

Installing MicroPython on ESP32

Aldo Núñez Tovar

Introduction

We present a procedure to install MicroPython on the ESP32 microcontroller. It was carried out on the GNU/Linux platform and using open source tools.

We also install Thonny IDE, a lightweight program and multiplatform, to edit and execute MicroPython programs.

This procedure was done using Debian GNU/Linux 12 (bookworm). And the microcontroller was ESP32-WROOM-32U

This tutorial can be downloaded from my [github](#) account.

Requirements

We need a computer with internet access and the microcontroller connected via the USB port, Figure 1.



Figure 1. Connection between computer and ESP₃₂

To install MicroPython we need a microcontroller with at least 16 KB of RAM memory and 256 KB of ROM memory.

The ESP32 WROOM has 520 KB of RAM memory and 448 KB of ROM memory

Components and accessories

1. Microcontroller ESP32
2. usb to micro usb cable

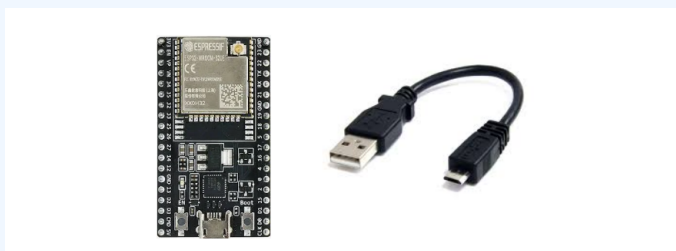


Figure 2. Components and accessories

Install MicroPython and tools

1. Download MicroPython
2. Download and install esptool program. This tool is to install MicroPython on the microcontroller.
3. Download and install the mpremote program. This tool is to upload and execute, MicroPython programs in the microcontroller.

We assume that the device used to communicate with the microcontroller is the file `/dev/ttyUSB0`. In other cases, the device could be different. List the `/dev` subdirectory to find out which device your computer is using.

```
$ ls -ltr /dev
```

1.- Download MicroPython.

First, we create the `esp32` subdirectory. It will be our workspace subdirectory.

```
$ mkdir esp32  
$ cd esp32
```

Next, go to: <https://micropython.org/download/> and select your microcontroller model.

In our case we choose ESP32 microcontroller: https://micropython.org/download/ESP32_GENERIC/

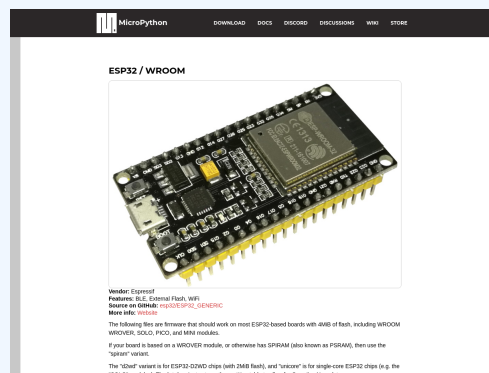


Figure 3. downloads

Then, scroll down until the Firmware section and select the most recent release: `v1.24.1 (2024-11-29) .bin`. Download this file to the `esp32` subdirectory

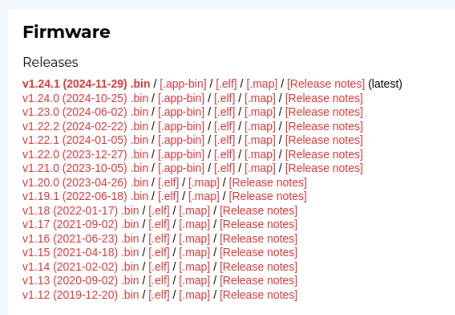


Figure 4. Firmware

The downloaded file is: `ESP32_GENERIC-20241129-v1.24.1.app-bin`

2. Download esptool

```
~/esp32/$ git clone https://github.com/espressif/esptool.git
```

After executing the previous command, the `esptool` subdirectory will have been created.

First, erase the entire flash memory.

Change to the `esptool` subdirectory

```
~/esp32/$ cd esptool
~/esp32/esptool/$ python -m esptool --port /dev/ttyUSB0 erase_flash
```

Then, install MicroPython on the microcontroller

```
~/esp32/$ cd esptool
~/esp32/esptool/$ python -m esptool --chip esp32 --port /dev/ttyUSB0 --baud
460800 write_flash -z 0x1000 ../ESP32_GENERIC-20241129-v1.24.1.bin
```

At this point and if there wasn't any problem we have installed MicroPython successfully on the ESP32

Once installed MicroPython, press the EN button on the ESP32.

