Academic Program Workshop: CircuitPython PyKit Explorer

A Leading Provider of Smart, Connected and Secure Embedded Control Solutions

Who Is Microchip Technology?

What Do We Do?

Microchip At a Glance

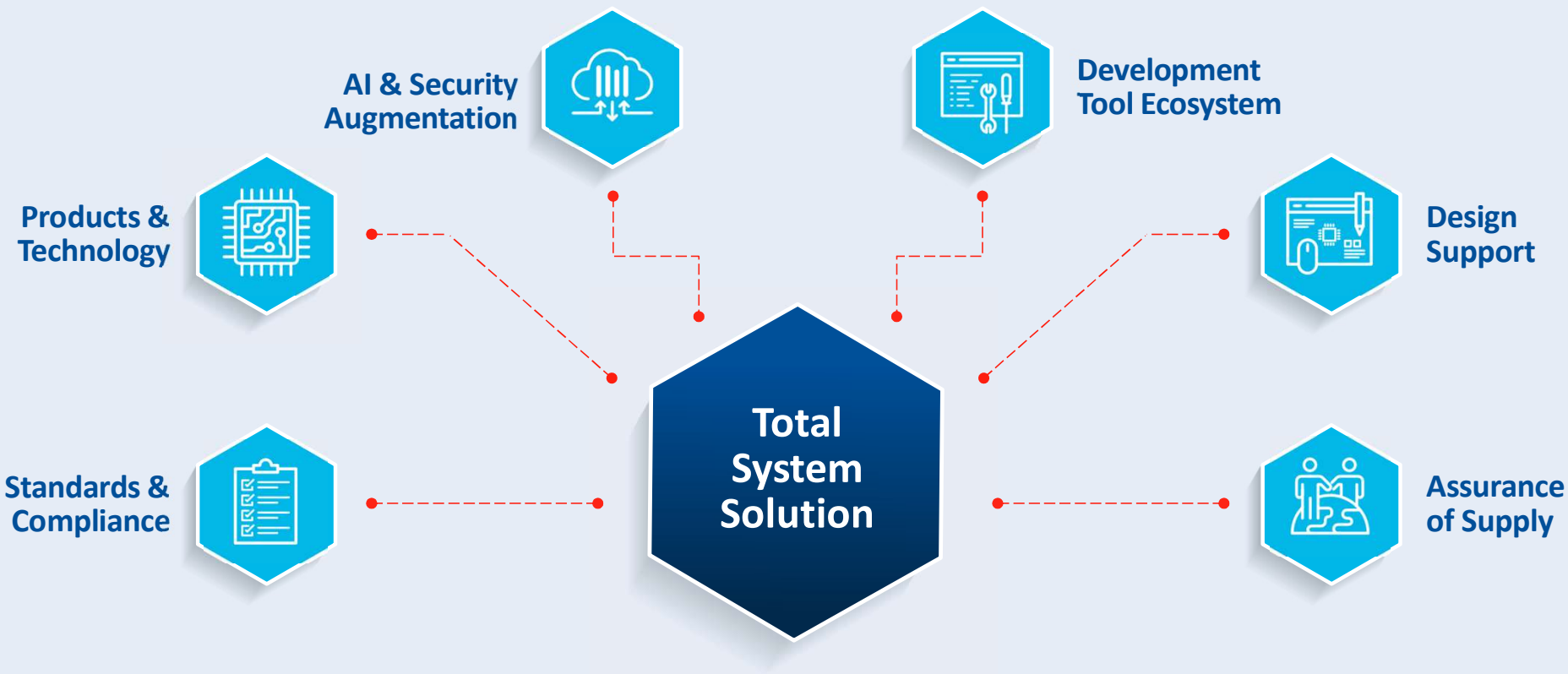
Founded
February 14, 1989

Headquartered in
Chandler, AZ
'The Silicon Desert'

17,000+
Employees

100,000+
Product Offerings

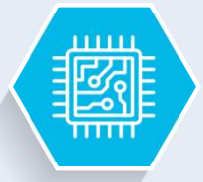
Making Innovative Design Easier



An integrated portfolio of products, software, design tools and organizational support to make your design easier.

MICROCHIP

Our Portfolio



Processors

8-, 16- & 32-Bit MCUs
Digital Signal Controllers
32- & 64-bit Microprocessors
FPGAs PLDs

Connectivity

CAN, CAN2 and CAN FD
Bluetooth, Wi-Fi®, Zigbee
Ethernet, Optical Networking
RF Modules & Microwave Solutions

Touch

MaxTouch® Touchscreen Controllers
GestIC® Air Gesture Controllers
Proximity Touchscreens
MCUs with Capacitive Touch

Sensor & Motor Drive

Current/Voltage/Power Monitor ICS
CO/Smoke Detector Horn Driver ICs
Motor Drivers
Temperature Sensors

Security Products

Authentication ICs
Root of Trust Controllers
MCUs w/ Integrated Security
Secure FPGAs and SoC FPGAs
Trust Platforms

Interface & Connectivity

CAN, CAN2 and CAN FD
Ethernet Switches, PHYs, Controllers
Power-over-Ethernet (PoE)
LIN Bus Networking
Line Circuits and Drivers
PCIe™ Switches
USB Hubs, Switches and Transceivers
USB-C Power Delivery Controllers

Analog & Mixed Signal

In Amps, Op Amps & PGAs
DACs & ADCs
AC/DC Power & Energy Devices
Current/Voltage/Power Monitors
Digital Potentiometers
Voltage References

