

INDEX:

1. Building and flashing Micropython	2
1.1 Overview:	2
1.2 You will need:	2
1.3 Building MicroPython on Linux	2
1.4 Flashing Micropython to an STM32 Board	8
1.5 Testing Micropython Flash	10

1. Building and flashing Micropython

1.1 Overview:

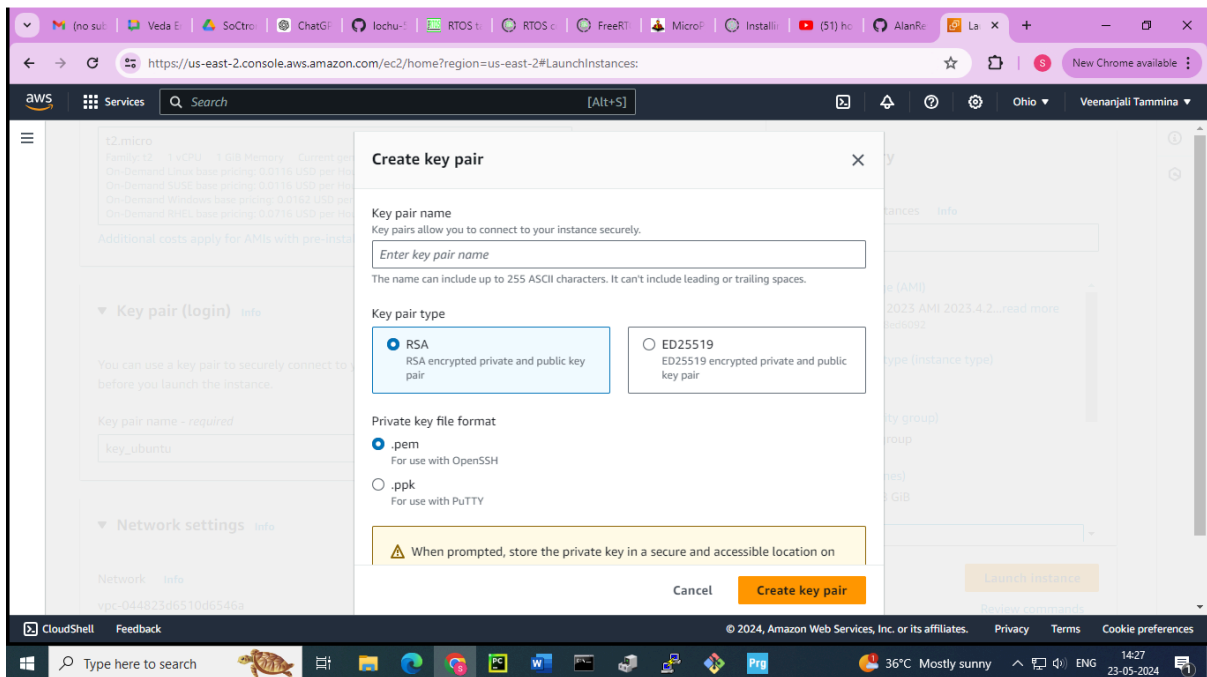
- Use a Linux computer (or a virtual machine on a Windows host) to compile MicroPython and generate a hex file for flashing
 - If linux system is not available you can create a separate linux based virtual machine in AWS or azure and use its command shell to generate .hex files
- Use the STM32CubeProgrammer to flash the hex file to an STM32 board on a Linux/Mac/Windows computer

