** folder is meant to access files stored on SD cards – this is only works with boards like the PyBoard that come with an SD card slot.

The **uPy_lib** shows the built-in IDE library files.

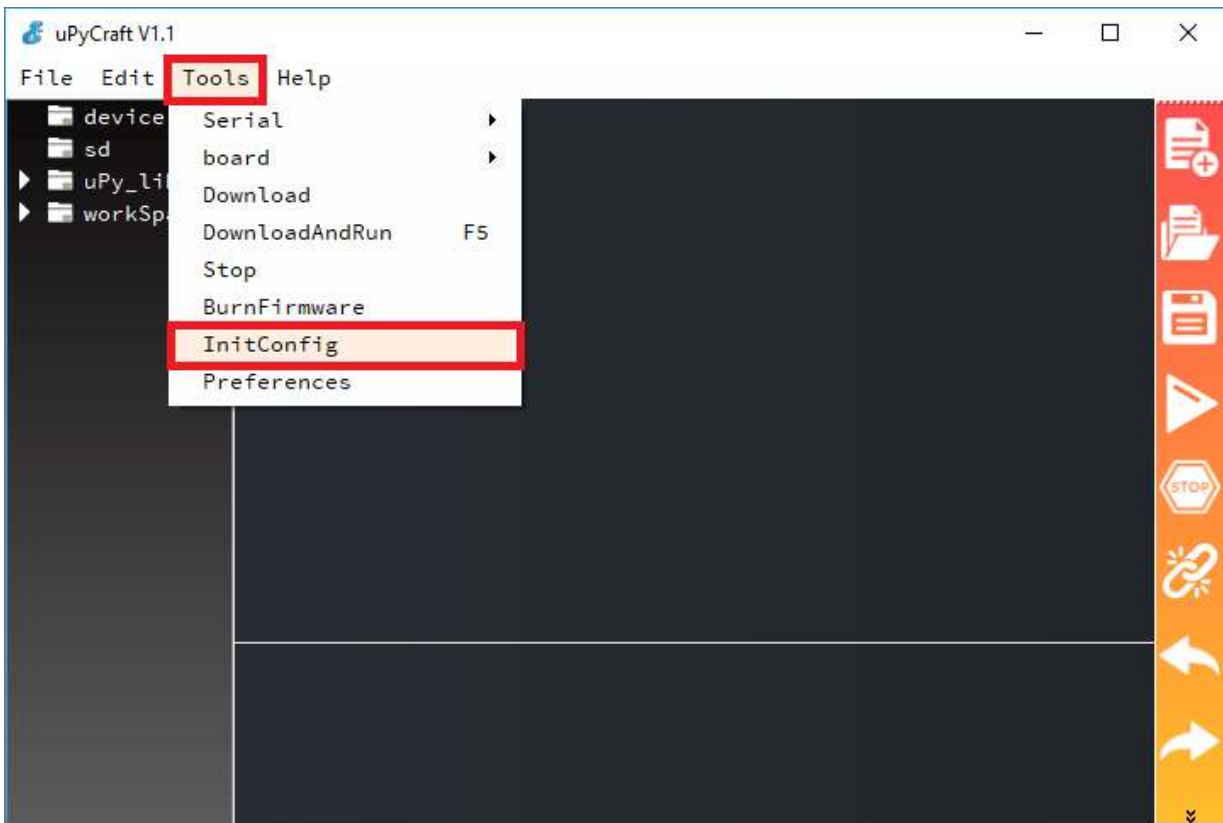
Finally, the **workspace** is a directory to save your files. These files are saved in your computer in a directory defined by you. This is a specially useful to keep all your files organized at hand.

When using uPycraft for the first time, to select your working directory, click the **workspace** folder. A new window pops up for you to chose your **workspace** path. Create a new folder or select an existing folder to be your working directory.

Then, go to **File > Reflush Directory** to update the directory.



Note: to change your user directory, simply go to **Tools > InitConfig** and click the **workspace** directory folder to chose a different path.