Power Management

Silicon Carbide Products
MOSFETs
LDOs and Battery Management
DC-DC Switching Converters
Voltage Supervisors & References
Battery Chargers Gate Drivers
AC-DC Power Converters
Electromechanical Voltage Relays

Clock & Timing

Atomic Clocks
GNSS Disciplined Oscillators
Clock Generators and Buffers
Oscillators
Real-Time Clocks
Jitter Attenuators
PCIe™ Timers
High Performance Timing Systems

Memory & Storage

Serial EERAM & Serial EEPROM
Serial & Parallel FLASH Memory
Serial SRAM & Serial NVSRAM
Adaptec Host Bus Adaptor
Adaptec SmartRAID RAID Adaptors
Smart IOC I/O Controllers
SXP SAS Expanders

MICROCHIP

How Many Microchip Parts Are In This Kit?

Total System Solution

- **Voltage Regulators**
  - MCP1826S – 3.3V output
  - MIC5309 – 1.8V output

Power Management

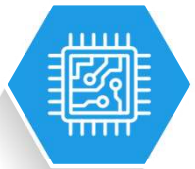
LDO

- **Interface**
  - ATA6561 CAN Transceiver

Interface & Connectivity

CAN

- **Microcontroller**
  - ATSAME51 32-bit MCU



Processors

32-Bit MCU

Touch

MCUs with Capacitive Touch

- **Bluetooth Low Energy (BLE)**
  - RNBD451 BLE Module

Connectivity

CAN, Bluetooth

- **Flash Memory**
  - SST26VF032B – 4MB Flash storage

Memory & Storage

Quad Serial FLASH Memory

MICROCHIP

What Is This Workshop About?

Microchip Academic Program

- **Introduction to Embedded Systems Programming**
  - CircuitPython programming
  - CircuitPython is a fork of MicroPython
  - Both are Python implementations optimized for microcontrollers

- **What is an Embedded System?**
  - Specialized computer system with a dedicated function
  - Combines hardware and software
  - Controls a specific task within a larger mechanical or electronic device

MICROCHIP

Difference between FPGAs and MCUs

Field Programmable Gate Arrays vs Microcontrollers

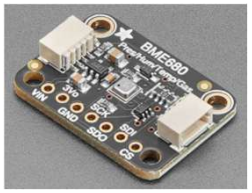
- **Each solves different problems**

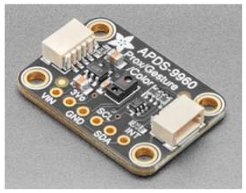
- **Microcontroller:**
  - Sequential-access processor firmware
- **FPGA:**
  - Instructions and config data defines its functionality
  - Parallel processing
- **How to decide which to use?**
  - Design requirements always drive implementation

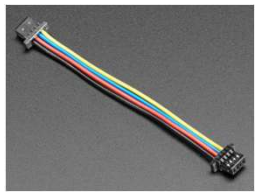
MICROCHIP

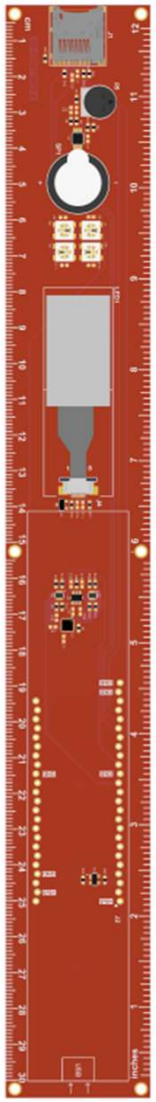
What's In The Kit?

- **Curiosity CircuitPython dev board**
- **PyKit Ruler**
- **Adafruit BME680 Breakout:**
  - Temp, Humidity, Pressure, Gas sensor
- **Adafruit APDS9960 Breakout:**
  - Proximity, Gesture, Color sensor
- **QWIIC / STEMMA cable**
- **USB-A to USB-C cable**
- **Header Pins to connect Ruler and dev board**
  - 1 x 23-pin, 1 x 25-pin