1.2 You will need:

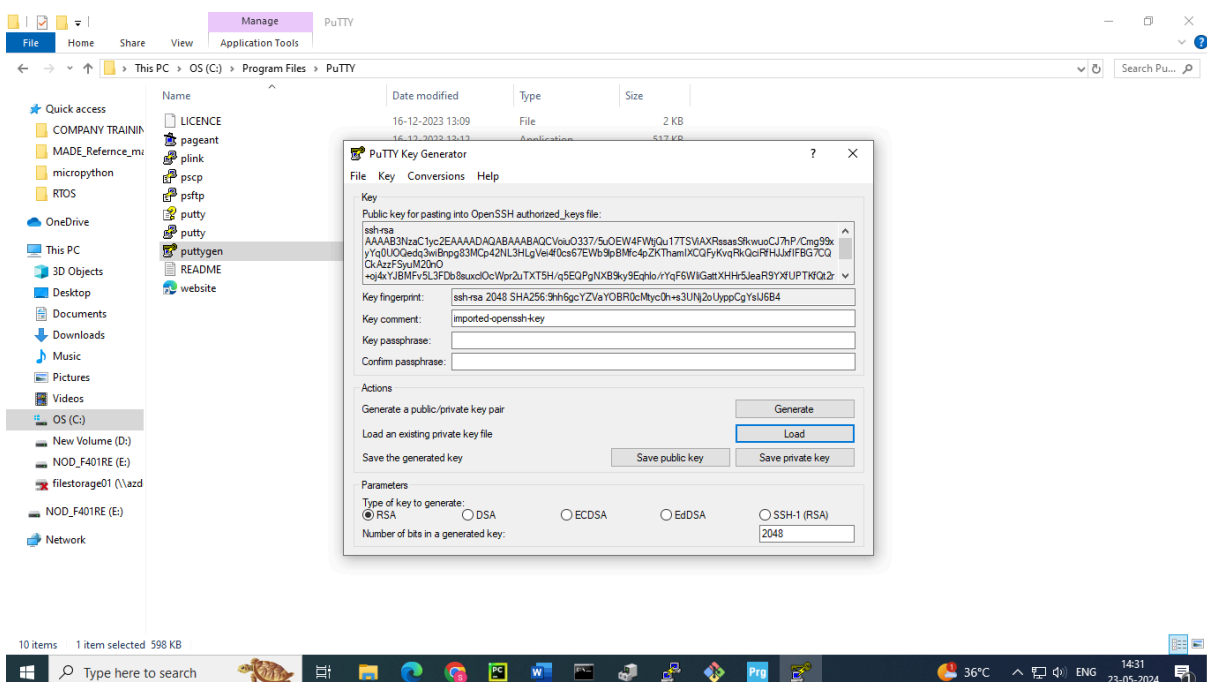
- Computer(s) with Linux installed, e.g. Ubuntu
- STM32 Nucleo or STM32 Discovery microcontroller board
- STM32CubeProgrammer
- Serial communication utility software such as PuTTY
- MicroUSB cable