2. Editor

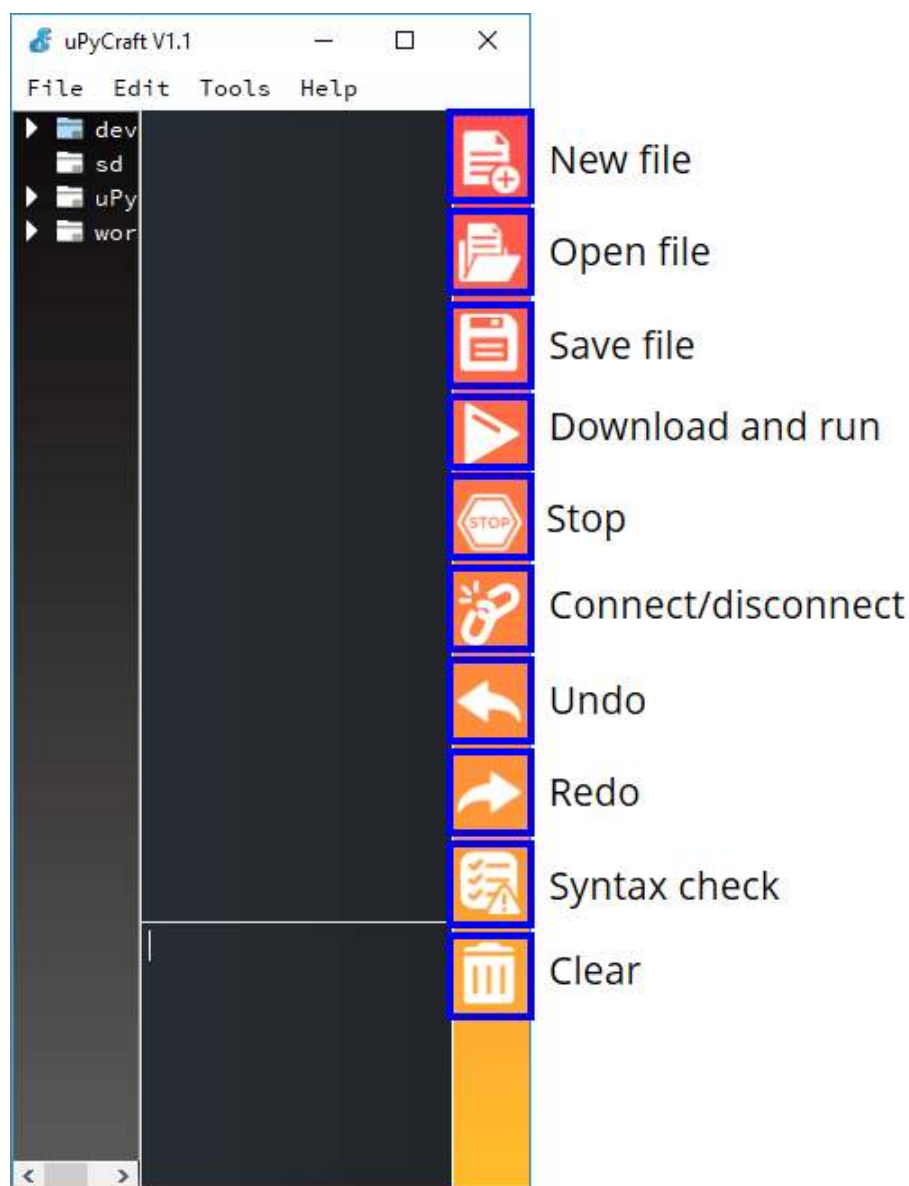
The Editor section is where you write your code and edit your `.py` files. You can open more than one file, and the Editor will open a new tab for each file.

3. MicroPython Shell/terminal

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

4. Tools

The icons placed at the rightmost side allow you to quickly perform tasks. Each button is labeled in the figure below:



- **New file:** creates a new file on the Editor;
- **Open file:** open a file from your computer;
- **Save file:** saves a file;
- **Download and run:** upload the code to your board and execute the code;
- **Stop:** stop the execution of the code – it's the same as entering CTRL+C on the Shell to stop all scripts from running;
- **Connect/Disconnect:** connect or disconnect to your board via Serial. You must select the serial port first in **Tools > Serial**;
- **Undo:** undo last change in the code Editor;
- **Redo:** redo last change in the code Editor;
- **Syntax check:** checks the syntax of your code;
- **Clear:** clear the Shell/terminal window messages.

Running Your First Script

To get you familiar with the process of writing a file and executing code on your ESP32/ESP8266 boards, we'll upload a new script that simply blinks the on-board LED of your ESP32 or ESP8266.

Establishing a communication with the board

After having the MicroPython firmware installed on your board and having the board connected to your computer through an USB cable, follow the next steps:

1. Go to **Tools > Board** and select the board you're using.
2. Go to **Tools > Port** and select the com port your ESP is connected to.
3. Press the **Connect** button to establish a serial communication with your board.



Connect/disconnect

4. The `>>>` should appear in the Shell window after a successful connection with your board. You can type the print command to test if it's working:

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

It should print the “Hello” message. Only if you see that message, you can continue with this tutorial. Otherwise, make sure you have established a serial communication with your board or that you've flashed successfully the MicroPython firmware on your board.