3. Install mpremote

We can use any serial communication program to connect to the microcontroller. In our case we will install the `mpremote` program to interact with the ESP32

```
$ pip install mpremote
```

Now, we can connect to the microcontroller

```
$ mpremote connect /dev/ttyUSB0
Connected to MicroPython at /dev/ttyUSB0
Use Ctrl-] or Ctrl-x to exit this shell
```

Press Enter.

When the interpreter starts, it displays a message and the MicroPython prompt appears where programs can be written in a loop called "read-evaluation-print loop" or REPL.

```
>>>
```

Type `help()` and MicroPython will print messages to interact with ports and to configure WiFi

```
>>> help()
Welcome to MicroPython on the ESP32!
For online docs please visit http://docs.micropython.org/
For access to the hardware use the 'machine' module:
import machine
pin12 = machine.Pin(12, machine.Pin.OUT)
pin12.value(1)
.....
```

We can interact with the `esp32` through the MicroPython interpreter.

For example, let's turn on and turn off a LED connected to Port 2 through a 220 Ohms Resistor, [Figure 5](#)

First, import the `Port` class from the `machine` module

```
>>> from machine import Pin
```

Then, declare a `led` object using the Port 2 as output

```
>>> led = Pin(2, Pin.OUT)
```

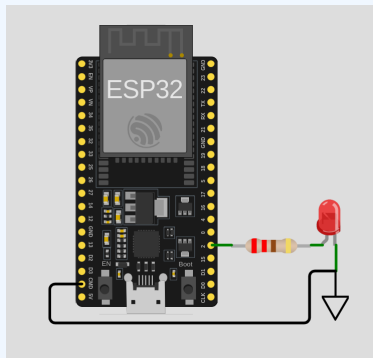


Figure 5. LED connected to Port 2

Turn on the LED

```
>>> led.on()
```

Turn off the LED

```
>>> led.off()
```

As you can see it is easy to interact with the microcontroller using MicroPython.

Create your first program in MicroPython

Let's create the programs subdirectory in esp directory

```
~/esp32/$ mkdir programs
```

```
~/esp32/$ cd programs
```

Now, inside programs we are going to edit blink.py program. Open your favorite text editor.

```
1 from machine import Pin
2 from time import sleep
3
4 '''
5     Blink a Led connected to port 2
6     of the esp32 microcontroller
7 '''
8
9 PORT = 2
10 SECONDS = 0.5
11
12 led = Pin(PORT, Pin.OUT)
13
14 while(True):
15     led.on()
16     sleep(SECONDS)
17     led.off()
18     sleep(SECONDS)
```

Execute blink.py

```
~/esp32/programs/$ mpremote run blink.py
```

Now, we can observe the blinking of the LED every 0.5 seconds. To stop the execution of the blink.py program press Ctrl+c.

To list the files in the flash memory of the microcontroller

```
~/esp32/programs$ mpremote ls
```

```
ls :
    139 boot.py
```

To upload the `blink.py` program to the microcontroller

```
~/esp32/programs$ mpremote fs cp blink.py :blink.py
```

Now, execute the list command again

```
~/esp32/programs$ mpremote ls
ls :
    318 blink.py
    139 boot.py
```

To print the options of `mpremote` program type:

```
~/esp32/programs$ mpremote --help
```

Installing Thonny IDE

Thonny is a free Python IDE alternative. It is suitable for those who are just starting to learn Python.

1. Download

Go to <https://thonny.org/> and download the version of the program corresponding to your operating system: Windows, Mac or Linux

