**INDEX:**

1. [Building and flashing Micropython 2](#_TOC_250005)
   1. [Overview: 2](#_TOC_250004)
   2. [You will need: 2](#_TOC_250003)
   3. [Building MicroPython on Linux 2](#_TOC_250002)
   4. [Flashing Micropython to an STM32 Board 8](#_TOC_250001)
   5. [Testing Micropython Flash 10](#_TOC_250000)

# Building and flashing Micropython

## Overview:

* + - Use a Linux computer (or a virtual machine on a Windows host) to compile MicroPython and generate a hex file for flashing

o If linux system is not available you can create a separate linux based virtual machine in AWS or azure and use its command shell to generate .hex files

* + - Use the STM32CubeProgrammer to flash the hex file to an STM32 board on a Linux/Mac/Windows computer

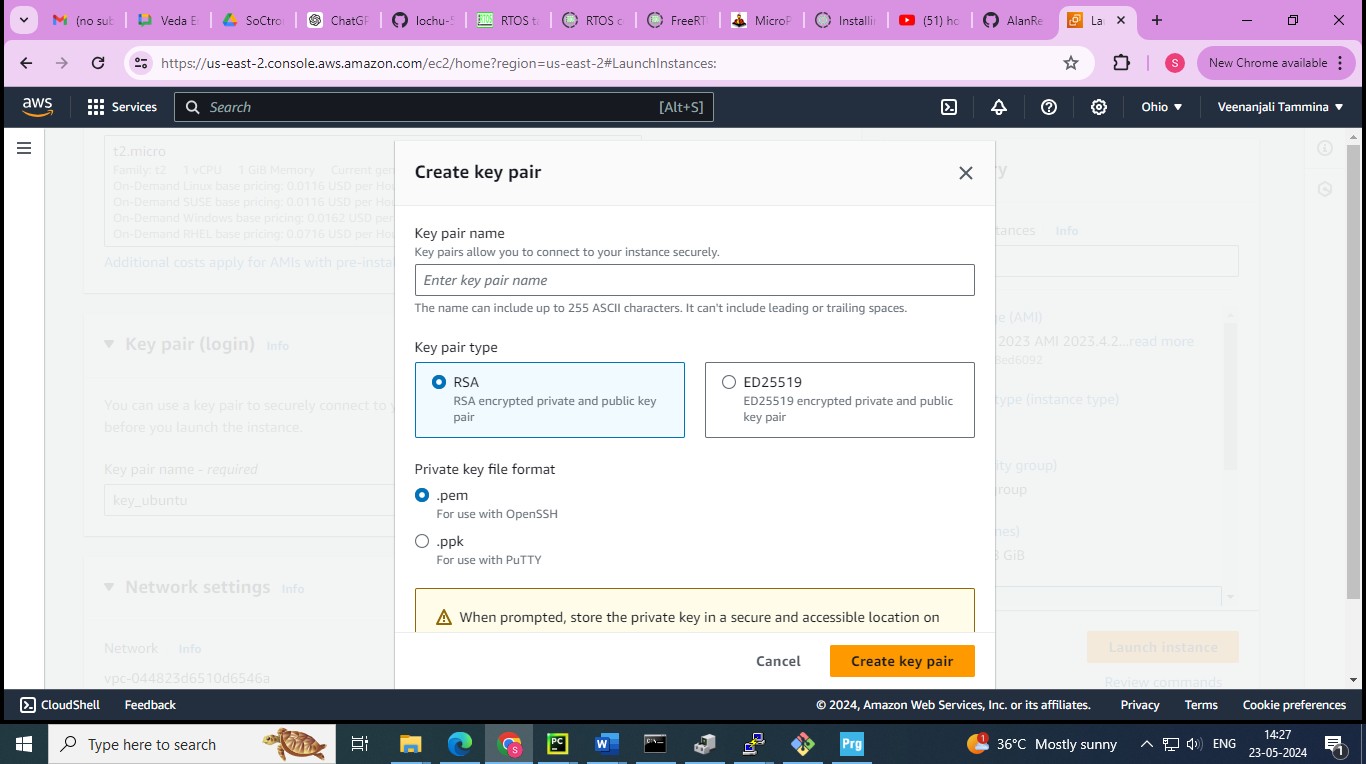
## You will need:

* + - Computer(s) with Linux installed, e.g. Ubuntu
    - STM32 Nucleo or STM32 Discovery microcontroller board
    - [STM32CubeProgrammer](https://www.st.com/en/development-tools/stm32cubeprog.html)
    - [Serial communication utility software such as PuTTY](https://putty.org/)
    - MicroUSB cable

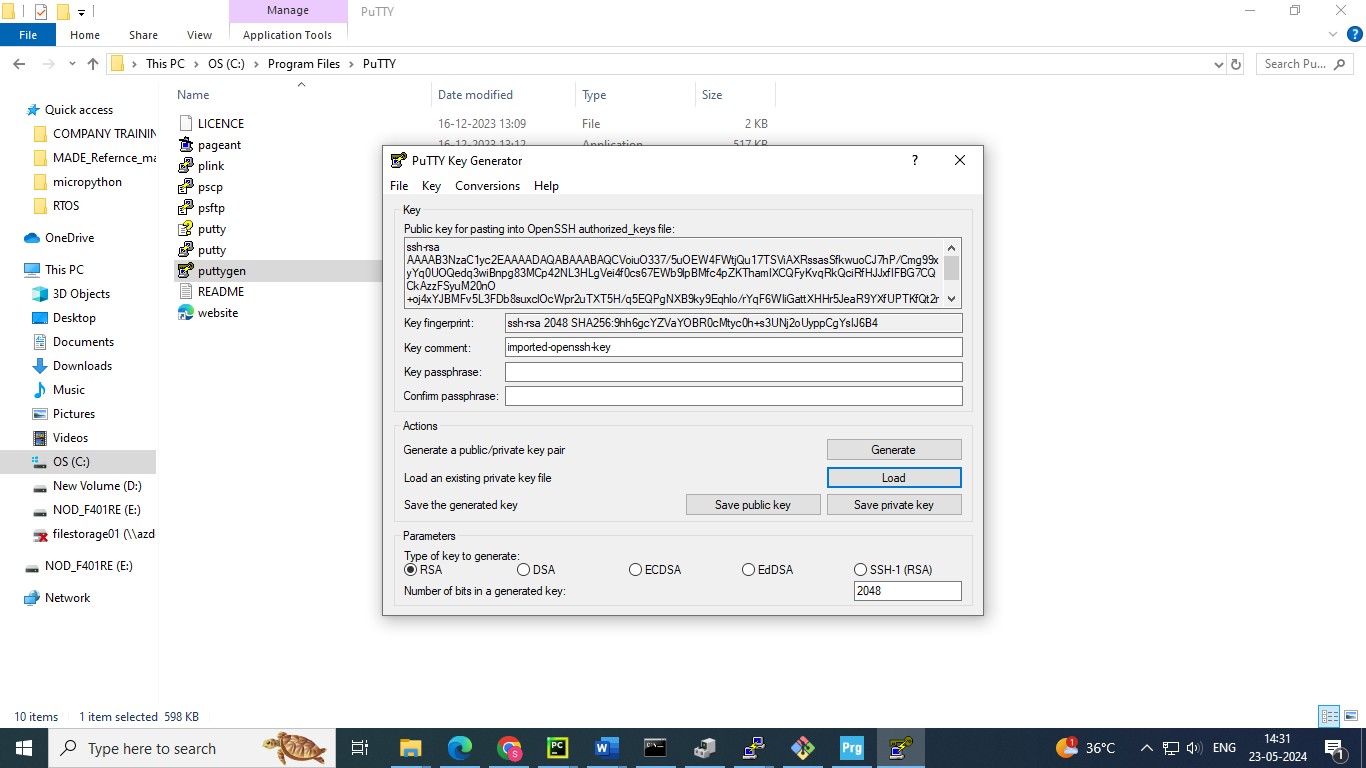
## Building MicroPython on Linux

1. Create a VM in aws or ubuntu
2. While creating VM , we need to create a key-pair of type rsa named key\_ubuntu with extension .ppk and download it to use with puTTy
   * This key is used for secure communication and authentication. Instead of using a password to log into a remote server, you can use an RSA key pair for more