1.3 Building MicroPython on Linux

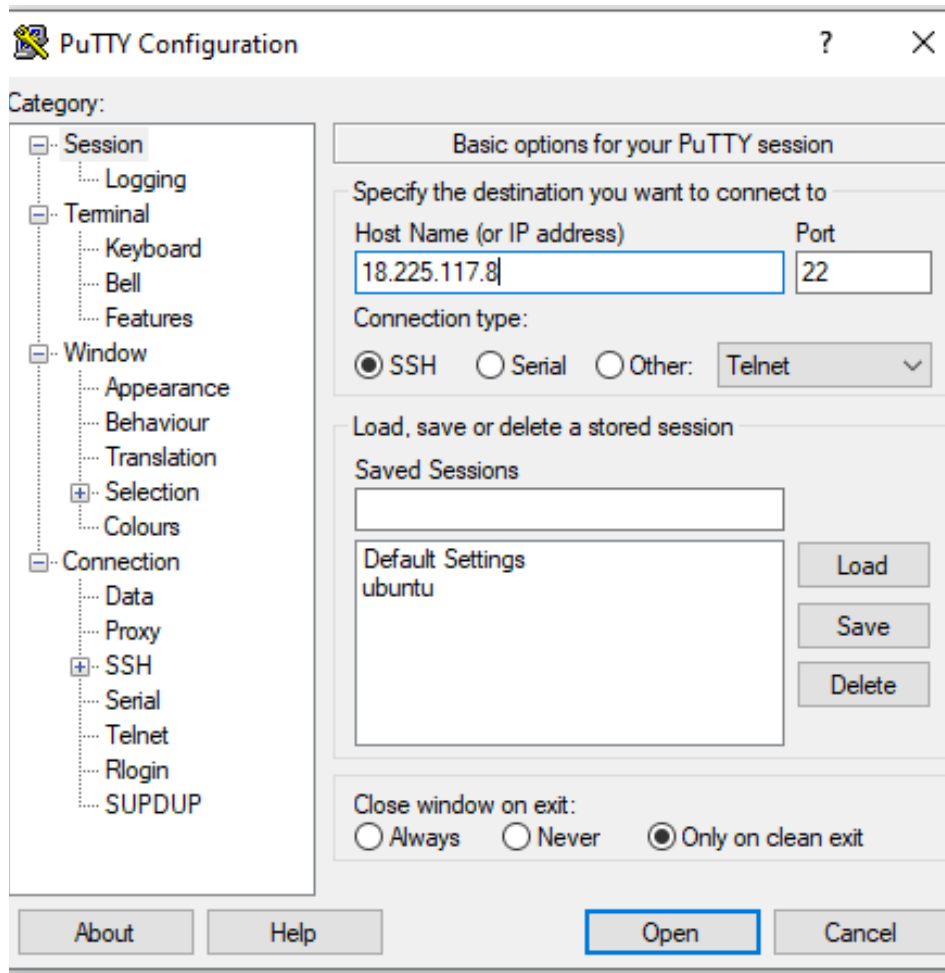
1. Create a VM in aws or ubuntu
2. While creating VM , we need to create a key-pair of type rsa named key_ubuntu with extension .ppk and download it to use with puTTY
 - This key is used for secure communication and authentication. Instead of using a password to log into a remote server, you can use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



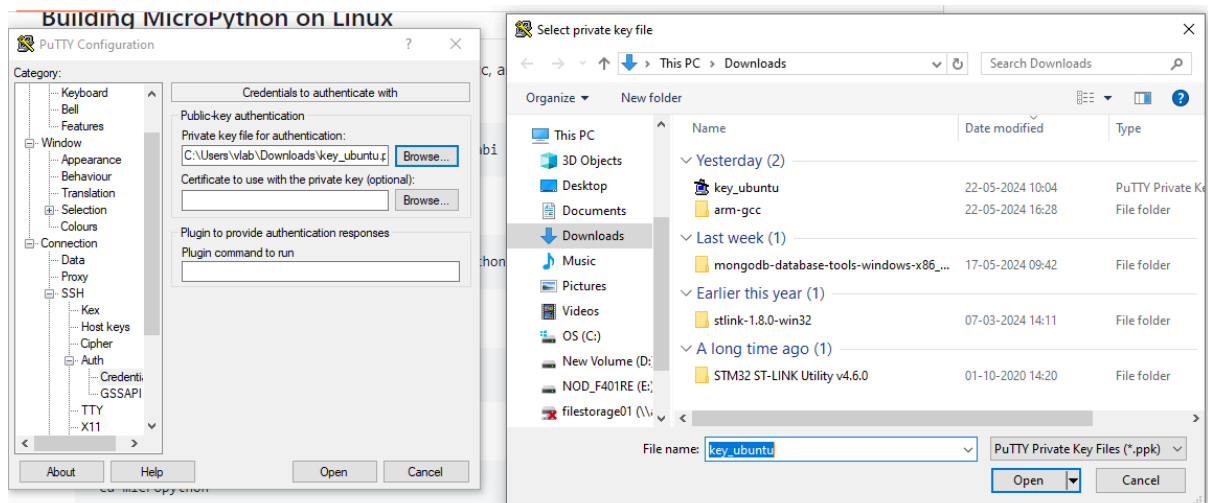
- o If you downloaded .pem key file format by mistake , then convert it into .ppk extension using puttygen application downloaded in your local machine.
- o Load the .pem key you have the save both public and private keys with .ppk extension



