Embedded System Design Techniques™

Rapid Prototyping Embedded Systems using MicroPython

Session 4: Building and Customizing MicroPython

May 5th, 2016

Jacob Beningo, CSDP

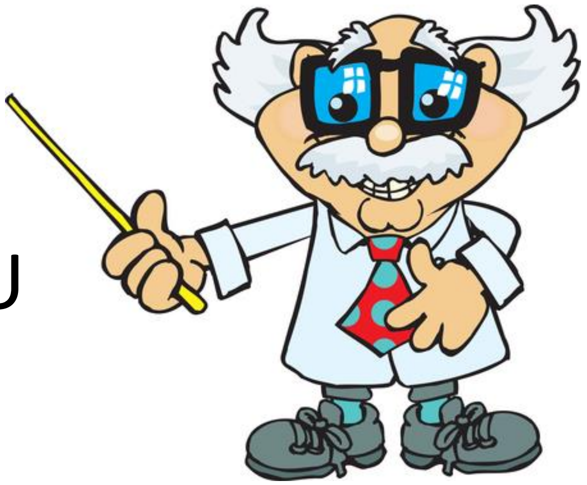
Presented by:

Course Overview

- Introduction to MicroPython
- Libraries and Peripheral Control
- Rapid Prototyping
- **Building and Customizing Micro Python**
- Python Scripting for Testing and Debug

Session Overview

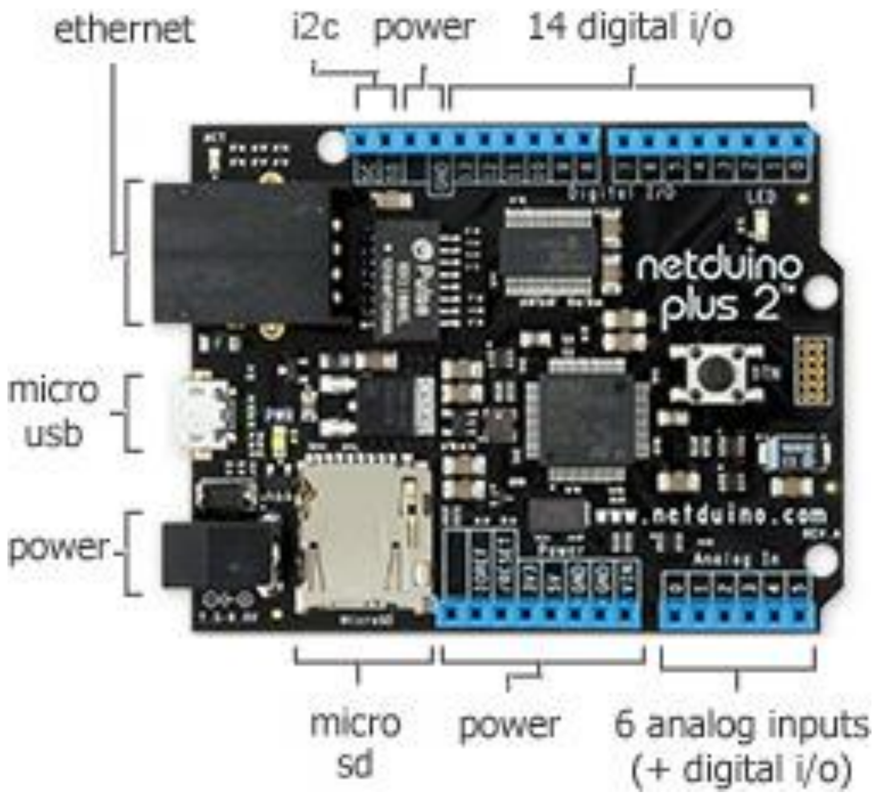
- Our Target Hardware
- Creating a virtual machine
- Installing MicroPython
- Building MicroPython
- Installing DFU Tools
- Installing MicroPython via DFU
- Verifying the installation



Our Target Hardware

Why the netduino plus 2?

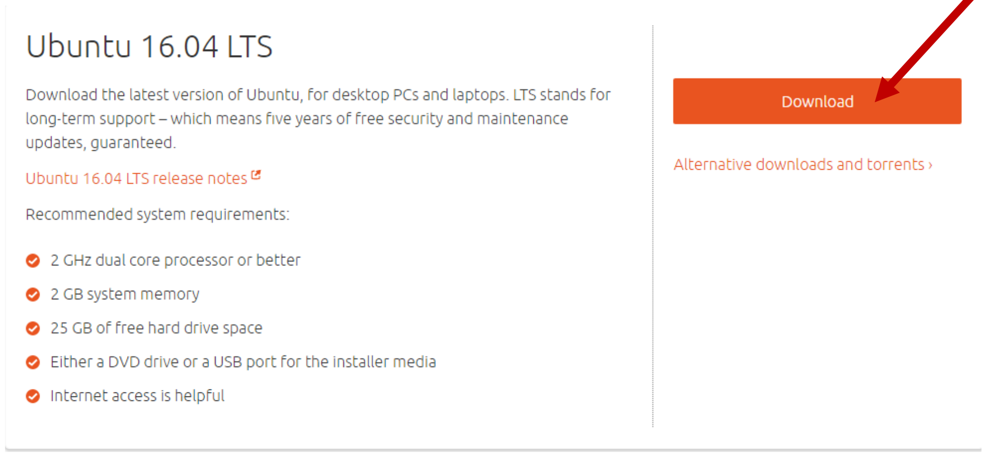
- Arduino form factor
- Same processor as PyBoard
- uSD
- Ethernet Controller
- Micro USB (Terminal)
- 2 LEDs
- Great first step at building and deploying MicroPython



Creating a Virtual Machine

Visit https://www.youtube.com/channel/UC9k8GahBTE0IVJxOsL4WhOA for step by step video instruction.

1) Install a virtual machine tool
   - Virtual Box
   - Vmware
2) Download Ubuntu 16.04 LTS
   - http://www.ubuntu.com/download/desktop



Ubuntu 16.04 LTS

Download the latest version of Ubuntu, for desktop PCs and laptops. LTS stands for long-term support – which means five years of free security and maintenance updates, guaranteed.