Creating the *main.py* file on your board

1. Press the “**New file**” button to create a new file.



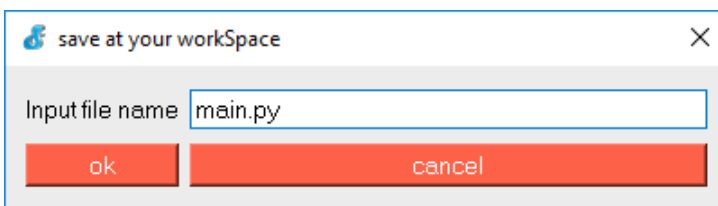
New file

2. Press the “**Save file**” button to save the file in your computer.

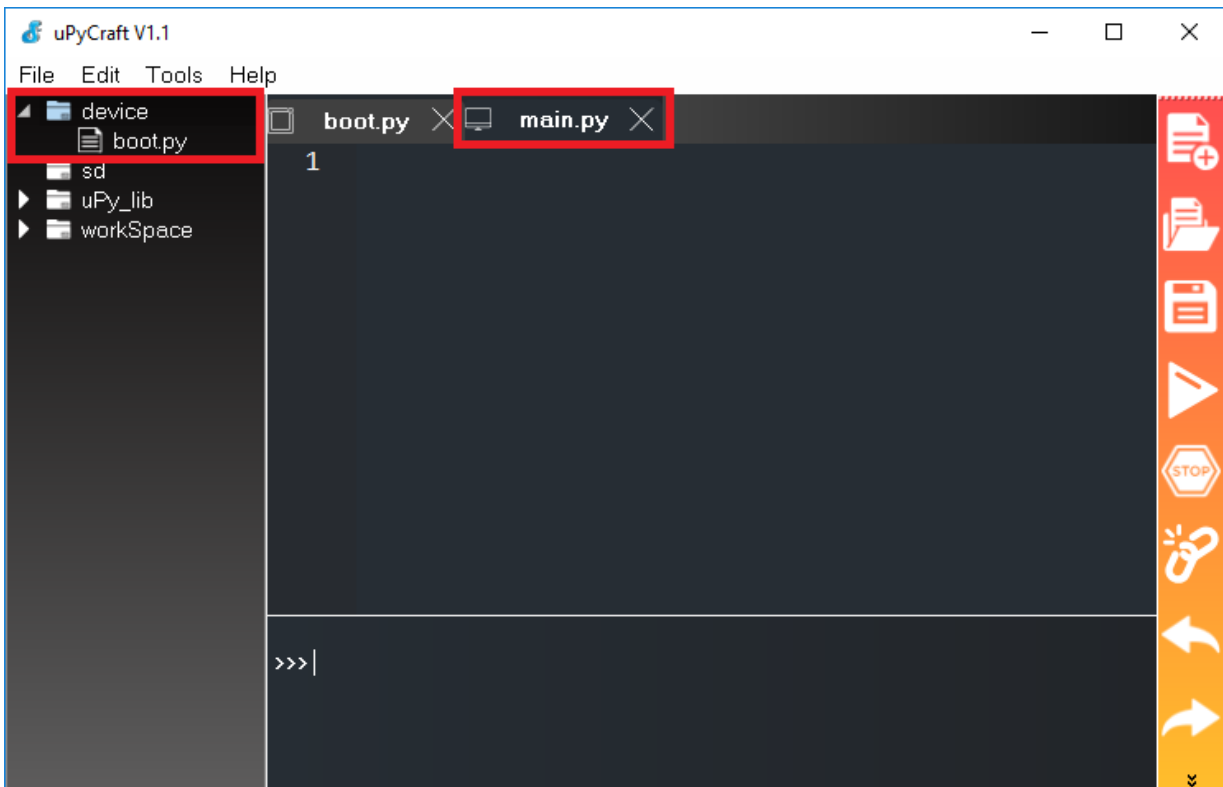


Save file

3. A new window opens, name your file *main.py* and save it in your computer:



4. After that, you should see the following in your uPyCraft IDE (the *boot.py* file in your device and a new tab with the *main.py* file):