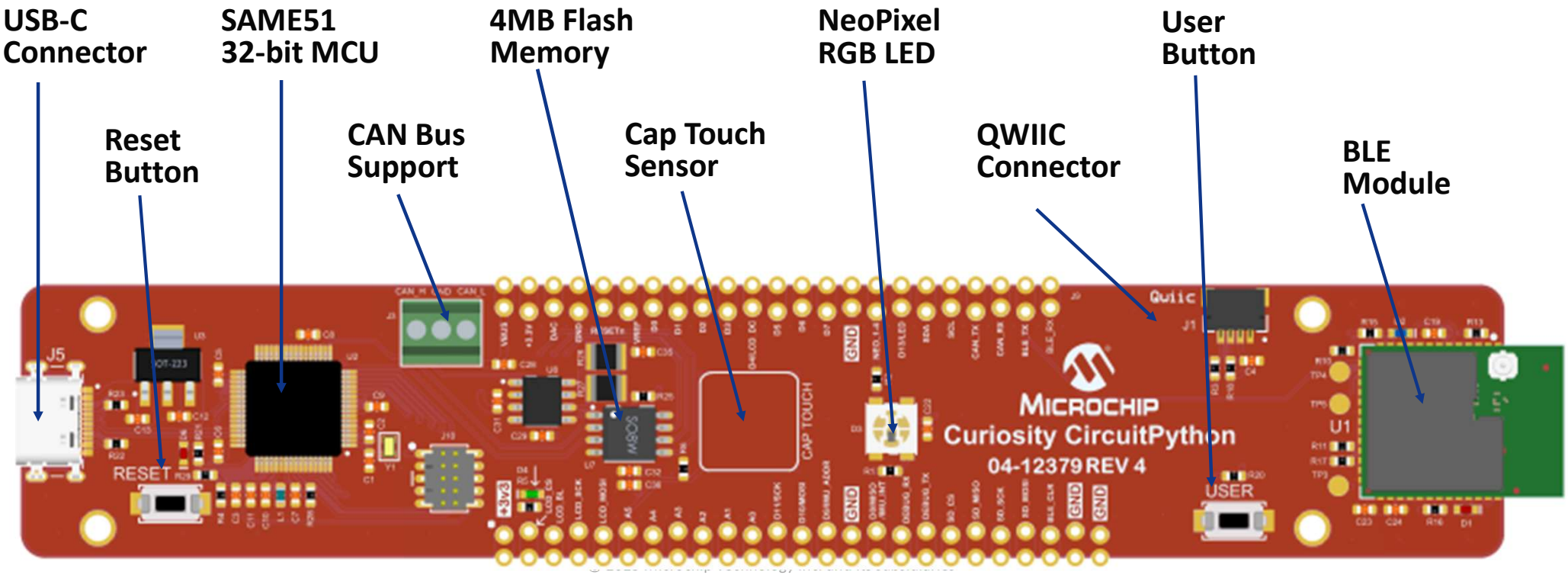




MICROCHIP

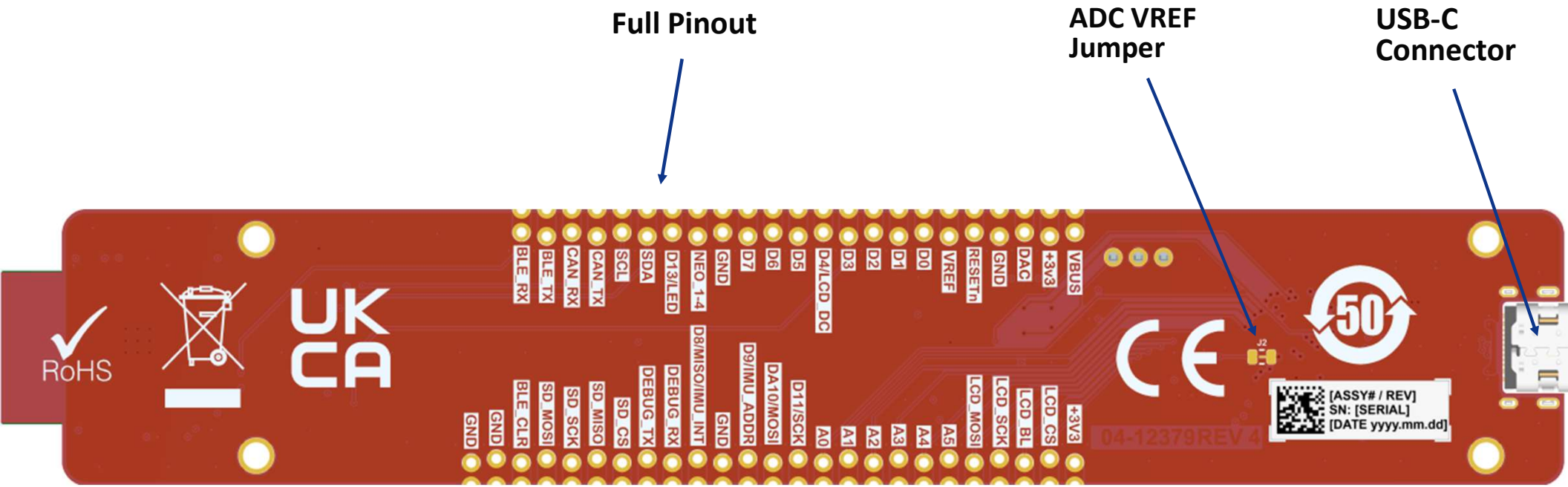
What Hardware Does The Kit Have?

Curiosity CircuitPython dev board



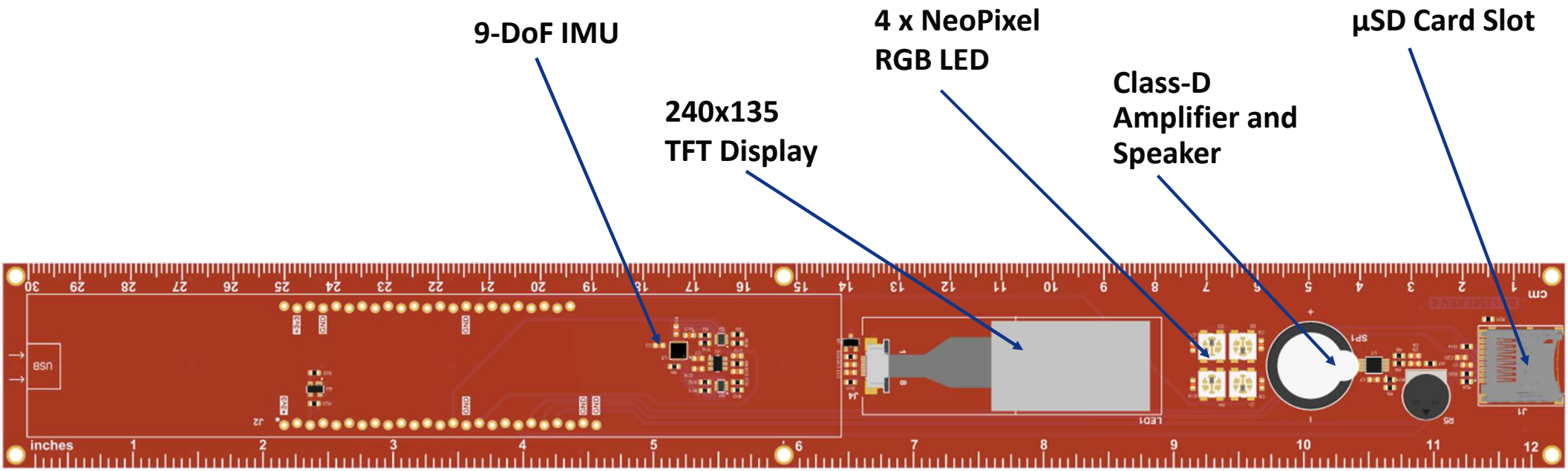
What Hardware Does The Kit Have?