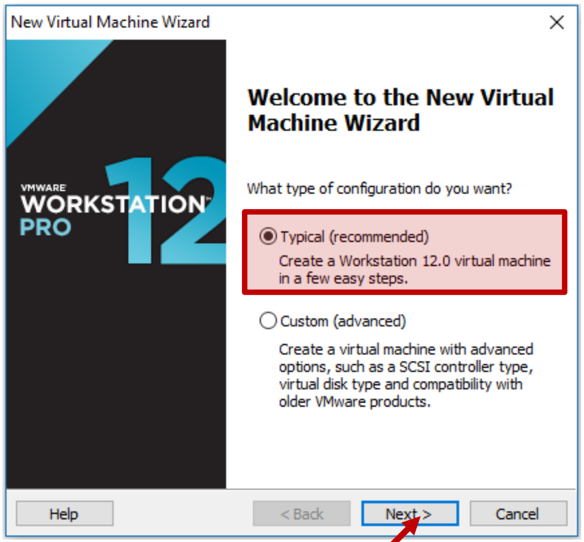
Ubuntu 16.04 LTS release notes

Recommended system requirements:

- 2 GHz dual core processor or better
- 2 GB system memory
- 25 GB of free hard drive space
- Either a DVD drive or a USB port for the installer media
- Internet access is helpful

Creating a Virtual Machine

1



New Virtual Machine Wizard

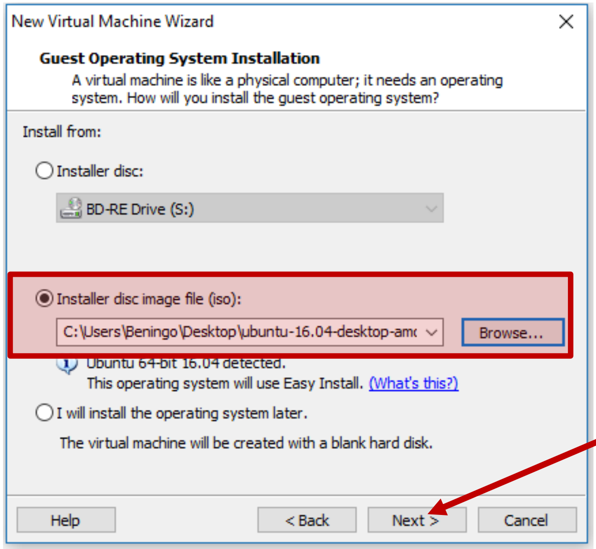
Welcome to the New Virtual Machine Wizard

What type of configuration do you want?

[● Typical (recommended)]
Create a Workstation 12.0 virtual machine in a few easy steps.

○ Custom (advanced)
Create a virtual machine with advanced options, such as a SCSI controller type, virtual disk type and compatibility with older VMware products.

2



New Virtual Machine Wizard

Guest Operating System Installation
A virtual machine is like a physical computer; it needs an operating system. How will you install the guest operating system?

Install from:

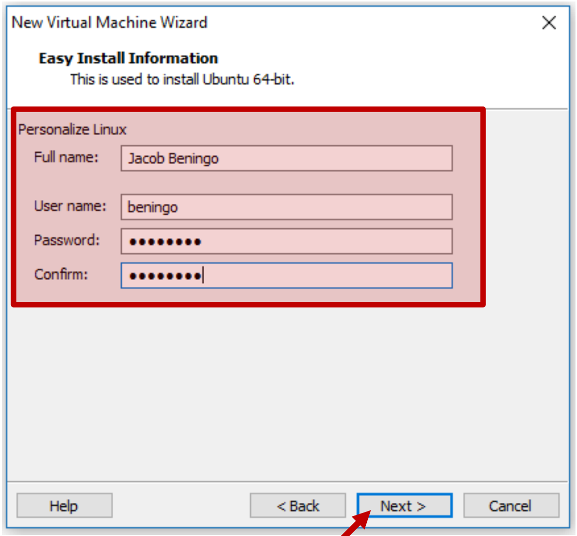
○ Installer disc:
BD-RE Drive (S:)

[● Installer disc image file (iso):]
C:\Users\Beningo\Desktop\ubuntu-16.04-desktop-amc

Ubuntu 64-bit 16.04 detected.
This operating system will use Easy Install. (What's this?)

○ I will install the operating system later.
The virtual machine will be created with a blank hard disk.

3



New Virtual Machine Wizard

Easy Install Information
This is used to install Ubuntu 64-bit.

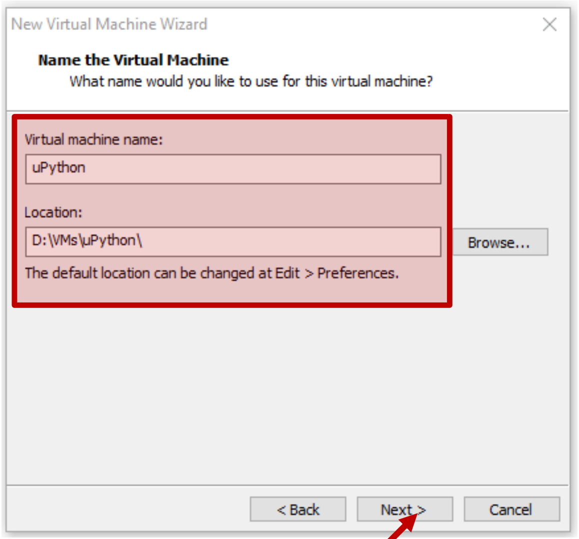
Personalize Linux

Full name:	Jacob Beningo
User name:	beningo
Password:	••••••••
Confirm:	••••••••

Presented by:

Creating a Virtual Machine

4



New Virtual Machine Wizard

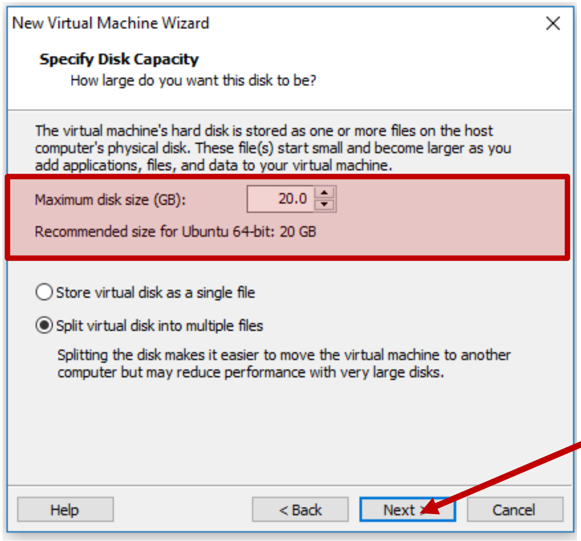
Name the Virtual Machine
What name would you like to use for this virtual machine?