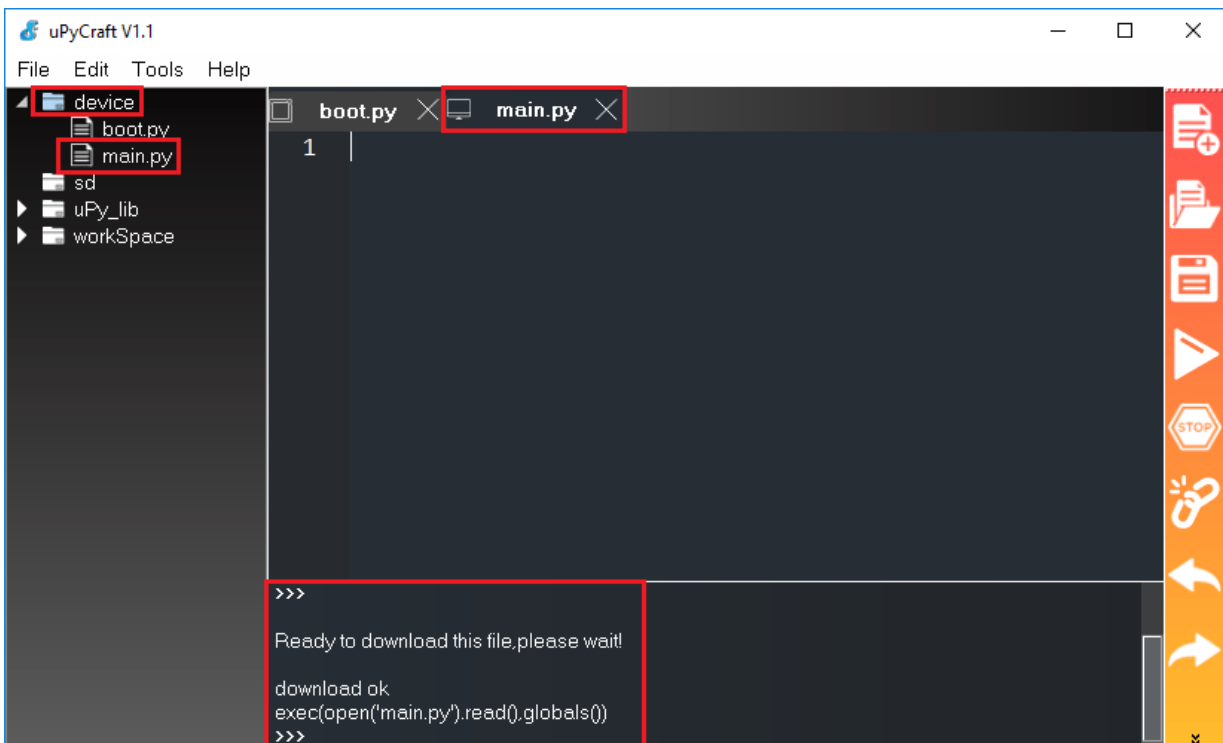


5. Click the **“Download and run”** button to upload the file to your ESP board:



Download and run

6. The device directory should now load the *main.py* file. Your ESP has the file *main.py* stored.



Uploading the blink LED script

1. Copy the following code to the Editor on the *main.py* file:

```
# Complete project details at https://RandomNerdTutorials.com
from machine import Pin
from time import sleep
```

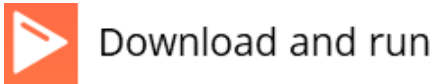
```
led

= Pin(2, Pin.OUT)while True:
    led.value(not led.value())
    sleep(0.5)
```

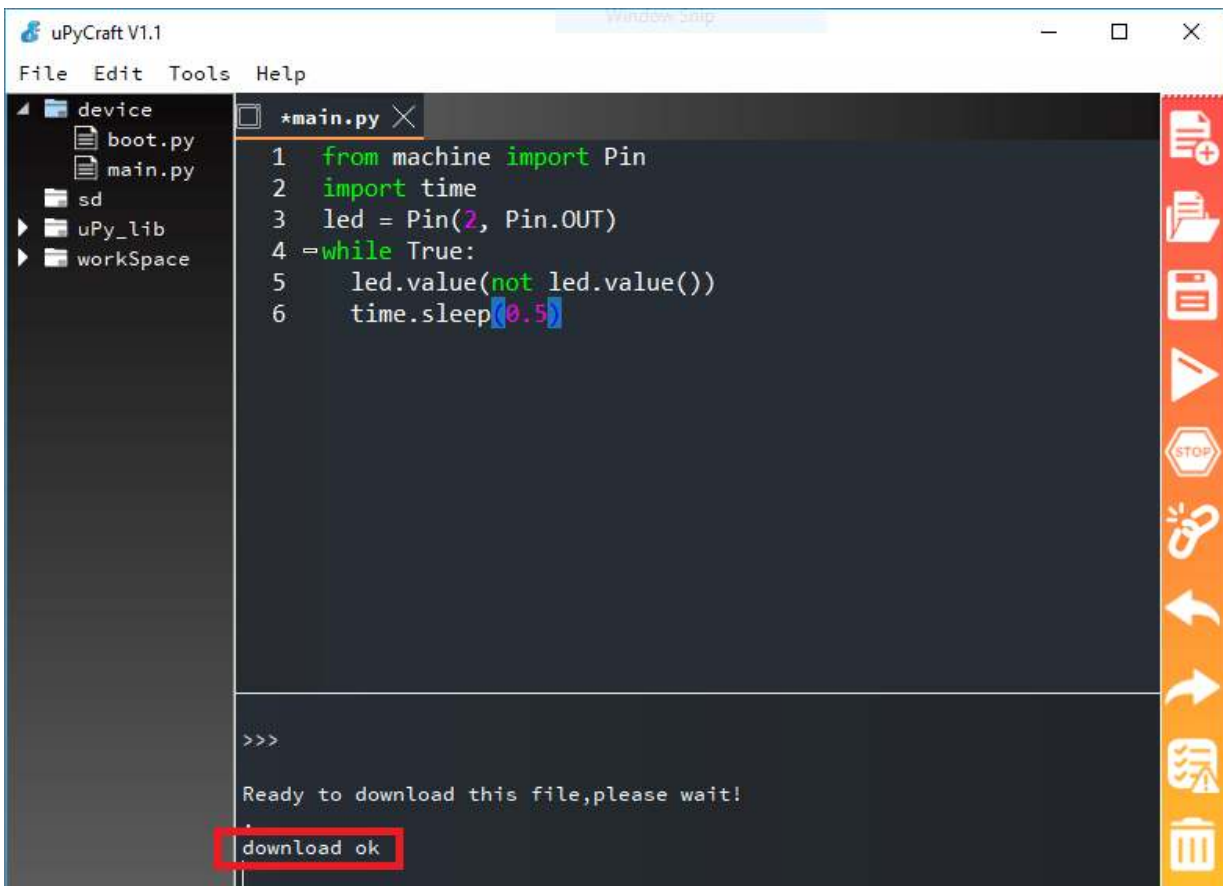
2. Press the “**Stop**” button to stop any script from running in your board:



3. Click the “**Download and Run** button” to upload the script to the ESP32 or ESP8266:



4. You should see a message saying “download ok” in the Shell window.



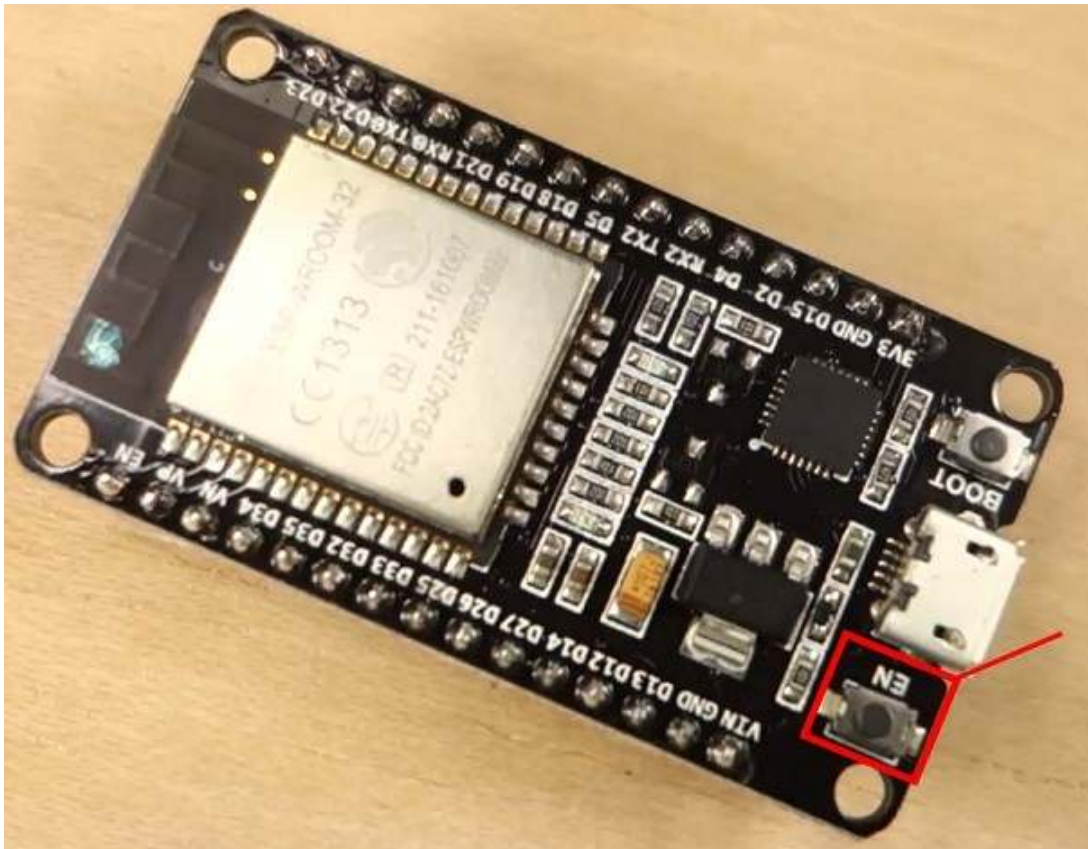
Testing the script

To run the script that was just uploaded to your board, you need to follow these steps:

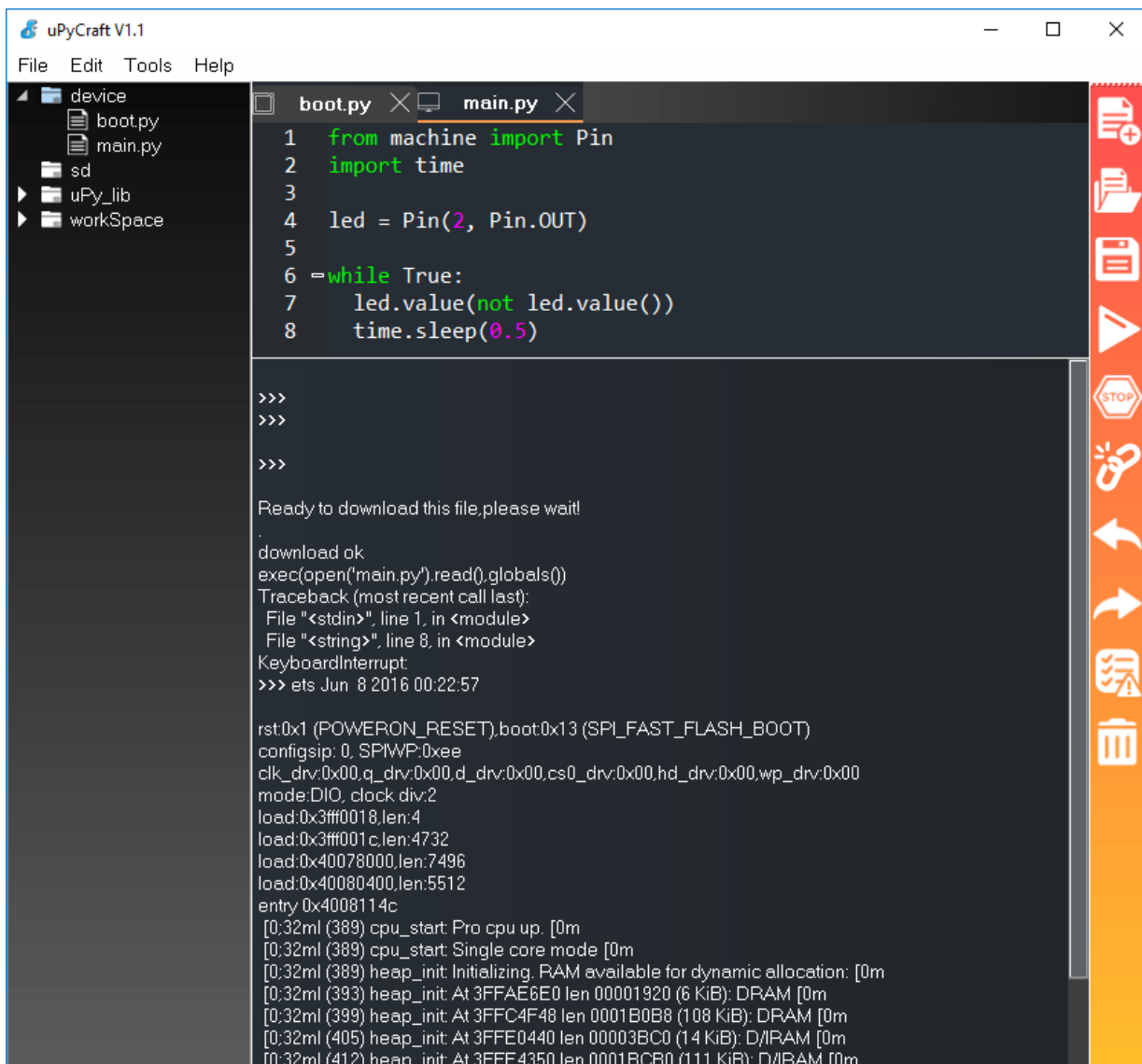
1. Press the “**Stop**” button



2. Press the on-board ESP32/ESP8266 **EN** (ENABLE) or **RST** (RESET) button to restart your board and run the script from the start:



If you're using an ESP32, your Terminal messages should look something as shown in the following figure after a EN/RST button press:



The screenshot shows the uPyCraft V1.1 IDE interface. On the left is a file explorer with a tree view containing 'device', 'boot.py', 'main.py', 'sd', 'uPy_lib', and 'workSpace'. The main editor window has two tabs: 'boot.py' and 'main.py', with 'main.py' selected. The code in 'main.py' is as follows:

```
1 from machine import Pin
2 import time
3
4 led = Pin(2, Pin.OUT)
5
6 while True:
7     led.value(not led.value())
8     time.sleep(0.5)
```