Curiosity CircuitPython dev board



What Hardware Does The Kit Have?

PyKit Ruler baseboard



MICROCHIP

What Hardware Does The Kit Have?

PyKit Ruler baseboard - backside

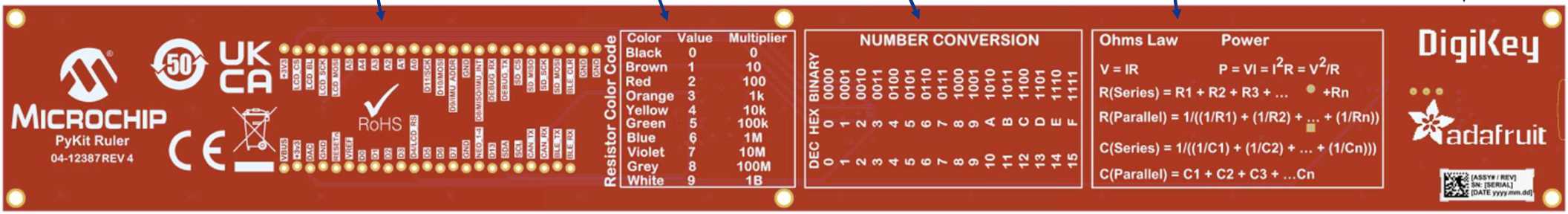
Full Pinout

Resistor Color Code

Decimal
Binary
Hex
Number Conversions

Useful Equations:
Ohm's Law
Series/Parallel
Resistors/Capacitors

Our Partners



MICROCHIP

Do We Need To Solder?

- **No Soldering Required!**

- **Both boards use a friction fit**
  - Easy to put together
  - Easy to take apart
  - Good signal integrity!

MICROCHIP

Fit header pins to Curiosity CircuitPython dev board

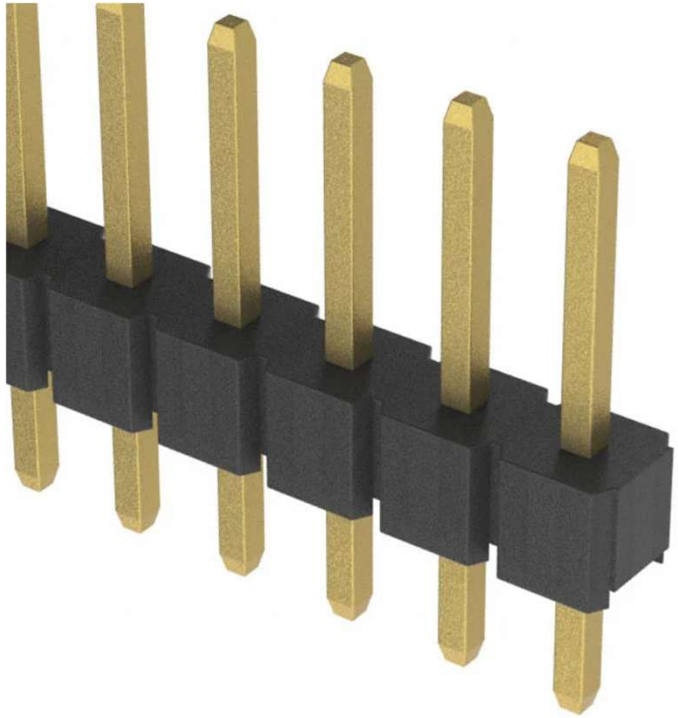
- **Group task:**

1. Fit **short side of header pins** to Curiosity CircuitPython board

   Start from one end of the header and work your way down , gently wiggling while pushing pins in

2. Connect Curiosity CircuitPython board to Ruler

   Start from USB end of Curiosity board and work your way down, gently wiggling while pushing pins in