Virtual machine name: uPython

Location: D:\VMs\uPython\

The default location can be changed at Edit > Preferences.

5



New Virtual Machine Wizard

Specify Disk Capacity
How large do you want this disk to be?

The virtual machine's hard disk is stored as one or more files on the host computer's physical disk. These file(s) start small and become larger as you add applications, files, and data to your virtual machine.

Maximum disk size (GB): 20.0

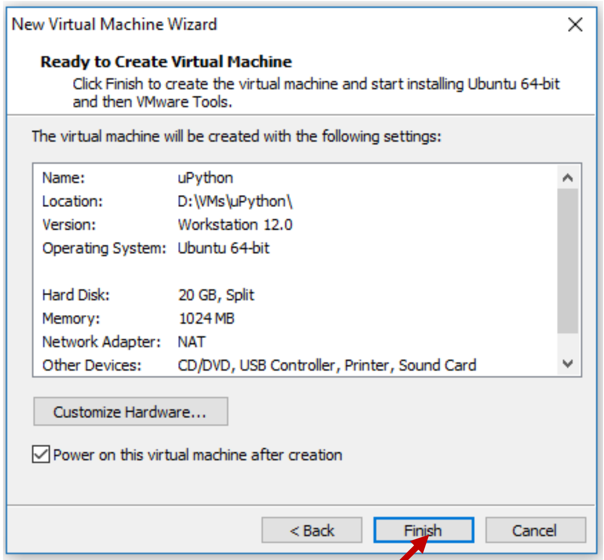
Recommended size for Ubuntu 64-bit: 20 GB

○ Store virtual disk as a single file

● Split virtual disk into multiple files

Splitting the disk makes it easier to move the virtual machine to another computer but may reduce performance with very large disks.

6



New Virtual Machine Wizard

Ready to Create Virtual Machine
Click Finish to create the virtual machine and start installing Ubuntu 64-bit and then VMware Tools.

The virtual machine will be created with the following settings:

Name:	uPython
Location:	D:\VMs\uPython\
Version:	Workstation 12.0
Operating System:	Ubuntu 64-bit
Hard Disk:	20 GB, Split
Memory:	1024 MB
Network Adapter:	NAT
Other Devices:	CD/DVD, USB Controller, Printer, Sound Card

☑ Power on this virtual machine after creation

Installing MicroPython

1) Open a terminal
2) Install gcc toolchain:

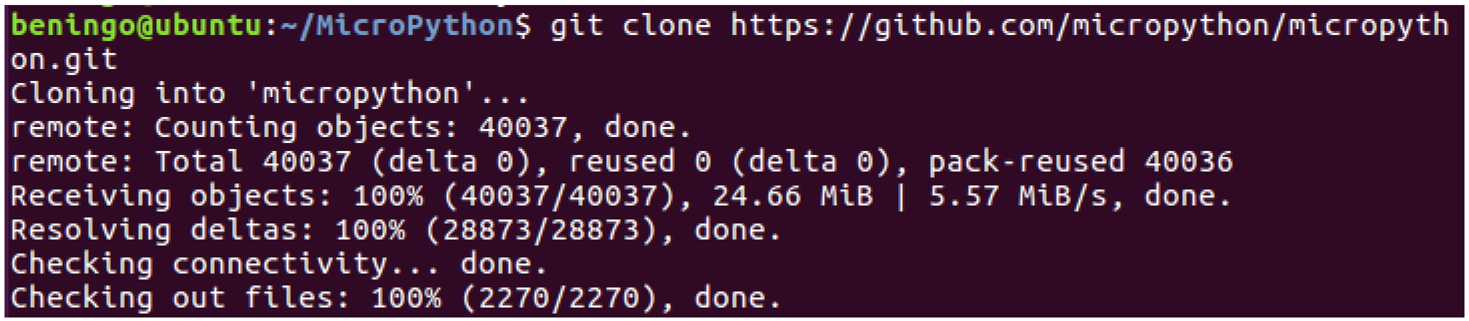
```
sudo apt-get install gcc-arm-none-eabi
```

3) Install git

```
sudo apt-get install git
```

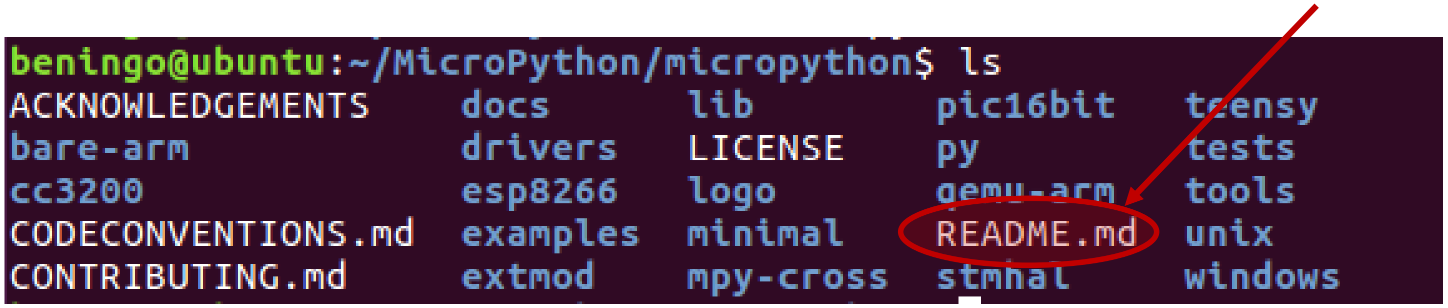
4) Install MicroPython

```
git clone https://github.com/micropython/micropython.git
```



```
beningo@ubuntu:~/MicroPython$ git clone https://github.com/micropython/micropyth
on.git
Cloning into 'micropython'...
remote: Counting objects: 40037, done.
remote: Total 40037 (delta 0), reused 0 (delta 0), pack-reused 40036
Receiving objects: 100% (40037/40037), 24.66 MiB | 5.57 MiB/s, done.
Resolving deltas: 100% (28873/28873), done.
Checking connectivity... done.
Checking out files: 100% (2270/2270), done.
```

MicroPython



```
beningo@ubuntu:~/MicroPython/micropython$ ls
ACKNOWLEDGEMENTS      docs       lib         pic16bit   teensy
bare-arm              drivers    LICENSE     py         tests
cc3200                esp8266    logo        qemu-arm   tools
CODECONVENTIONS.md    examples   minimal     README.md  unix
CONTRIBUTING.md       extmod     mpy-cross   stmhal     windows
```