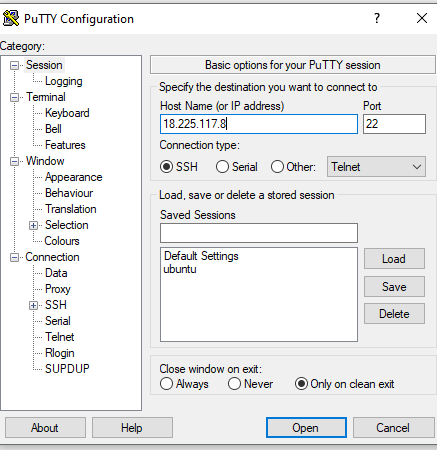
secure and convenient access.



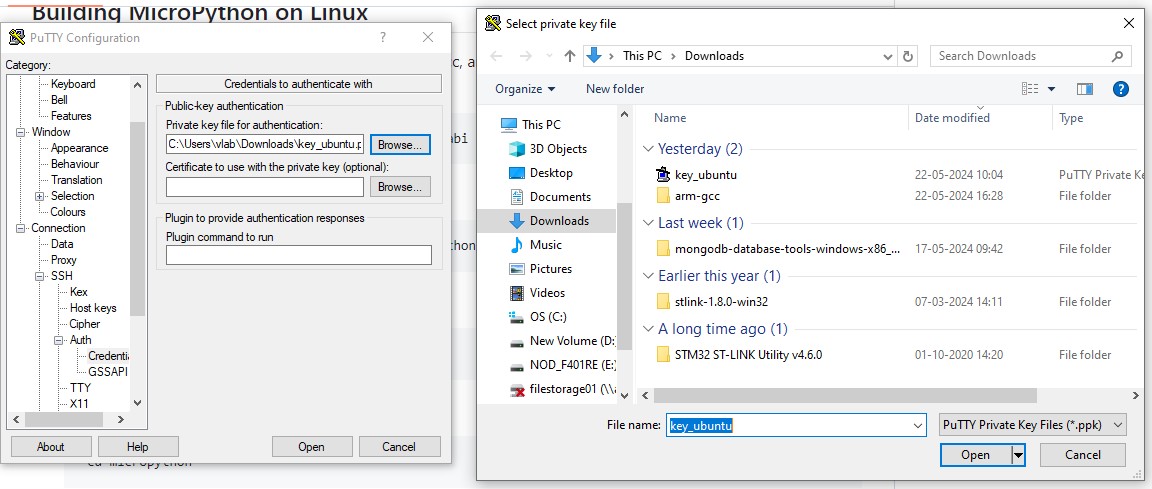
* + If you downloaded .pem key file format by mistake , then convert it into .ppk extension using puttygen application downloaded in your local machine.
  + Load the .pem key you have the save both public and private keys with .ppk extension



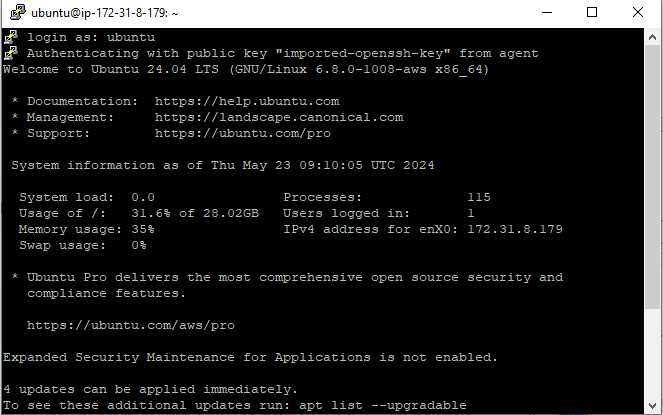
* + Open puTTY □ select connection type SSH □enter your VM Ip address □ check port number 22 □ go to SSH/Auth/credentials