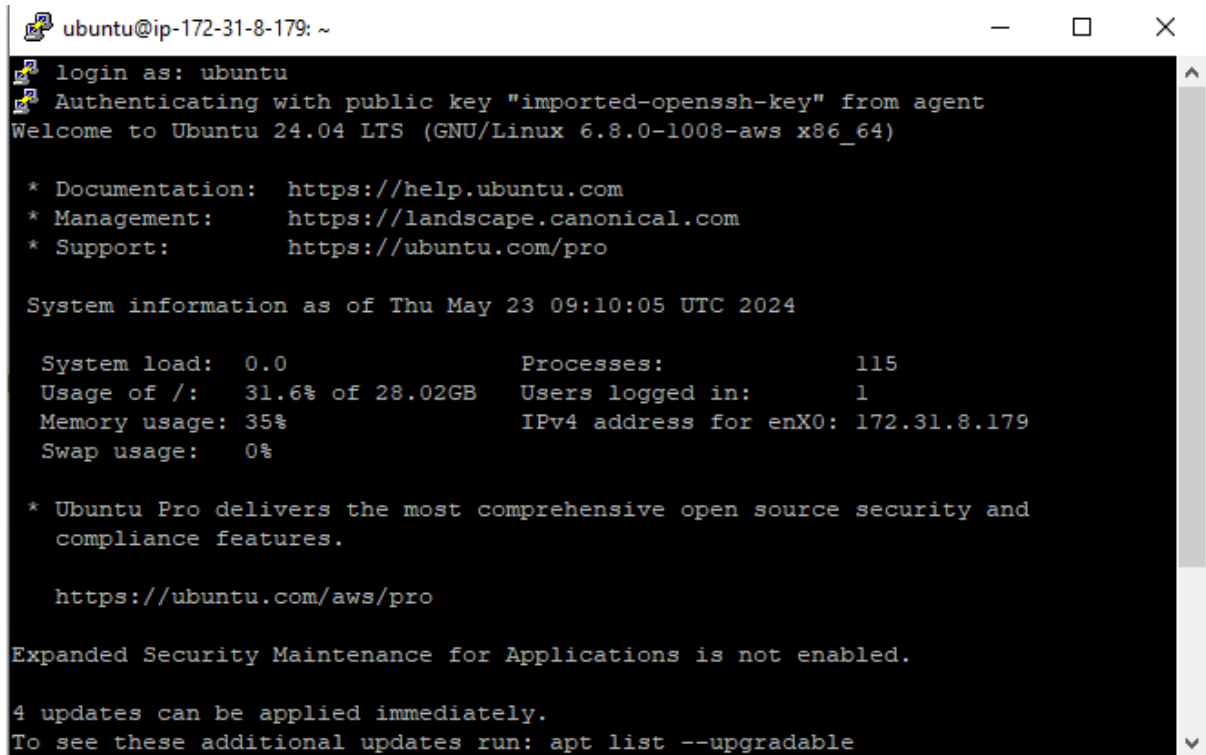
- o Open puTTY ☐ select connection type SSH ☐ enter your VM Ip address ☐ check port number 22 ☐ go to SSH/Auth/credentials



- o Browse the .ppk key_ubuntu from your local machine ☐ click open



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



```
ubuntu@ip-172-31-8-179: ~  
login as: ubuntu  
Authenticating with public key "imported-openssh-key" from agent  
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1008-aws x86_64)  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:       https://ubuntu.com/pro  
  
System information as of Thu May 23 09:10:05 UTC 2024  
  
System load:  0.0                Processes:            115  
Usage of /:   31.6% of 28.02GB   Users logged in:     1  
Memory usage: 35%                IPv4 address for enX0: 172.31.8.179  
Swap usage:   0%  
  
* Ubuntu Pro delivers the most comprehensive open source security and  
  compliance features.  
  
  https://ubuntu.com/aws/pro  
  
Expanded Security Maintenance for Applications is not enabled.  
  
4 updates can be applied immediately.  
To see these additional updates run: apt list --upgradable
```