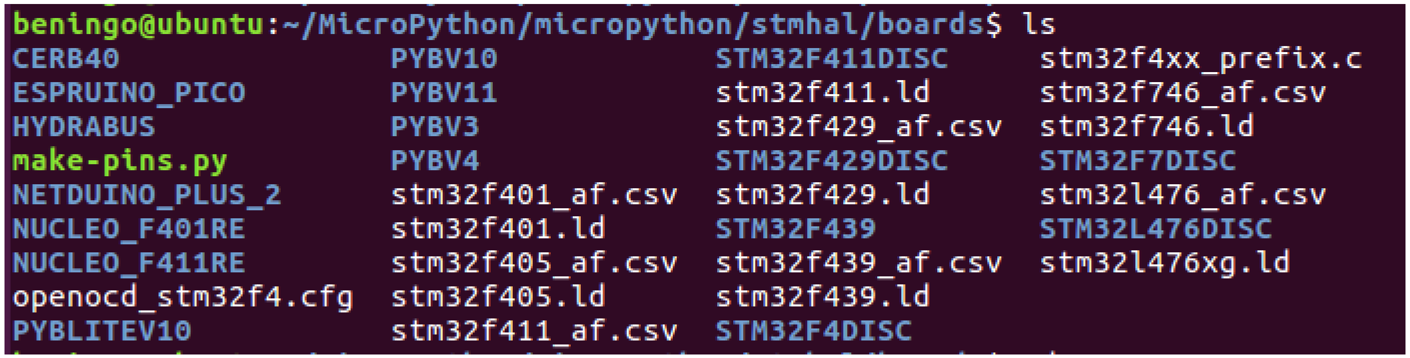
Folder	Purpose
Bare-arm	Minimal version for ARM MCU’s
Teensy	uP version for Teensy 3.1
Pic16bit	uP for 16 bit Microchip parts
Cc3200	uP for CC3200 from TI
Esp8266	uP for esp8266 wifi module
Py	Core Python implementation, compiler, runtime, etc
Stmhal	uP for STM32F405RG using St’s HAL

Building MicroPython

1) Enter the stmhal directory

```
cd stmhal
```

2) Examine boards directory for available board builds

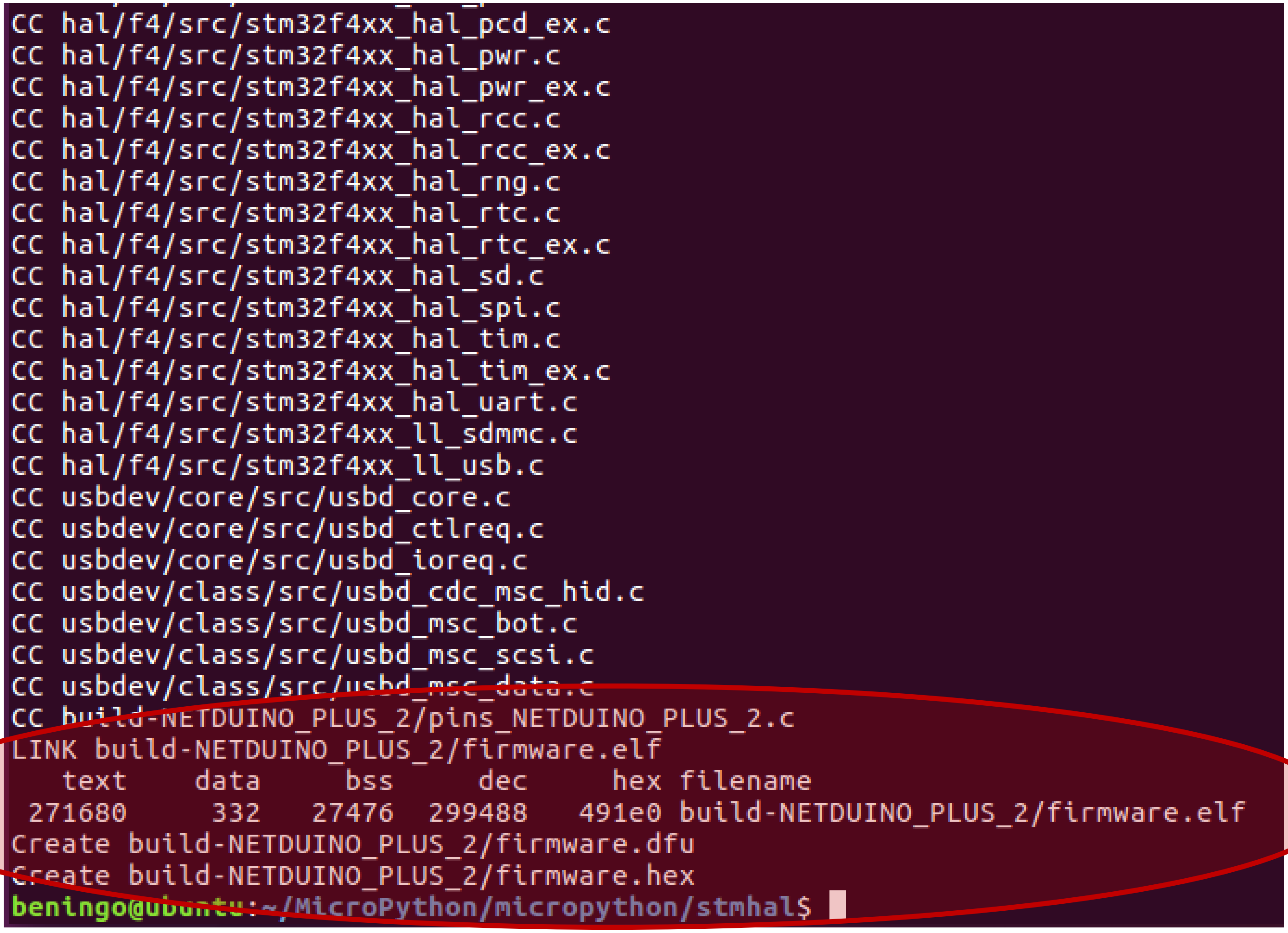


```
beningo@ubuntu:~/MicroPython/micropython/stmhal/boards$ ls
CERB40               PYBV10             STM32F411DISC      stm32f4xx_prefix.c
ESPRUINO_PICO        PYBV11             stm32f411.ld       stm32f746_af.csv
HYDRABUS             PYBV3              stm32f429_af.csv   stm32f746.ld
make-pins.py         PYBV4              STM32F429DISC      STM32F7DISC
NETDUINO_PLUS_2      stm32f401_af.csv   stm32f429.ld       stm32l476_af.csv
NUCLEO_F401RE        stm32f401.ld       STM32F439          STM32L476DISC
NUCLEO_F411RE        stm32f405_af.csv   stm32f439_af.csv   stm32l476xg.ld
openocd_stm32f4.cfg  stm32f405.ld       stm32f439.ld
PYBLITEV10           stm32f411_af.csv   STM32F4DISC
```