* + Browse the .ppk key\_ubuntu from your local machine □ click open



* + Enter the ubuntu as username if you created ubuntu VM
  + It will navigate to your linux VM command shell



1. In Linux or VirtualBox Debian Linux, install git, make, gcc, and gcc-arm-none-eabi by going to the command terminal and executing the following command

### sudo apt-get install git make gcc gcc-arm-none-eabi

1. Clone the MicroPython source repository by calling

### git clone <https://github.com/micropython/micropython/>

1. Go to the MicroPython directory by calling

### cd micropython

1. Go to the port directory and update STM32 submodules by calling

### cd ports/stm32

1. build the STM32 firmware for a specific board by calling

### make BOARD={your-board-model-here}

In my case, I use

### make BOARD=NUCLEO\_F401RE

1. Upon success, the bottom of the message appears as: **LINK build-NUCLEO\_F401RE/firmware.elf text data bss dec hex filename 290172 48 21172 311392 4c060**

### build- NUCLEO\_F401RE/firmware.elf

### GEN build-NUCLEO\_F401RE/firmware0.bin GEN build-NUCLEO\_F401RE/firmware1.bin GEN build-NUCLEO\_F401RE/firmware.dfu GEN build-NUCLEO\_F401RE/firmware.hex

1. Now we need get those firmware files generated into our local machine , So using upload those files

**ubuntu@ip-172-31-8-179**:~$ cp

micropython/ports/stm32/build-NUCLEO\_F401RE/firm\*.\* bin\_files

**ubuntu@ip-172-31-8-179**:~$ cd bin\_files/

**ubuntu@ip-172-31-8-179**:~/bin\_files$ ls

firmware.dfu firmware.elf firmware.hex firmware.map firmware0.bin firmware1.bin

**ubuntu@ip-172-31-8-179**:~/bin\_files$ git remote add origin git@github.com:lochu-55/micropython.git

**ubuntu@ip-172-31-8-179**:~/bin\_files$ git add .

**ubuntu@ip-172-31-8-179**:~/bin\_files$ git commit -m "bin files" [master (root-commit) a3adc69] bin files

6 files changed; 41254 insertions(+) create mode 100644 firmware.dfu create mode 100755 firmware.elf