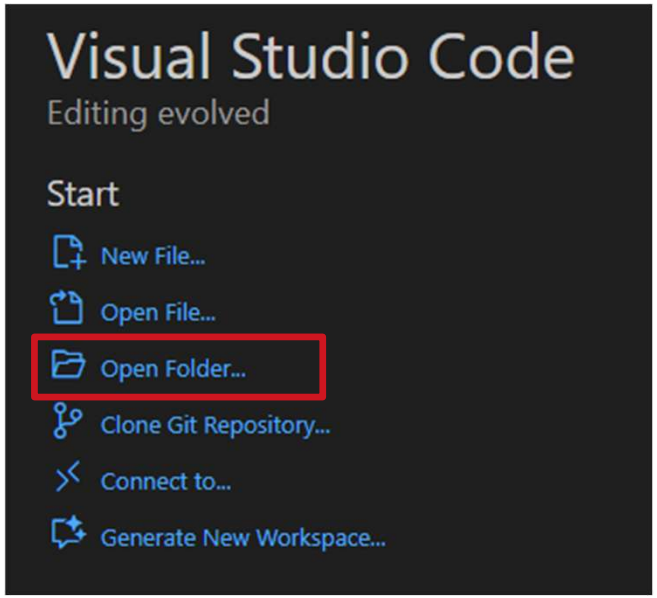


MICROCHIP

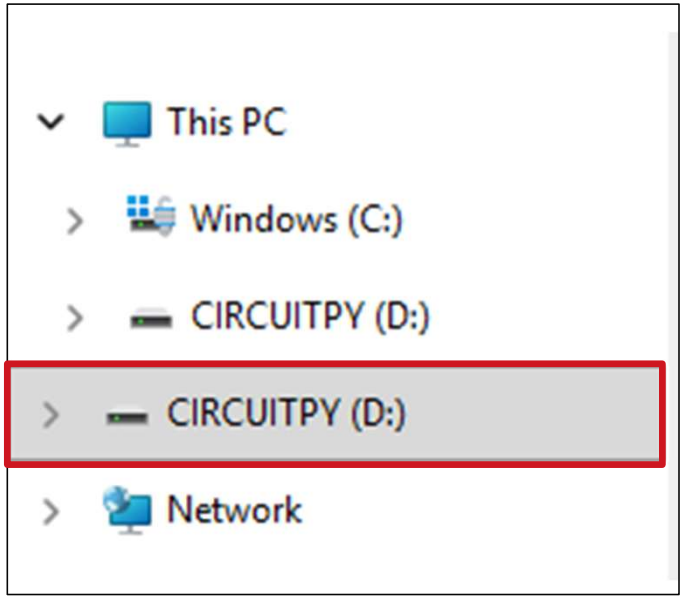
Editing CircuitPython Code in VS Code

Connect CircuitPython kit to your computer with USB cable

1. **Open VS Code:**



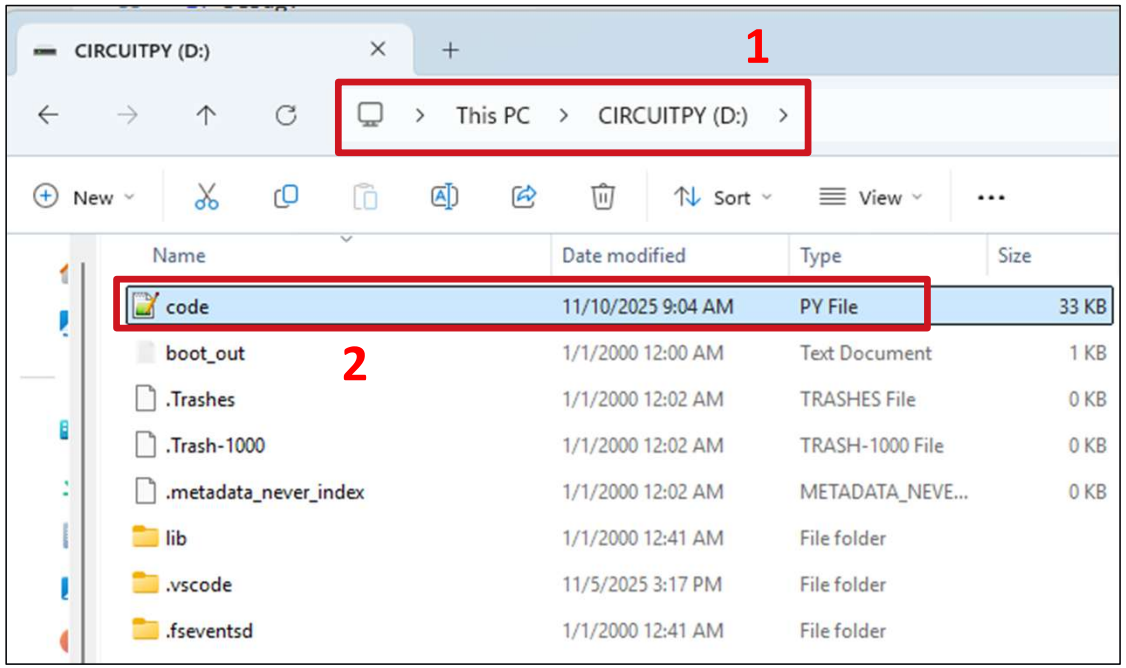
2. **Open CIRCUITPY folder:**



MICROCHIP

How to Run Your Code

Code.py file



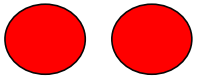
For your code to run on the board, it **must**:

1. Be in the **root directory** of D:\CIRCUITPY
2. Be named ***code.py***

MICROCHIP

NeoPixel Status Indicator

Visual Error Indication

- **NeoPixel - 1 Green Flash** 
  - Code completed successfully
- **NeoPixel - 2 Red Flashes** 
  - Code encountered an error – Check the Serial Monitor for info
- **NeoPixel - 3 Yellow Flashes** 
  - Safe Mode entered after bad crash!
  - Need to preset Reset button to exit Safe Mode