3) Build MicroPython

```
make BOARD=NETDUINO_PLUS_2
```

Building MicroPython



```
CC hal/f4/src/stm32f4xx_hal_pcd_ex.c
CC hal/f4/src/stm32f4xx_hal_pwr.c
CC hal/f4/src/stm32f4xx_hal_pwr_ex.c
CC hal/f4/src/stm32f4xx_hal_rcc.c
CC hal/f4/src/stm32f4xx_hal_rcc_ex.c
CC hal/f4/src/stm32f4xx_hal_rng.c
CC hal/f4/src/stm32f4xx_hal_rtc.c
CC hal/f4/src/stm32f4xx_hal_rtc_ex.c
CC hal/f4/src/stm32f4xx_hal_sd.c
CC hal/f4/src/stm32f4xx_hal_spi.c
CC hal/f4/src/stm32f4xx_hal_tim.c
CC hal/f4/src/stm32f4xx_hal_tim_ex.c
CC hal/f4/src/stm32f4xx_hal_uart.c
CC hal/f4/src/stm32f4xx_ll_sdmmc.c
CC hal/f4/src/stm32f4xx_ll_usb.c
CC usbdev/core/src/usbd_core.c
CC usbdev/core/src/usbd_ctlreq.c
CC usbdev/core/src/usbd_ioreq.c
CC usbdev/class/src/usbd_cdc_msc_hid.c
CC usbdev/class/src/usbd_msc_bot.c
CC usbdev/class/src/usbd_msc_scsi.c
CC usbdev/class/src/usbd_msc_data.c
CC build-NETDUINO_PLUS_2/pins_NETDUINO_PLUS_2.c
LINK build-NETDUINO_PLUS_2/firmware.elf
   text    data     bss     dec     hex filename
 271680     332   27476  299488   491e0 build-NETDUINO_PLUS_2/firmware.elf
Create build-NETDUINO_PLUS_2/firmware.dfu
Create build-NETDUINO_PLUS_2/firmware.hex
beningo@ubuntu:~/MicroPython/micropython/stmhal$
```

Presented by:

Installing DFU Tools

1) Install DFU Utilities

```
sudo apt-get install dfu-util
```

2) Create a udev rules files

```
sudo nano /etc/udev/rules.d/49-stmdiscovery.rules
```

3) Enter the rules content

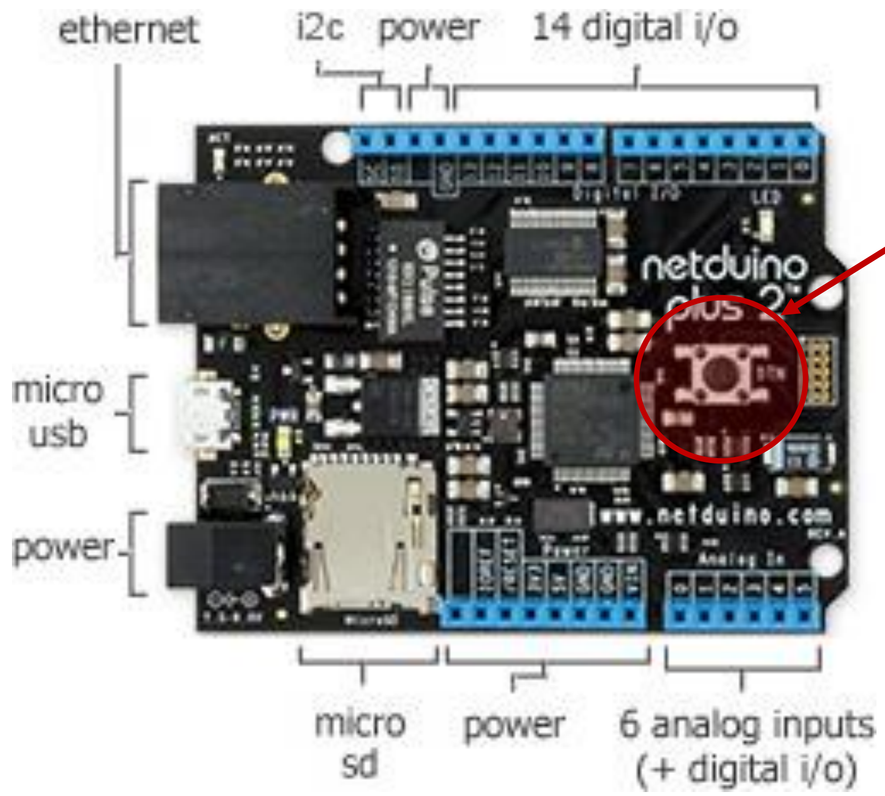
```
# f055:9800 - STM32F4 Discovery running MicroPython in USB Serial Mode (CN5)
ATTRS{idVendor}=="f055", ATTRS{idProduct}=="9800", ENV{ID_MM_DEVICE_IGNORE}="1"
ATTRS{idVendor}=="f055", ATTRS{idProduct}=="9800", ENV{MTP_NO_PROBE}="1"
SUBSYSTEMS=="usb", ATTRS{idVendor}=="f055", ATTRS{idProduct}=="9800", MODE:="0666"
KERNEL=="ttyACM*", ATTRS{idVendor}=="f055", ATTRS{idProduct}=="9800", MODE:="0666"
# 0483:df11 - STM32F4 Discovery in DFU mode (CN5)
SUBSYSTEMS=="usb", ATTRS{idVendor}=="0483", ATTRS{idProduct}=="df11", MODE:="0666"
```