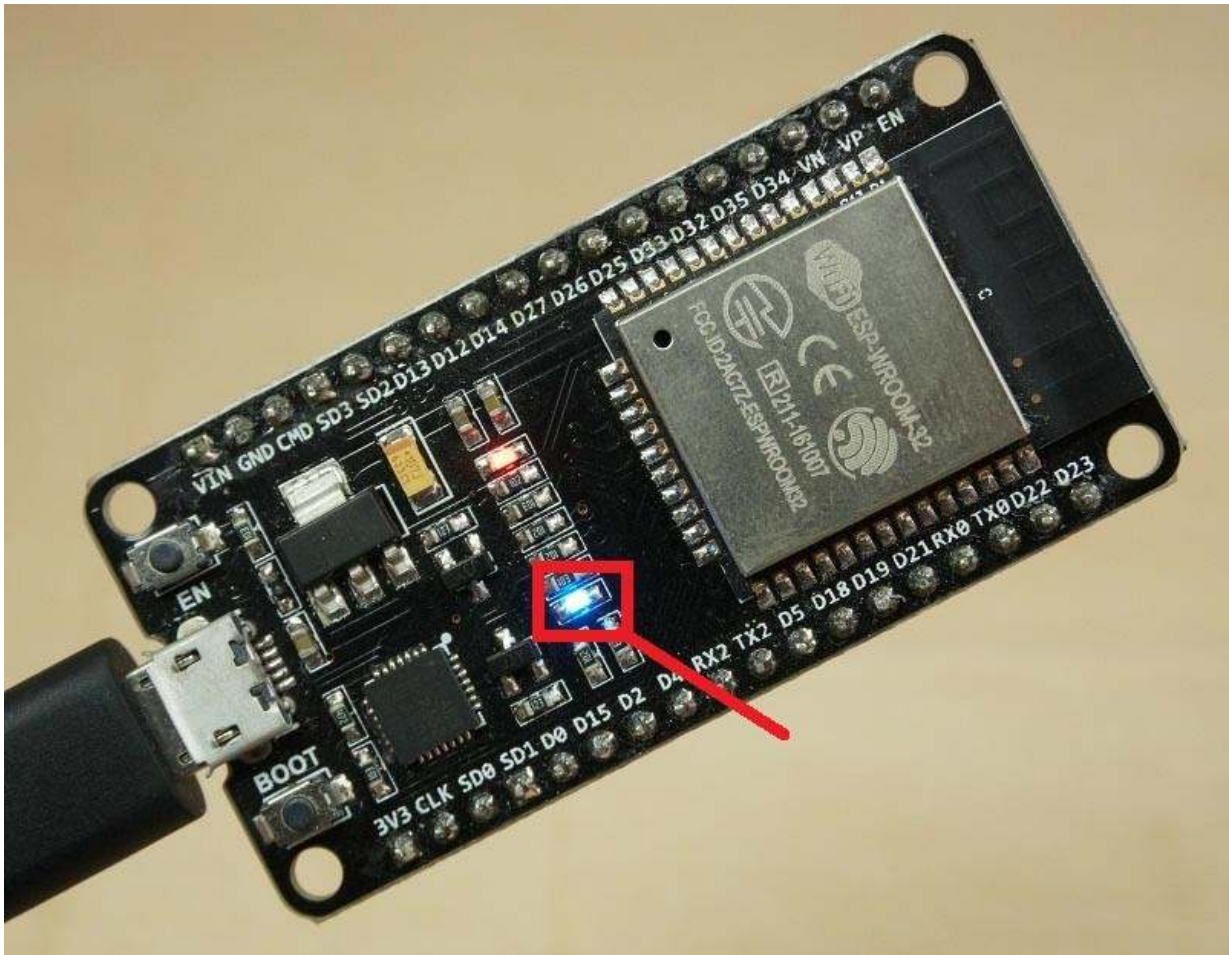
Below the code editor, the REPL output is displayed. It shows a successful download of the file, followed by a traceback error: 'KeyboardInterrupt'. The output then shows the system booting up, including hardware initialization and memory allocation details.

```
>>>
>>>
>>>

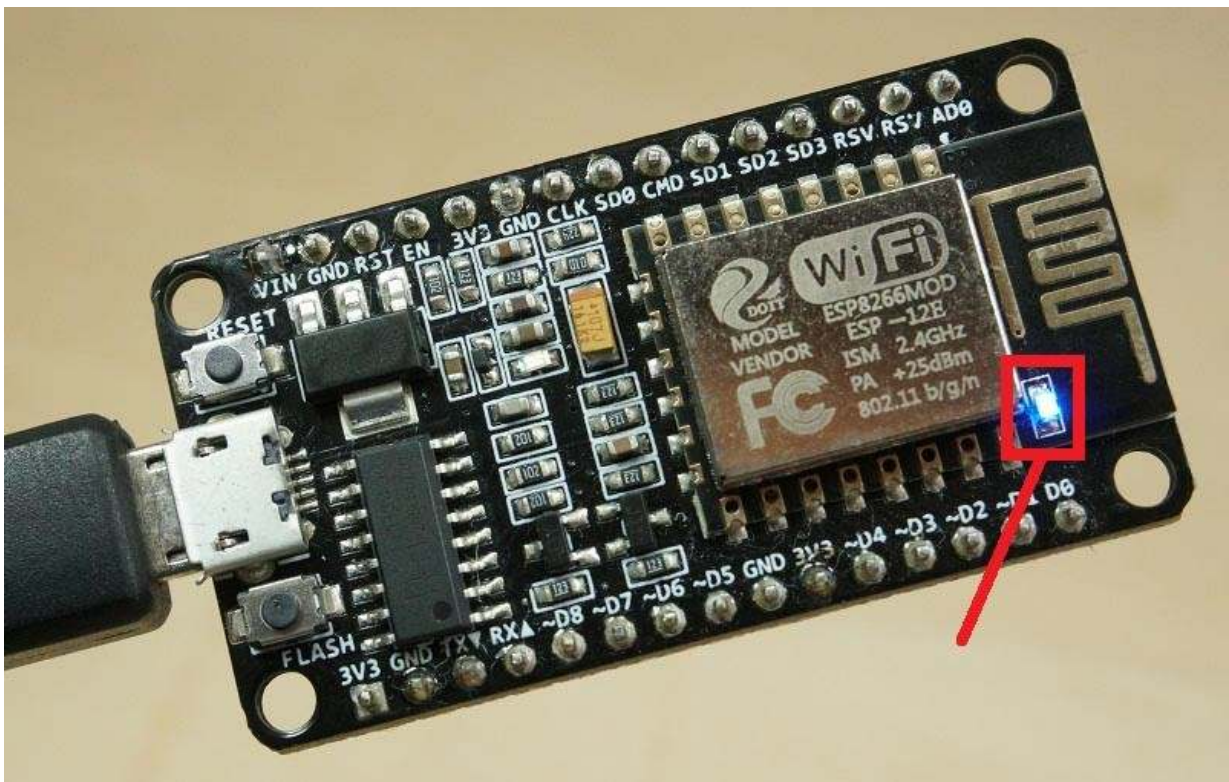
Ready to download this file, please wait!
.
download ok
exec(open('main.py').read(),globals())
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<string>", line 8, in <module>
KeyboardInterrupt
>>> ets Jun  8 2016 00:22:57

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2
load:0x3fff0018,len:4
load:0x3fff001c,len:4732
load:0x40078000,len:7496
load:0x40080400,len:5512
entry 0x4008114c
[0:32ml (389) cpu_start: Pro cpu up. [0m
[0:32ml (389) cpu_start: Single core mode [0m
[0:32ml (389) heap_init: Initializing. RAM available for dynamic allocation: [0m
[0:32ml (393) heap_init: At 3FFAE6E0 len 00001920 (6 KiB): DRAM [0m
[0:32ml (399) heap_init: At 3FFC4F48 len 0001B0B8 (108 KiB): DRAM [0m
[0:32ml (405) heap_init: At 3FFE0440 len 00003BC0 (14 KiB): D/IRAM [0m
[0:32ml (412) heap_init: At 3FFE4350 len 0001BCB0 (111 KiB): D/IRAM [0m
```