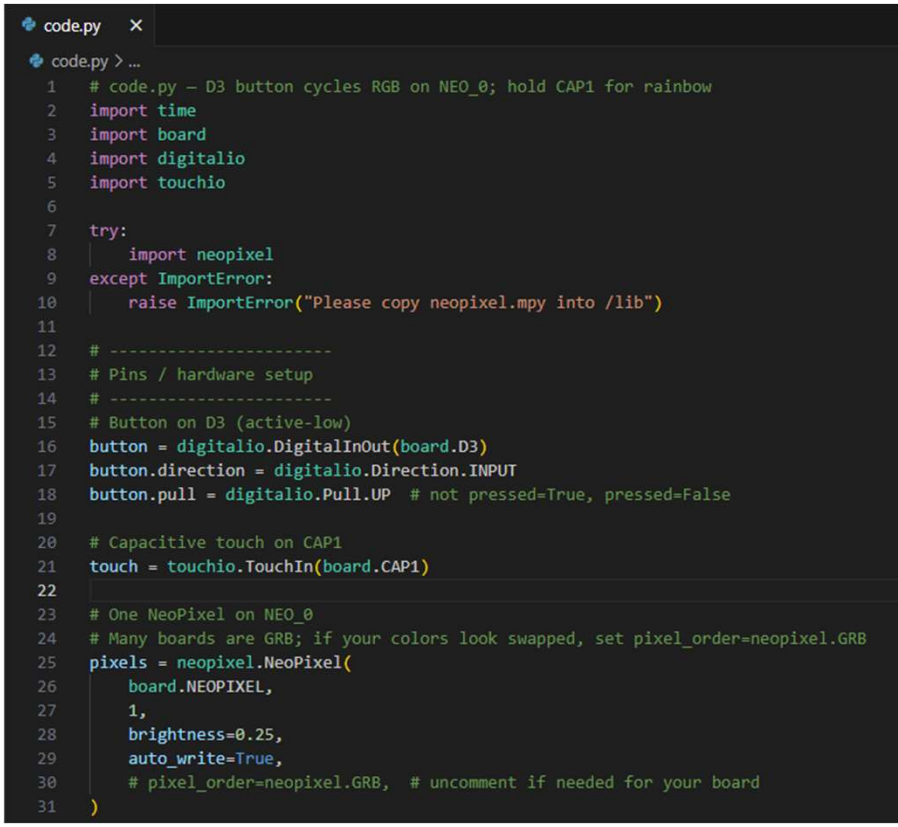
- **Note:** Do NOT confuse NeoPixel flash with BLE module Blue LED (advertising mode) flash!

MICROCHIP

Compiled Vs Interpreted Languages

What's the Difference?

- **Compiled languages:**
  - The entire source code is translated into machine code **before** execution.
  - After compiled, the executable file can be run directly without needing the original source code or the compiler
- **Interpreted languages:**
  - Source code is executed line by line, or instruction by instruction
  - Interpreter must be present on the system where the code is being run.

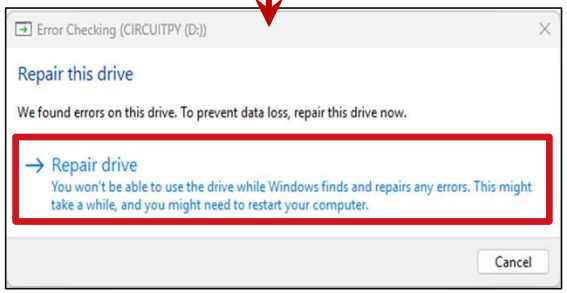


MICROCHIP

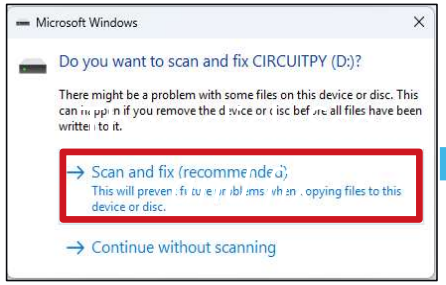
Connect The Curiosity CircuitPython

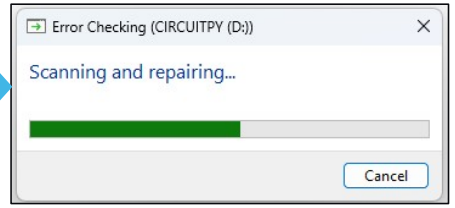
To your laptop using the included USB cable