4) Reload the udev rules

```
sudo udevadm control –reload-rules
```

Installing MicroPython via DFU

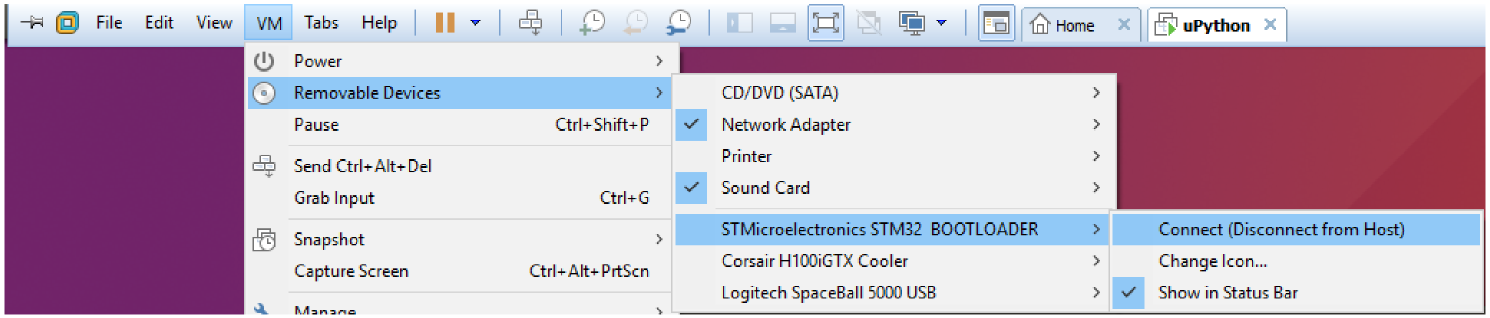
1) Enter bootloader mode



Press and hold during power-up

Installing MicroPython via DFU

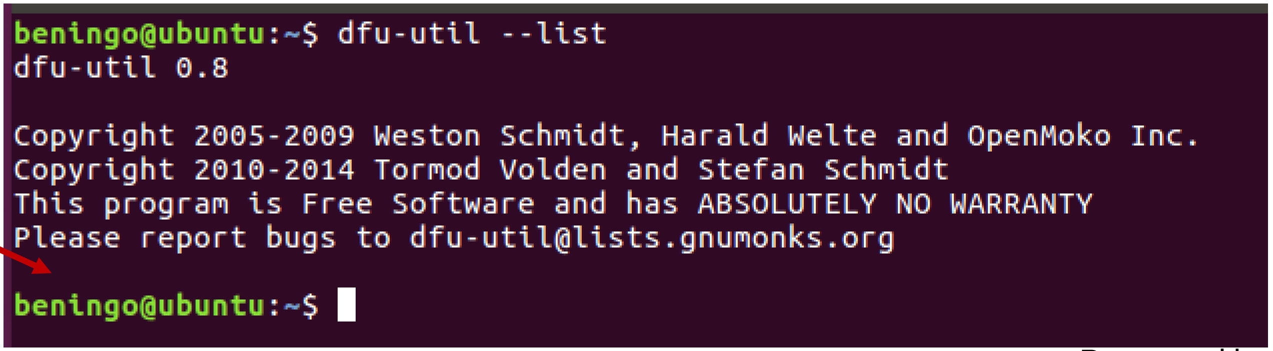
2) Connect Bootloader to Linux VM



3) Get the connect dfu device list using

```
dfu-util --list
```

Where's my list?!

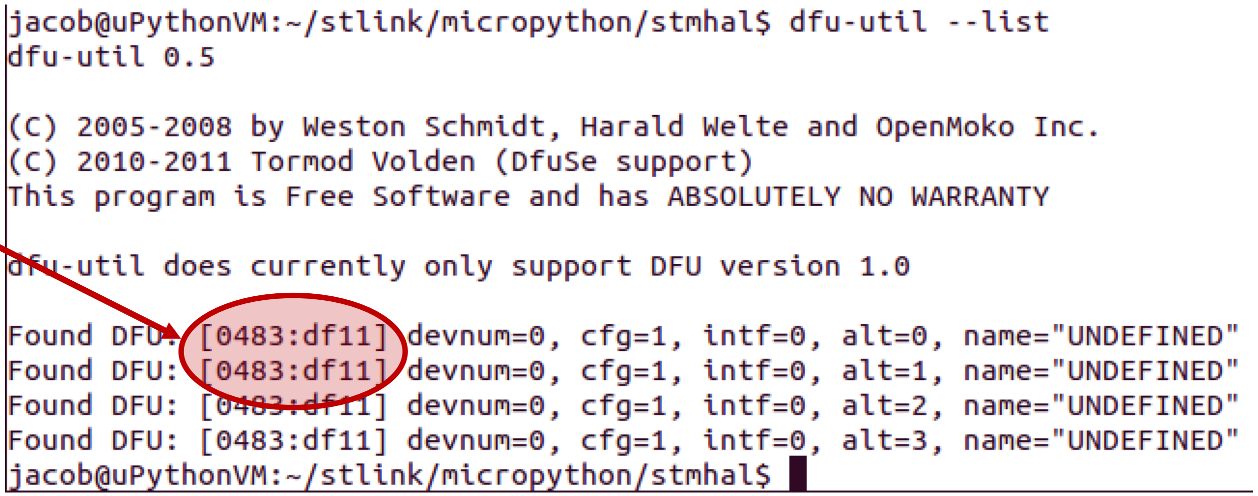


```
beningo@ubuntu:~$ dfu-util --list
dfu-util 0.8

Copyright 2005-2009 Weston Schmidt, Harald Welte and OpenMoko Inc.
Copyright 2010-2014 Tormod Volden and Stefan Schmidt
This program is Free Software and has ABSOLUTELY NO WARRANTY
Please report bugs to dfu-util@lists.gnumonks.org

beningo@ubuntu:~$ 
```