3. 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

4. Clone the MicroPython source repository by calling

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

5. Go to the MicroPython directory by calling

cd micropython

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

cd ports/stm32

7. build the STM32 firemoware for a specific board by callihng

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

In my case, I use

make BOARD=NUCLEO_F401RE

8. 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
```

9. Now we need get those firware 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

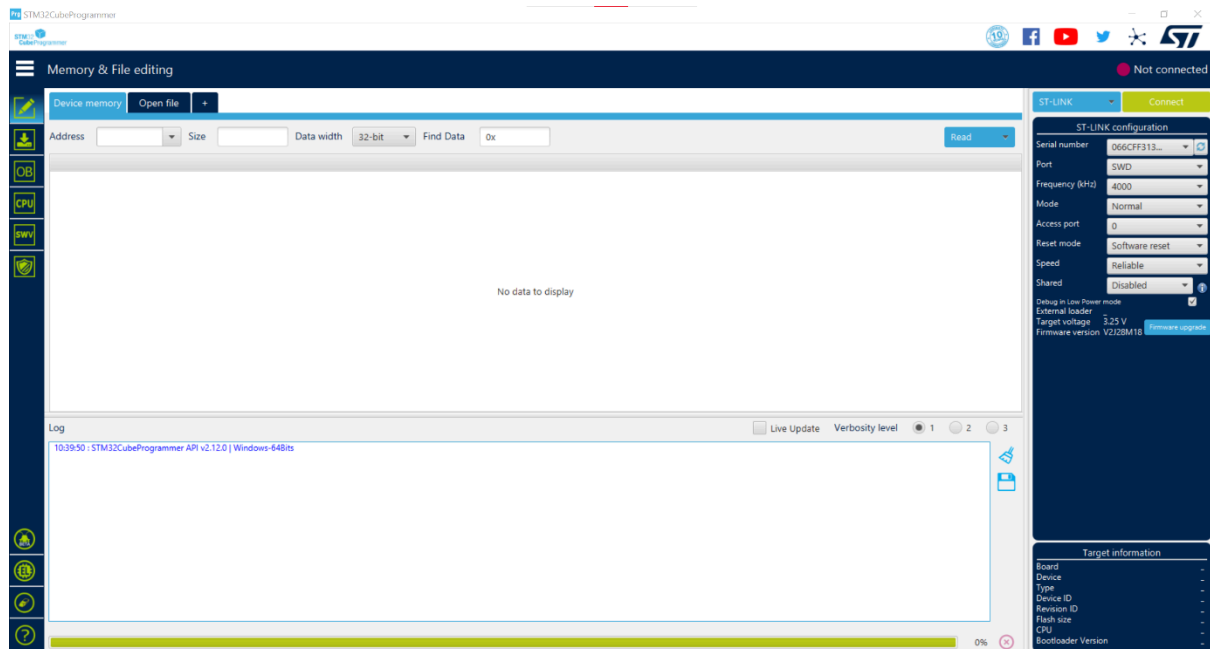
create mode 100755 firmware0.bin

create mode 100755 firmware1.bin

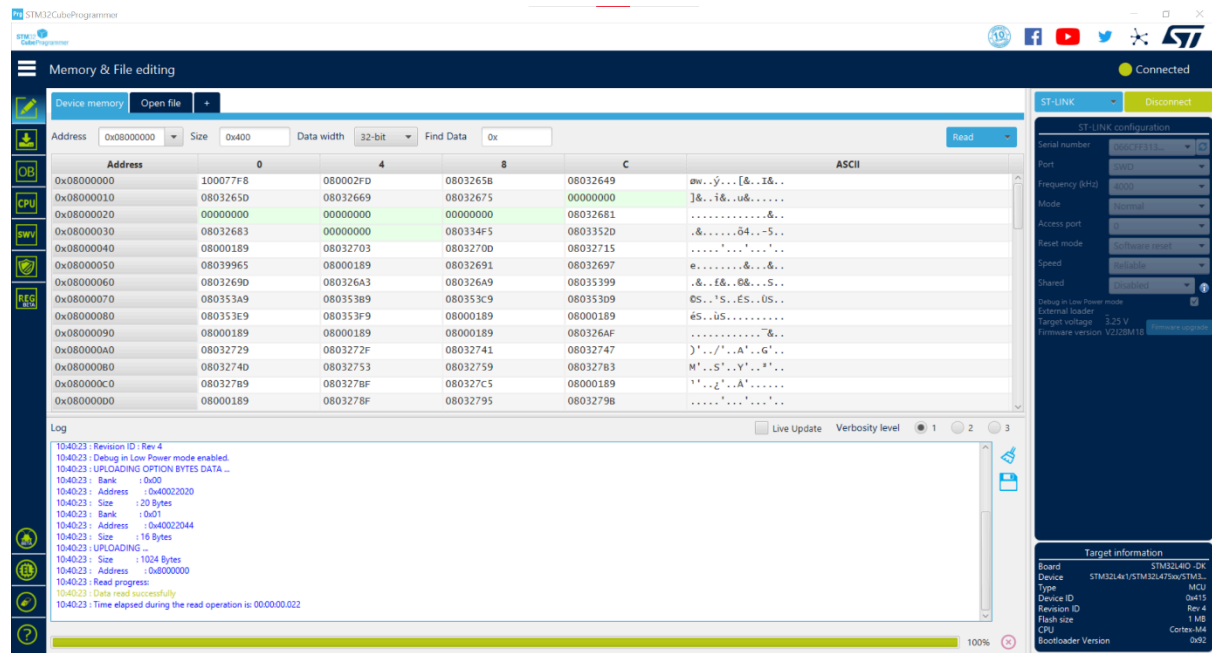
now clone back the same git repo where you stored those into your local machine using