If you see this, click *Repair Drive*

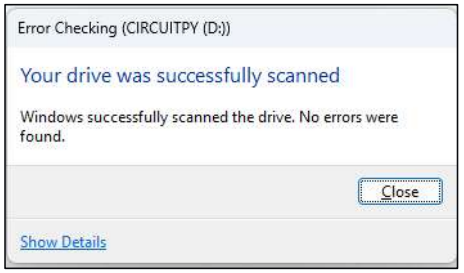


Then click *Scan and fix*





Click *Close*

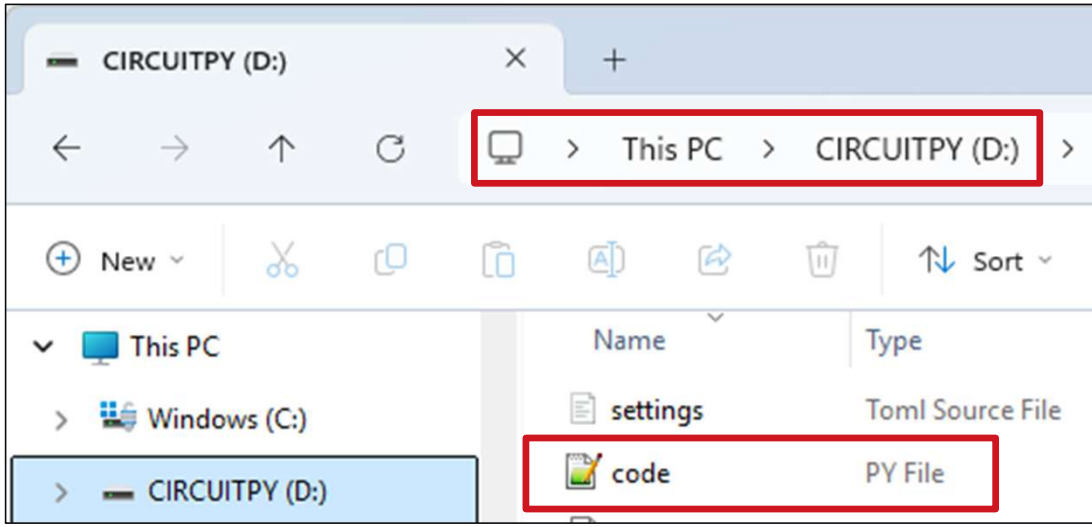


MICROCHIP

How Do I Program The Board?

Just Save The File `code.py`

- **Q: *Where* should you save your `code.py` file??**
- **A: In the root directory of:**
  - D:\CIRCUITPY
- **Note: Saving `code.py` inside a folder in D:\CIRCUITPY will NOT work! It MUST be in the ROOT DIRECTORY**
- **On Windows:**
  - Make sure you eject drive before pressing reset button or disconnecting!
  - Group task: Eject the drive



MICROCHIP

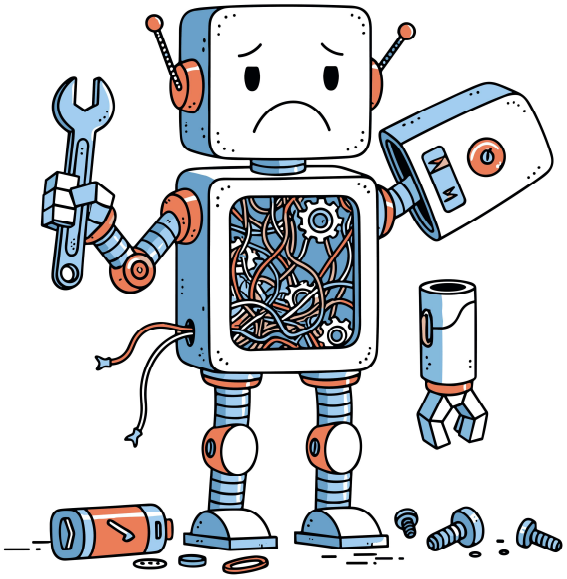
Troubleshooting!

Read Only File System Error When Saving

- **If student cannot save file and board gives Read Only File System error:**
  - First: Try Ejecting the drive, then press Reset button, then try saving file again
  - If that fails:
    - Filesystem may be corrupted!
    - Most likely source:
      - **Board has been reset/disconnected while performing write operations**
  - Before starting solution – save files from CIRCUITPY to host computer!!
- **Solution:**
  - Enter REPL
  - Type:

```
import storage
storage.erase_filesystem()
```

Then copy saved files back to D:\CIRCUITPY

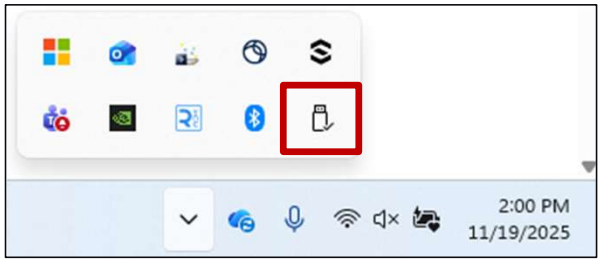


MICROCHIP

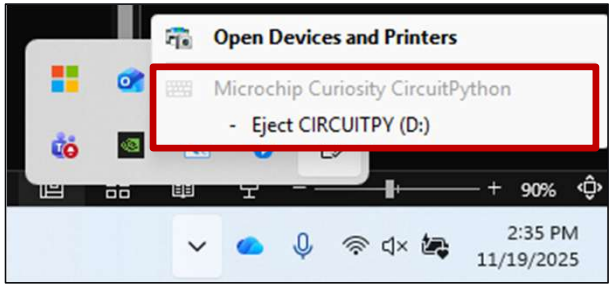
How To Eject The CIRCUITPY Drive

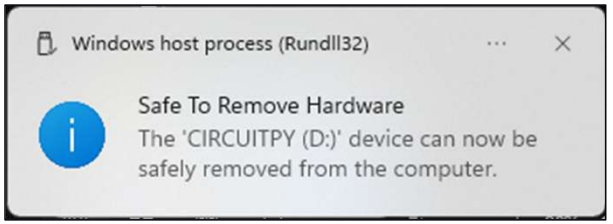
Always eject before disconnecting cable or pressing reset button