Installing MicroPython via DFU

3) Get the connect dfu device list using

```
dfu-util --list
```



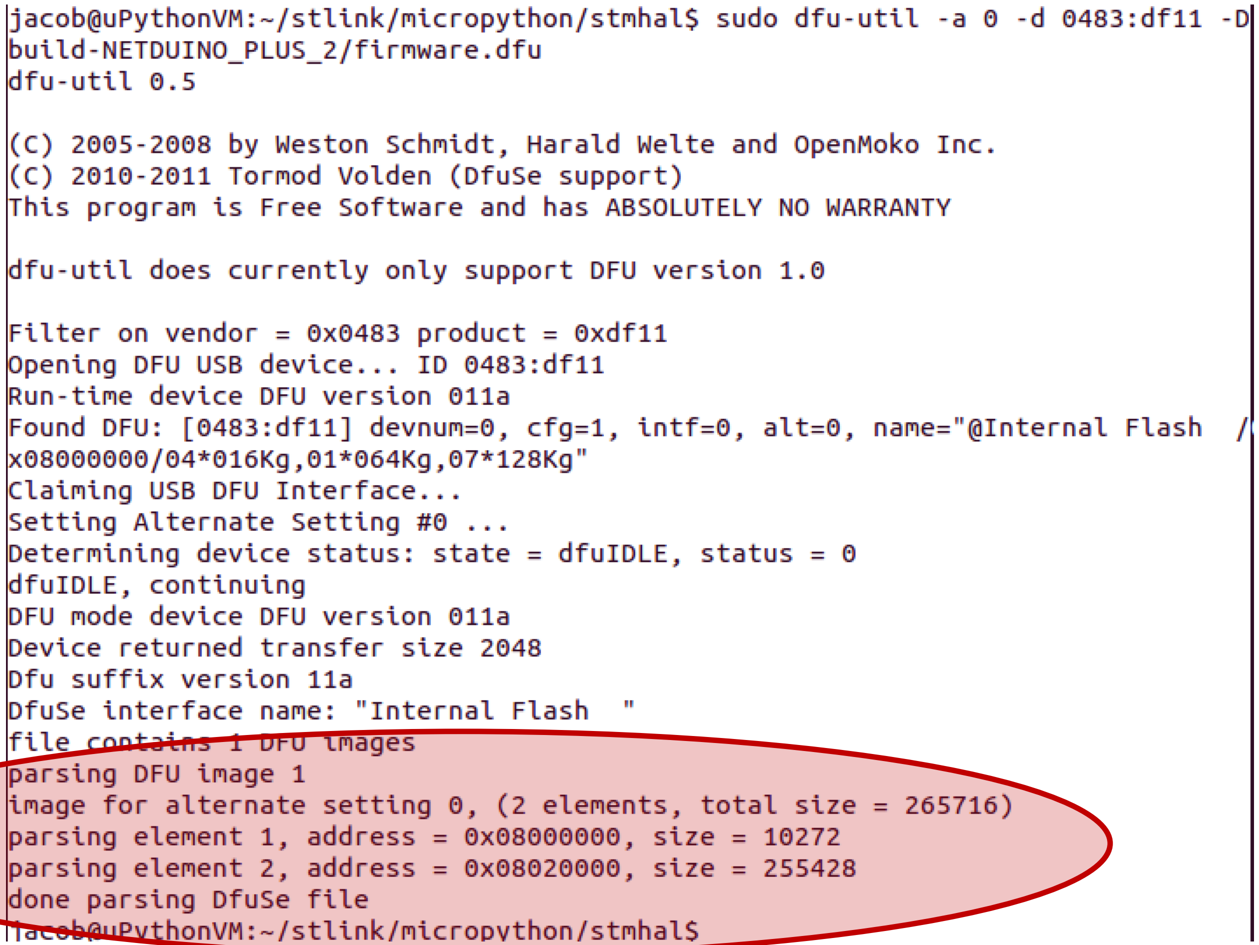
Device ID

```
jacob@uPythonVM:~/stlink/micropython/stmhal$ dfu-util --list
dfu-util 0.5

(C) 2005-2008 by Weston Schmidt, Harald Welte and OpenMoko Inc.
(C) 2010-2011 Tormod Volden (DfuSe support)
This program is Free Software and has ABSOLUTELY NO WARRANTY

dfu-util does currently only support DFU version 1.0

Found DFU: [0483:df11] devnum=0, cfg=1, intf=0, alt=0, name="UNDEFINED"
Found DFU: [0483:df11] devnum=0, cfg=1, intf=0, alt=1, name="UNDEFINED"
Found DFU: [0483:df11] devnum=0, cfg=1, intf=0, alt=2, name="UNDEFINED"
Found DFU: [0483:df11] devnum=0, cfg=1, intf=0, alt=3, name="UNDEFINED"
jacob@uPythonVM:~/stlink/micropython/stmhal$
```

4) Run the update command

```
dfu-util -a 0 -d 0483:df11 -D build-NETDUINO_PLUS_2/firmware.dfu
```

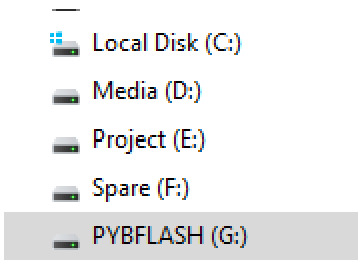
Installing MicroPython via DFU



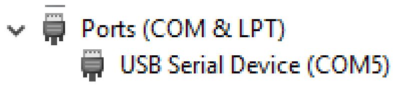
```
jacob@uPythonVM:~/stlink/micropython/stmhal$ sudo dfu-util -a 0 -d 0483:df11 -D
build-NETDUINO_PLUS_2/firmware.dfu
dfu-util 0.5

(C) 2005-2008 by Weston Schmidt, Harald Welte and OpenMoko Inc.
(C) 2010-2011 Tormod Volden (DfuSe support)
This program is Free Software and has ABSOLUTELY NO WARRANTY

dfu-util does currently only support DFU version 1.0

Filter on vendor = 0x0483 product = 0xdf11
Opening DFU USB device... ID 0483:df11
Run-time device DFU version 011a
Found DFU: [0483:df11] devnum=0, cfg=1, intf=0, alt=0, name="@Internal Flash  /
x08000000/04*016Kg,01*064Kg,07*128Kg"
Claiming USB DFU Interface...
Setting Alternate Setting #0 ...
Determining device status: state = dfuIDLE, status = 0
dfuIDLE, continuing
DFU mode device DFU version 011a
Device returned transfer size 2048
Dfu suffix version 11a
DfuSe interface name: "Internal Flash  "
file contains 1 DFU images
parsing DFU image 1
image for alternate setting 0, (2 elements, total size = 265716)
parsing element 1, address = 0x08000000, size = 10272
parsing element 2, address = 0x08020000, size = 255428
done parsing DfuSe file
jacob@uPythonVM:~/stlink/micropython/stmhal$
```

Verify MicroPython Install

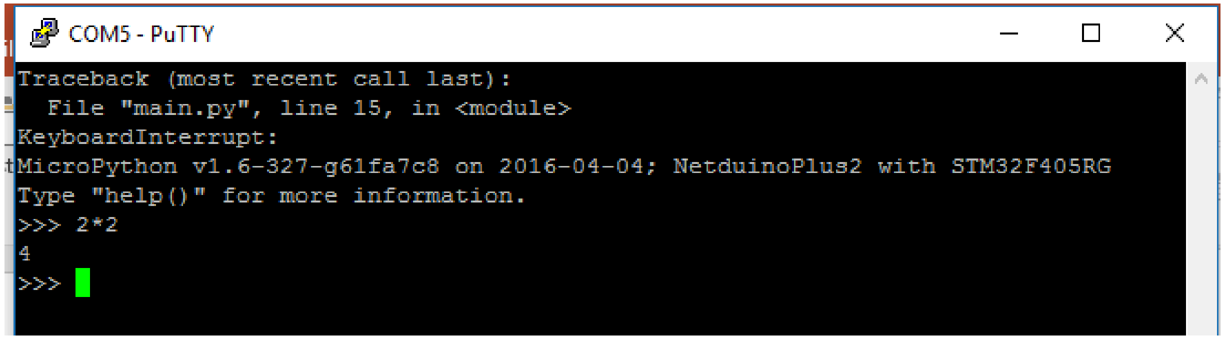
1) Verify when plugging into USB, PYBFLASH shows up