git-bash

1.4 Flashing Micropython to an STM32 Board

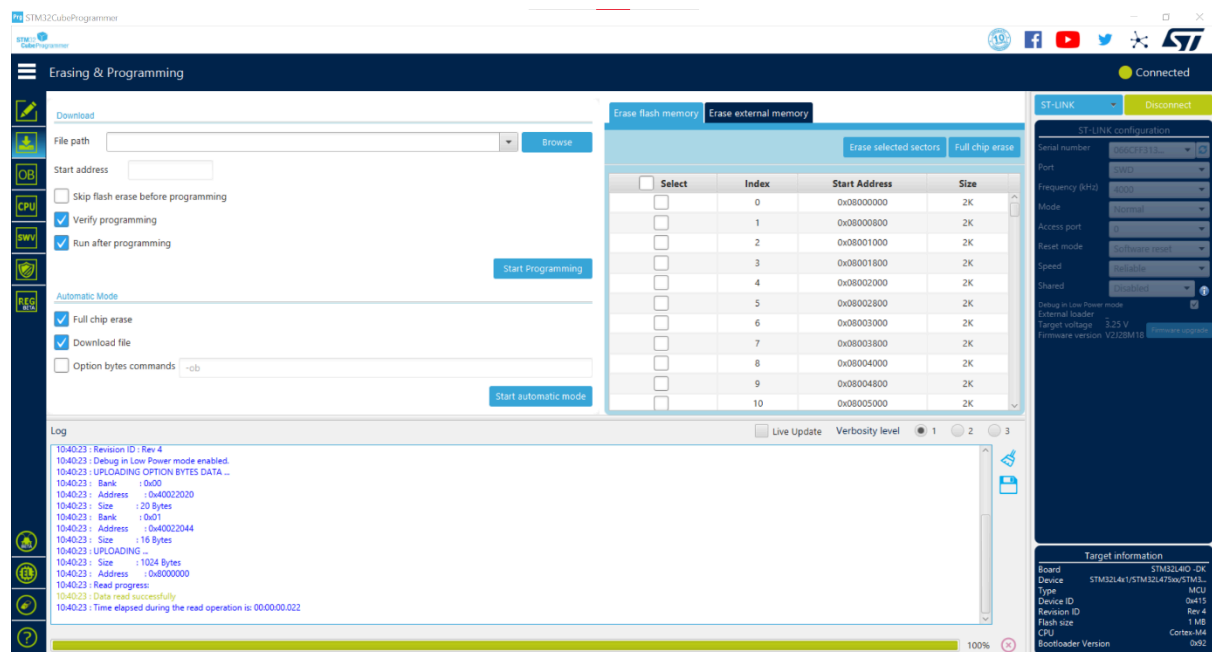
1. Connect the board to the computer through a microUSB cable.



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



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



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

As a result, here are the logs

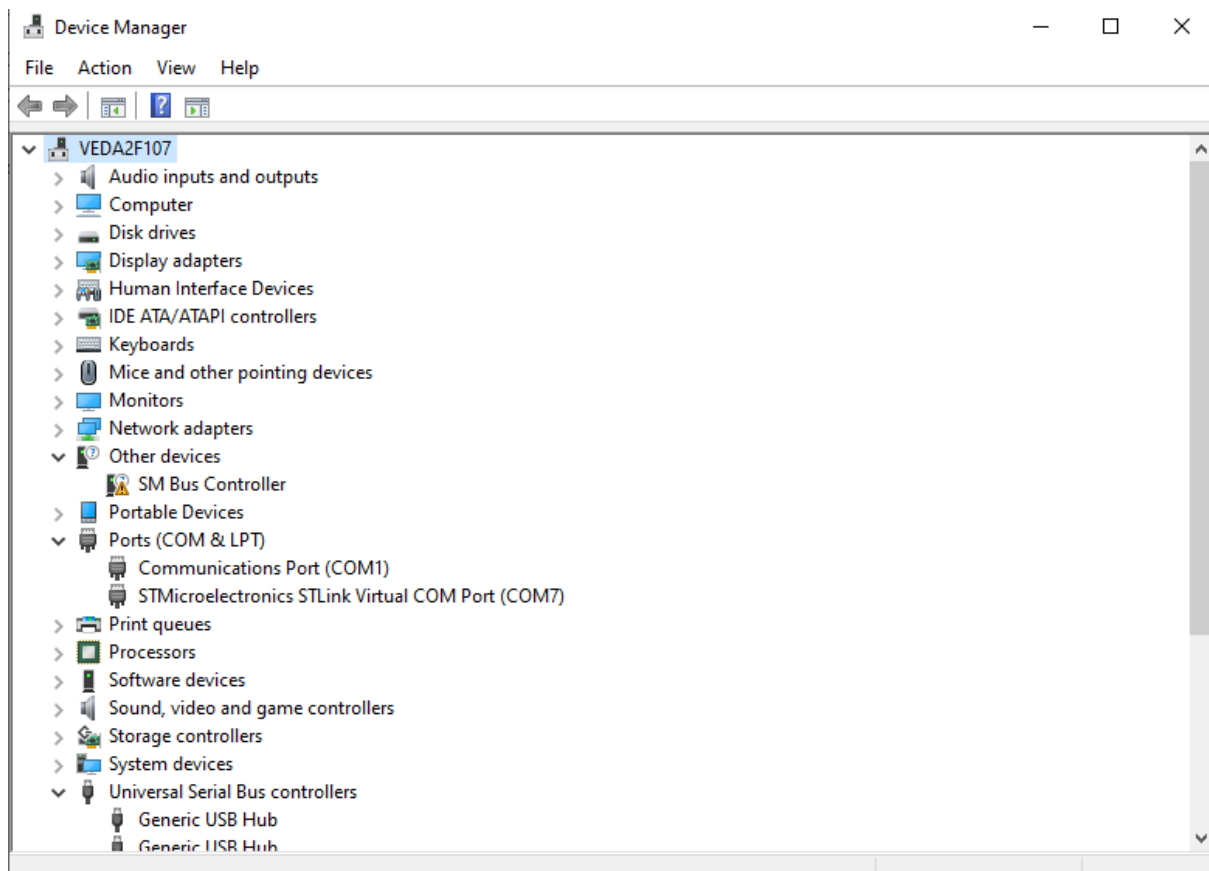
```