Your ESP32 or ESP8266 on-board LED should be blinking every 500 milliseconds. Here's where the ESP32's on-board LED is located:



Here's the ESP8266 on-board LED:



Troubleshooting Tips

We've discovered some common problems and error messages that occur with uPyCraft IDE. Usually restarting your ESP with the on-board EN/RST button fixes your problem. Or pressing the uPyCraft IDE **"Stop"** button and repeating your desired action. In case it

doesn't work for you, read these next common errors and discover how to solve them.

Error #1: You get the following message:

```
>>>
Select Serial Port could not open port 'COM4': FileNotFoundError(2, 'The system cannot find
```

Or an equivalent message:

```
>>>
could not open port 'COM4': PermissionError(13, 'A device attached to the system is not func
```

Unplug, and plug back your ESP board. Then, double-check that you've selected the right serial port in the **Tools > Serial** menu. Then, click the **“Connect/disconnect”** button to establish a serial communication. You should now be able to upload a new script or re-run new code.

This error might also mean that you have your serial port being used in another program (like a serial terminal or in the Arduino IDE). Double-check that you've closed all the programs that might be establishing a serial communication with your ESP board. Then, unplug and plug back your ESP board. Finally, restart the uPyCraft IDE – try to select the serial port in the **Tools > Serial** menu.

Error #2: Trouble uploading a new script.

```
>>>
already in download model,please wait.
```

Press the **“Stop”** button in uPyCraft IDE (1 or 2 times) to make sure any code that was running stops. After that, press the **“Download and run”** button to upload the new script to your ESP board.

Error #3: After uploading a new script, if you see the following message:

```
>>>
Ready to download this file,please wait!
...
download ok
os.listdir('.')
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
NameError: name 'os' isn't defined
```

Or this message:

```
>>>
Ready to download this file,please wait!
...
download ok
os.listdir('.')
OSError: [Errno 98]
```

It means the new file was uploaded to your board successfully. You can notice that it printed the **“download ok”** message. Press the ESP on-board **“EN/RST”** button to restart your board and re-run the new uploaded script from the beginning.

Error #4: Problem restarting your ESP board, running a new script or opening the serial port:

```
>>>
Brownout detector was triggered
```

The **“Brownout detector was triggered”** error message means that there's some sort of hardware problem. It's often related to one of the following issues:

- Poor quality USB cable;
- USB cable is too long;
- Board with some defect (bad solder joints);
- Bad computer USB port;

- Or not enough power provided by the computer USB port.

Solution: try a different shorter USB cable (with data wires), try a different computer USB port or use a USB hub with an external power supply.

Important: if you keep having constant problems or weird error messages, we recommend re-flashing your ESP board with the latest version of MicroPython firmware: [Flash/Upload MicroPython Firmware to ESP32 and ESP8266](#).

Error #5: When I try to open a serial communication with the ESP32/ESP8266 in uPyCraft IDE, sometimes it prompts the “Burn Firmware” window asking to re-flash the MicroPython firmware.

Basically, we think this is what’s happening: when you’re running a script in your board, sometimes it’s busy running that script and performing the tasks. So, you need to try opening the COM port multiple times or restart the ESP to catch it available to establish the serial communication with uPyCraft IDE.

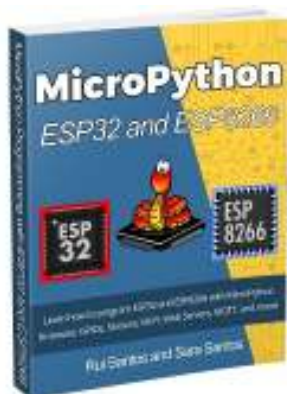
If you’re running a script that uses Wi-Fi, deep sleep, or it’s doing multiple tasks, I recommend trying 3 or 4 times to establish the communication. If you can’t, I recommend re-flash the ESP with MicroPython firmware.

Wrapping Up

We hope you’ve enjoyed learning how to program the ESP32 and ESP8266 boards using MicroPython firmware. More tutorials with MicroPython will be posted soon, so make sure you [subscribe to the RNT blog](#) and download our free electronics eBooks.

If you prefer to program the ESP32/ESP8266 with Arduino IDE, you might want to read one of these tutorials instead:

- [Getting Started with ESP32 – Arduino IDE](#)
- [\[Course\] Learn ESP32 with Arduino IDE](#)
- [Getting Started with ESP8266 – Arduino IDE](#)
- [\[Course\] Home Automation using ESP8266](#)



Recommended Resources