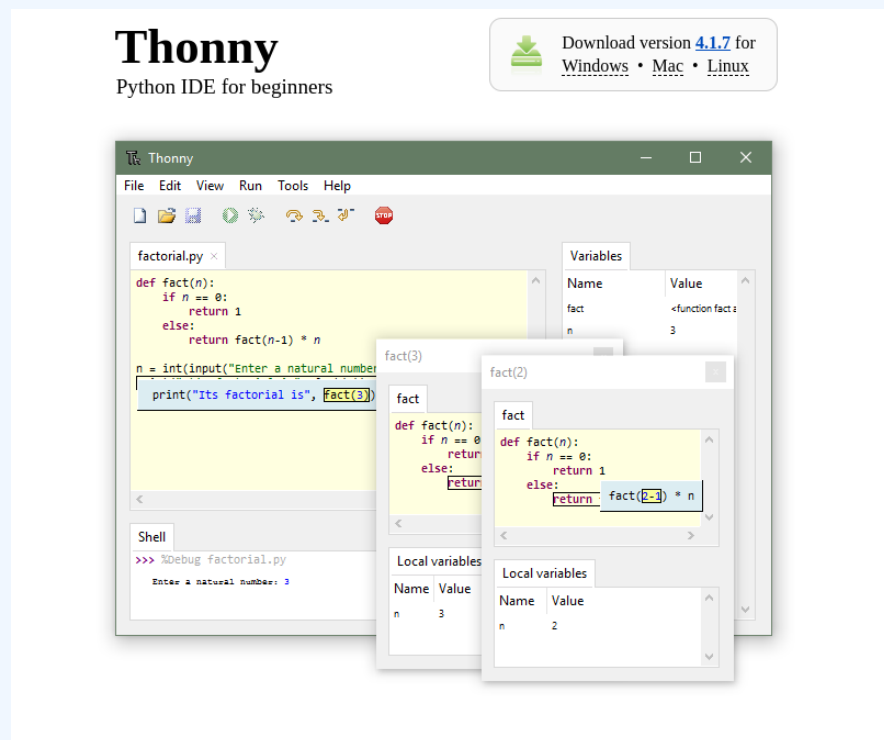


Figure 6. Download Thonny

2. Install:

In our case, we have downloaded `thonny-4.1.7-x86_64.tar.gz` file, corresponding to Linux OS, to `esp32` subdirectory

Unpack the downloaded file

```
1 ~/esp32$ tar zxvf thonny-4.1.7-x86_64.tar.gz
```

A subdirectory `thonny` will be created. Change to this subdirectory and install

```
1 ~/esp32$ cd thonny
2 ~/esp32/thonny$ ./install.py
```

Now, we are ready to open Thonny IDE.

At the top we have the editor and at the bottom the Python interpreter, Figure 7

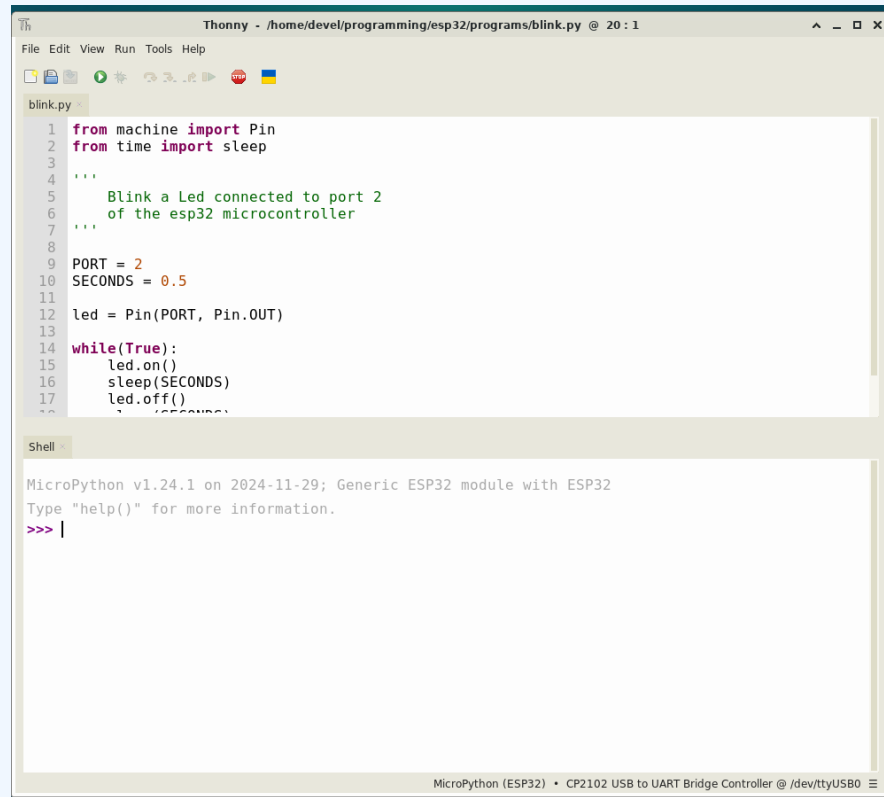


Figure 7. Thonny IDE

With this IDE we can easily interact with the microcontroller. We can edit, execute, upload, remove MicroPython programs.