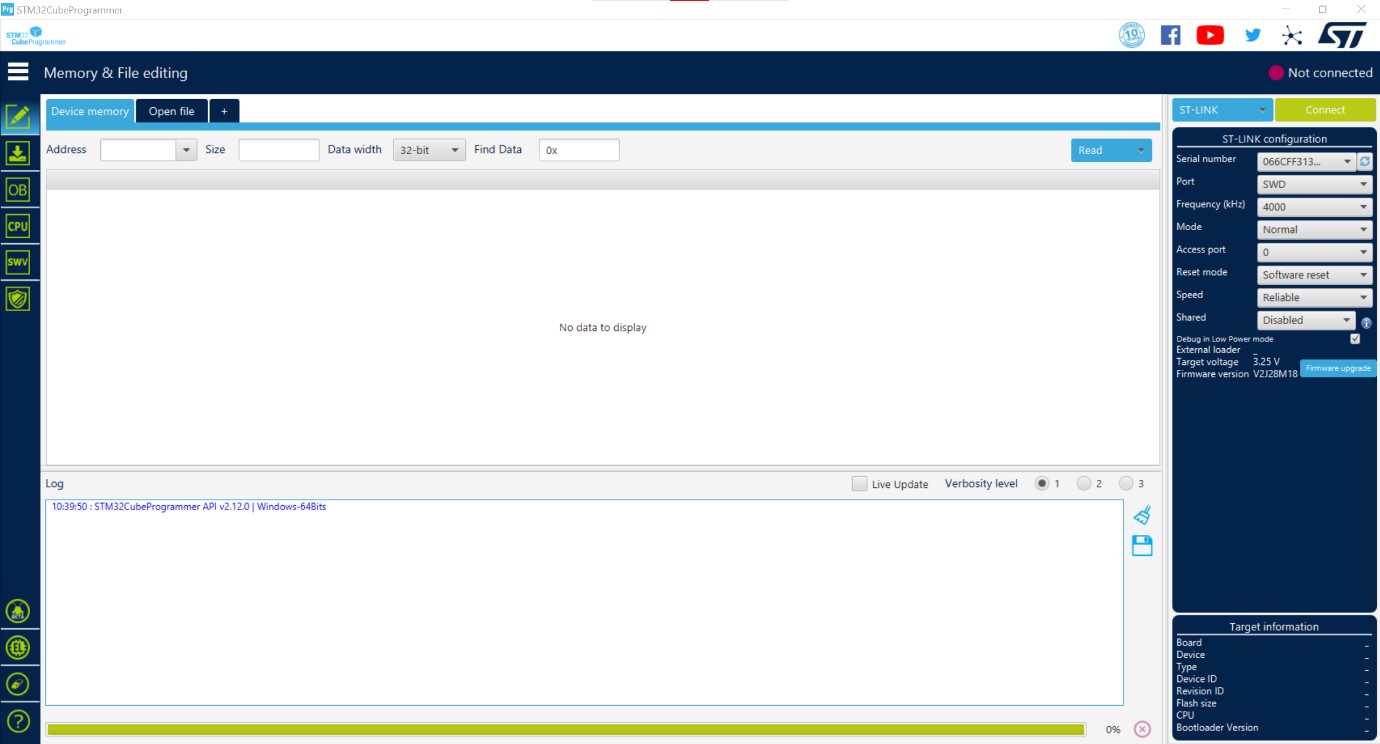
create mode 100644 firmware.hex

create mode 100644 firmware.map create mode 100755 firmware0.bin create mode 100755 firmware1.bin

now clone back the same git repo where you stored those into your local machine using git-bash

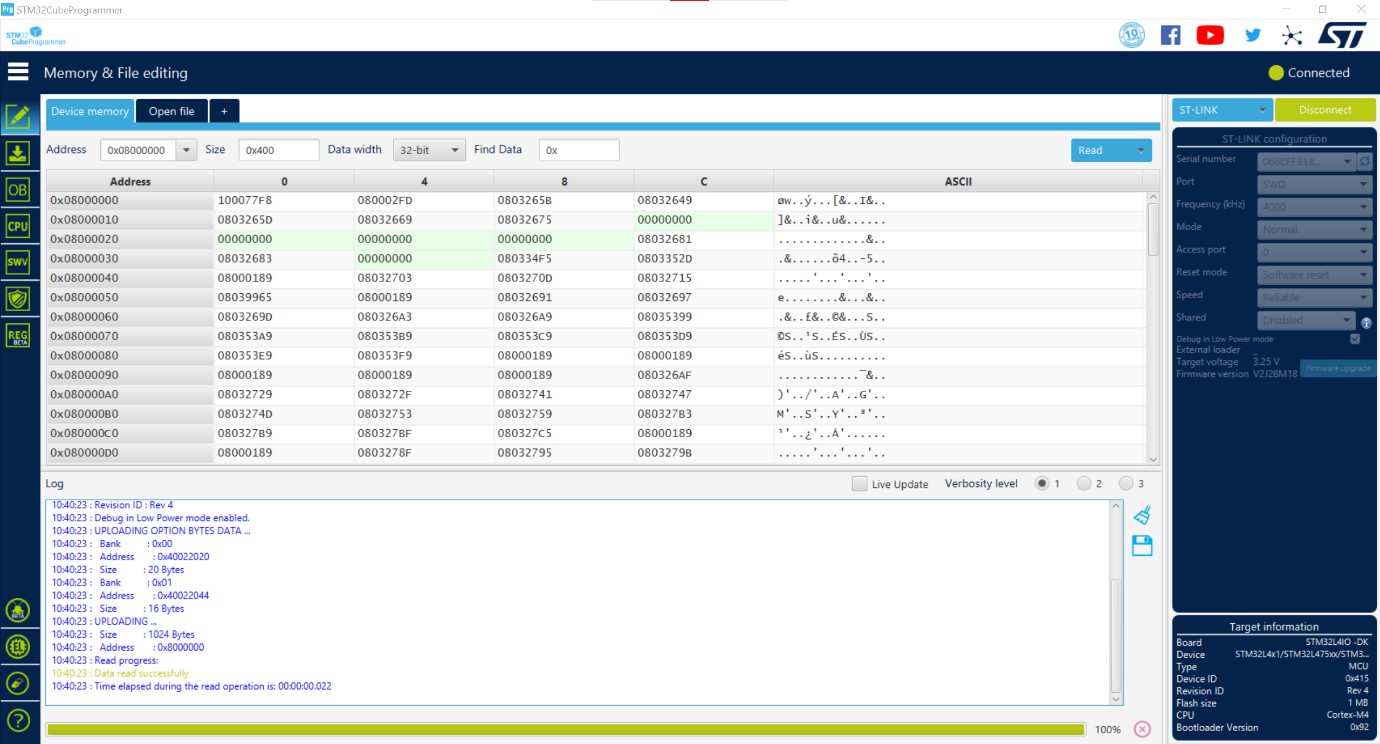
## Flashing Micropython to an STM32 Board