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
  
```

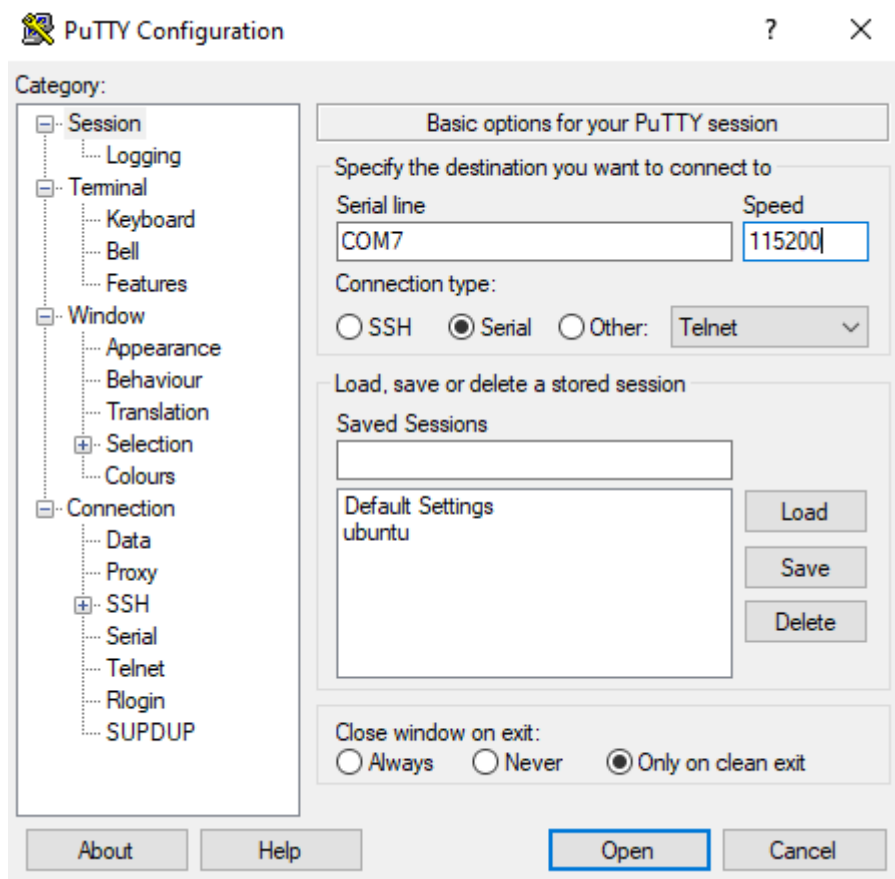

10:44:42 : RUNNING Program ...
10:44:42 : Address: : 0x08000000
10:44:42 : Application is running, Please Hold on...

1.5 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.

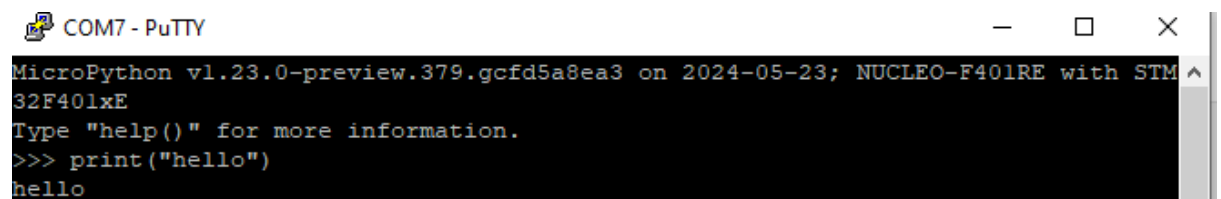


3. Open a serial communication utility software such as PuTTY.
4. Select “Serial” under Connection Type. Enter the COM7 port you recorded into the Serial Line field. Enter 115200 for speed (baudrate). Click Open.



5. 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")
```



6. If the message you tried to print successfully shows up in the terminal, congratulations, you have successfully flashed micropython onto your microcontroller board.