2) Verify the board shows up as a COM port



3) Verify REPL connection

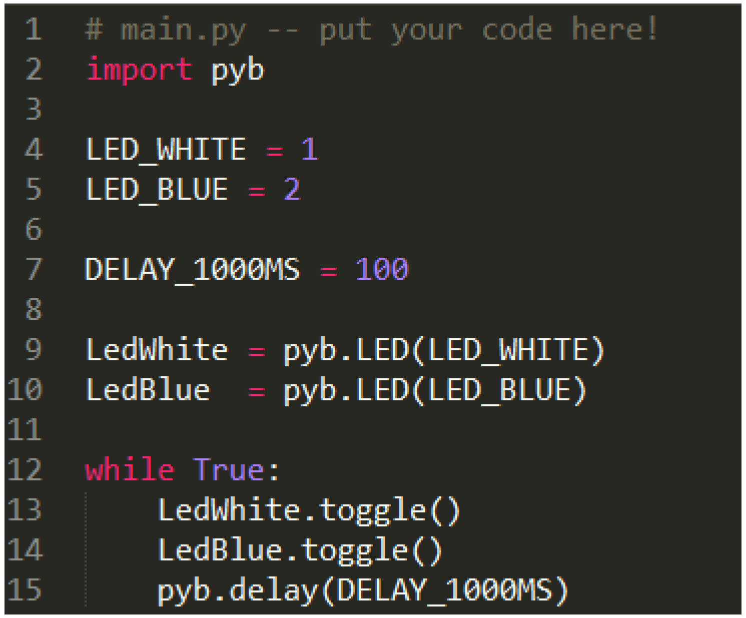


```
Traceback (most recent call last):
  File "main.py", line 15, in <module>
KeyboardInterrupt:
MicroPython v1.6-327-g61fa7c8 on 2016-04-04; NetduinoPlus2 with STM32F405RG
Type "help()" for more information.
>>> 2*2
4
>>> 
```

Presented by:

Verify MicroPython Install

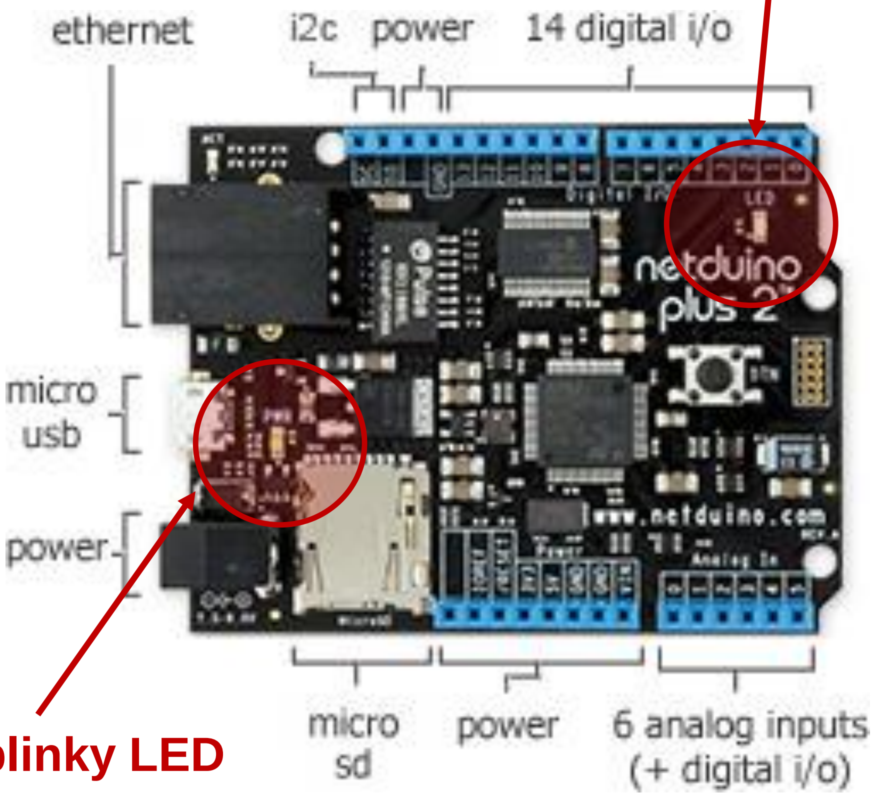
3) Write a short script to blink some



```
# main.py -- put your code here!
import pyb

LED_WHITE = 1
LED_BLUE = 2

DELAY_1000MS = 100

LedWhite = pyb.LED(LED_WHITE)
LedBlue  = pyb.LED(LED_BLUE)

while True:
    LedWhite.toggle()
    LedBlue.toggle()
    pyb.delay(DELAY_1000MS)
```

Blue blinky LED

White blinky LED



Additional Resources

- Download Course Material for
  - Updated C Doxygen Templates (Sept 2015)
  - Example source code
  - Templates
- Microcontroller API Standard
- EDN Embedded Basics Articles
- Embedded Bytes Newsletter
  - http://bit.ly/1BAHYXm

From www.beningo.com under

- Blog > CEC Rapid Prototyping with MicroPython

The Lecturer – Jacob Beningo

Jacob Beningo
Principal Consultant

Social Media / Contact

E : jacob@beningo.com
T : 248-719-6850
: Jacob_Beningo
: Beningo Engineering
: JacobBeningo
EDN : Embedded Basics

CONSULTING

- Secure Bootloaders
- Code Reviews
- Architecture Design
- Real-time Software
- Expert Firmware Analysis

EMBEDDED TRAINING

DOULOS CERTIFIED TRAINING PARTNER

BENINGO EMBEDDED GROUP

www.beningo.com

Presented by:

DesignNews

CEC CONTINUING EDUCATION CENTER

Digi-Key ELECTRONICS