1. Connect the board to the computer through a microUSB cable.



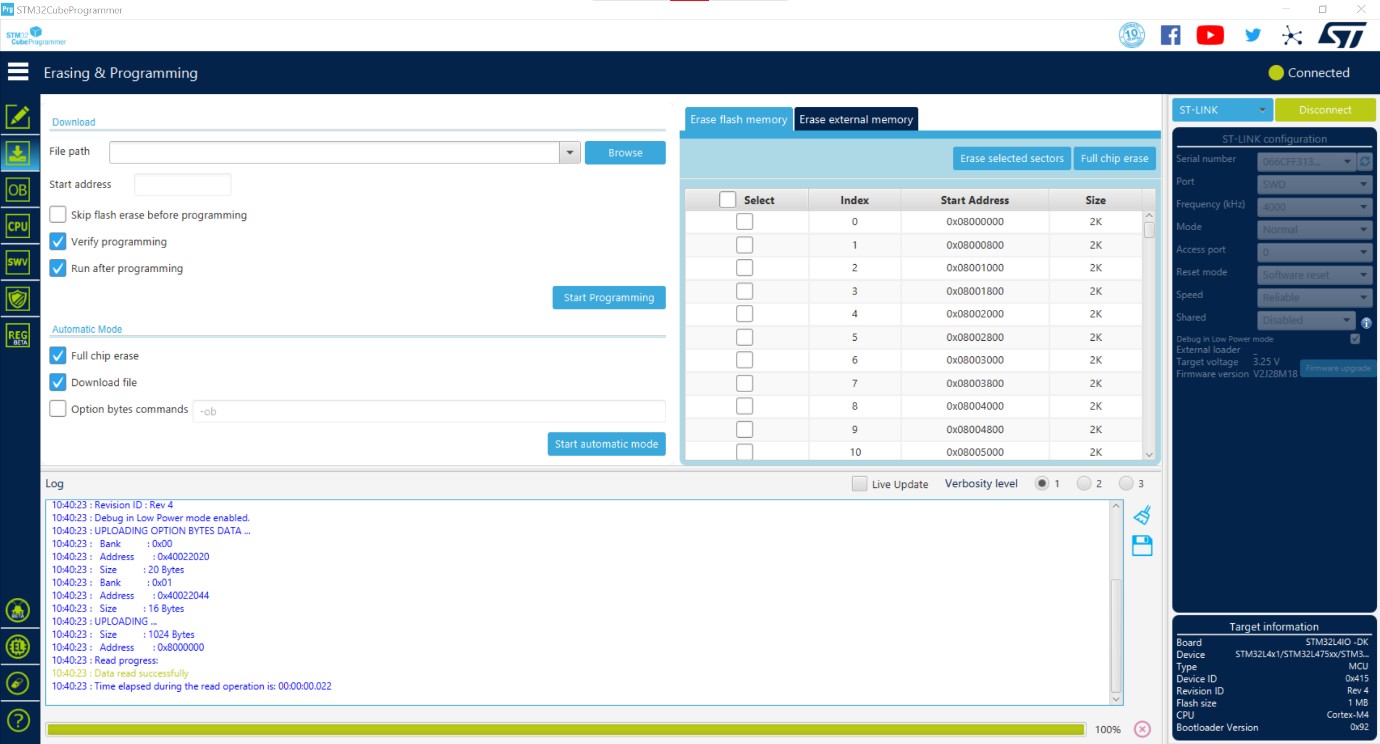
)

1. In the STM32CubeProgrammer, press Connect button near the top right corner.



)

1. Under Erase and programming, browse for firmware.hex file.



1. Flash the file by pressing the Start Programming button.

As a result, here are the logs 10:44:40 : File download complete

10:44:40 : Time elapsed during download operation: 00:00:09.719 10:44:40 : Verifying ...

10:44:40 : Read progress:

10:44:42 : Download verified successfully

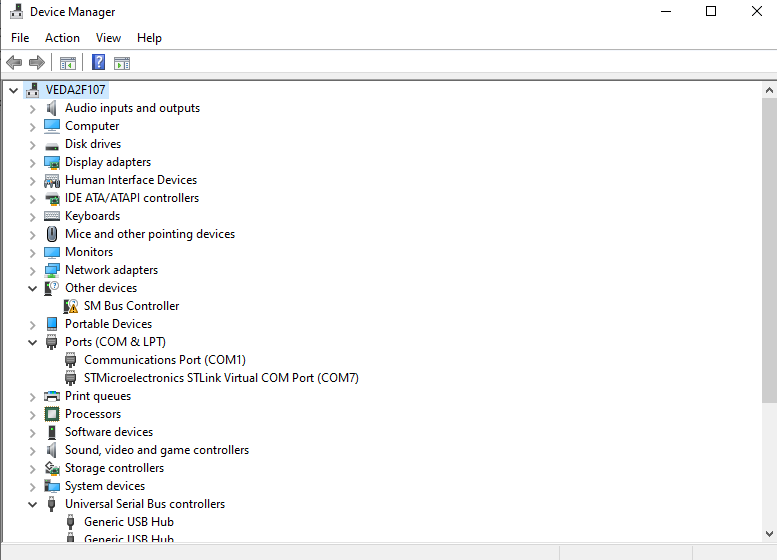
10:44:42 : RUNNING Program ...

10:44:42 : Address: : 0x08000000

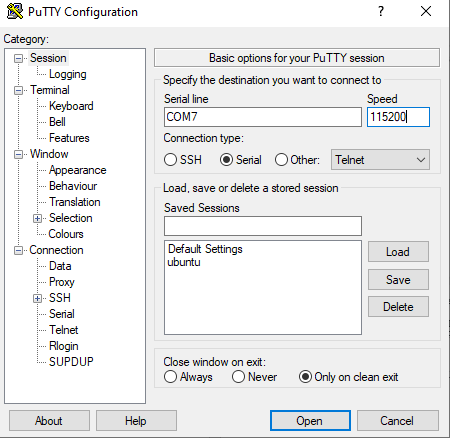
10:44:42 : Application is running, Please Hold on...

## Testing Micropython Flash

1. On a Windows Computer, open Windows Device Manager.
2. Expand “Ports (COM & LPT)". Look for the STLink Virtual COM Port entry and record the COM port. Once found, close device manager. In my case, my COM port to the STM32 board is COM3.

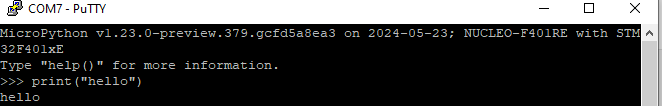


1. Open a serial communication utility software such as PuTTY.
2. Select “Serial” under Connection Type. Enter the COM7 port you recorded into the Serial Line field. Enter 115200 for speed (baudrate). Click Open.



1. Click inside the empty serial terminal window. Press the reset button on the board. A REPL display appears on the screen. In the REPL display, you can call the print function for Python such as

print(“Hello”)



1. If the message you tried to print successfully shows up in the terminal,

congratulations, you have successfully flashed micropython onto your microcontroller board.