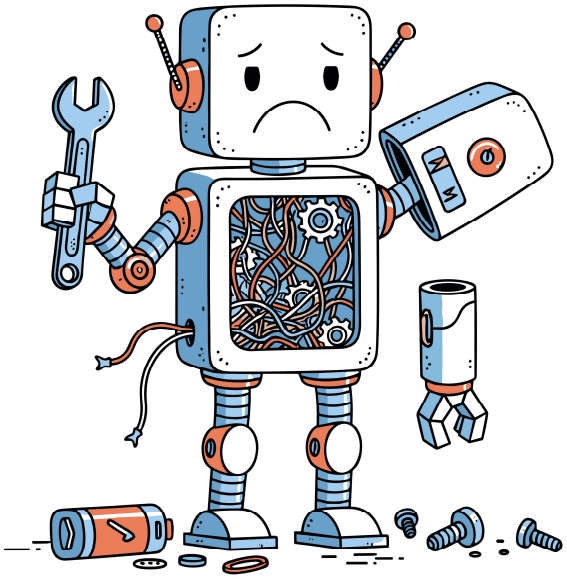
- Click ***Safely Remove Hardware and Eject Media*** button
  - Bottom right corner of Windows



  - Click ***Eject CIRCUITPY (D:)***





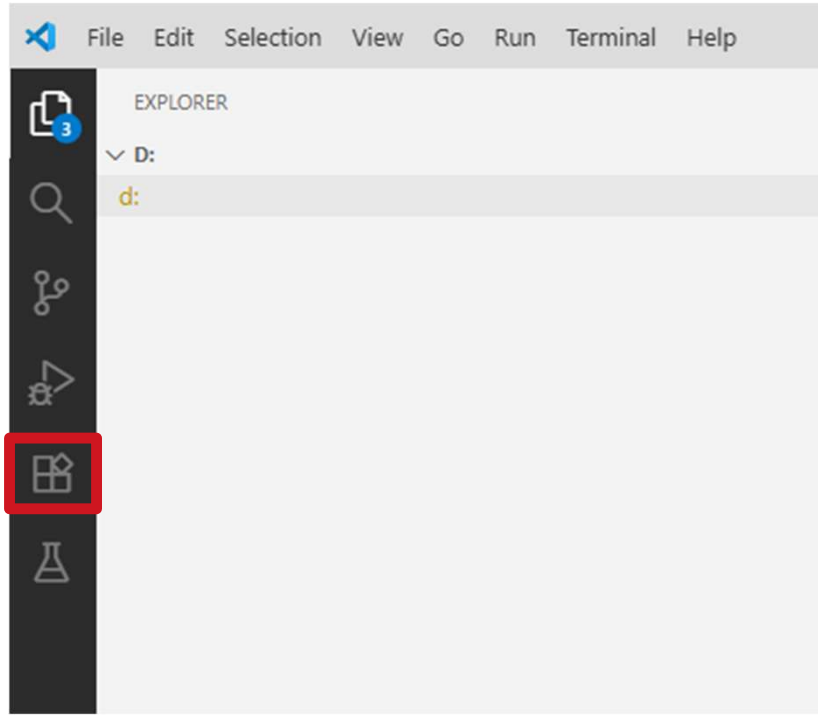


MICROCHIP

Open Serial Monitor

VS Code Extension

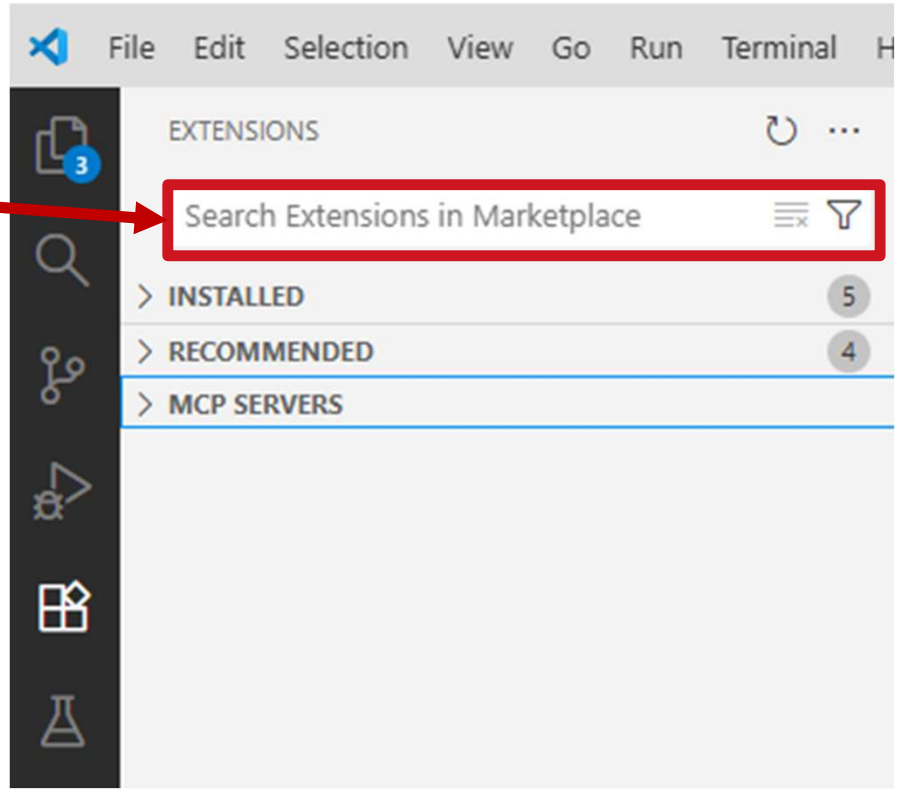
- Open *VS Code*
- Click on *Extensions* icon on left side



Search For Serial Monitor Extension

VS Code Extensions

- **Click on text field:**
  - *Search Extensions in Marketplace*

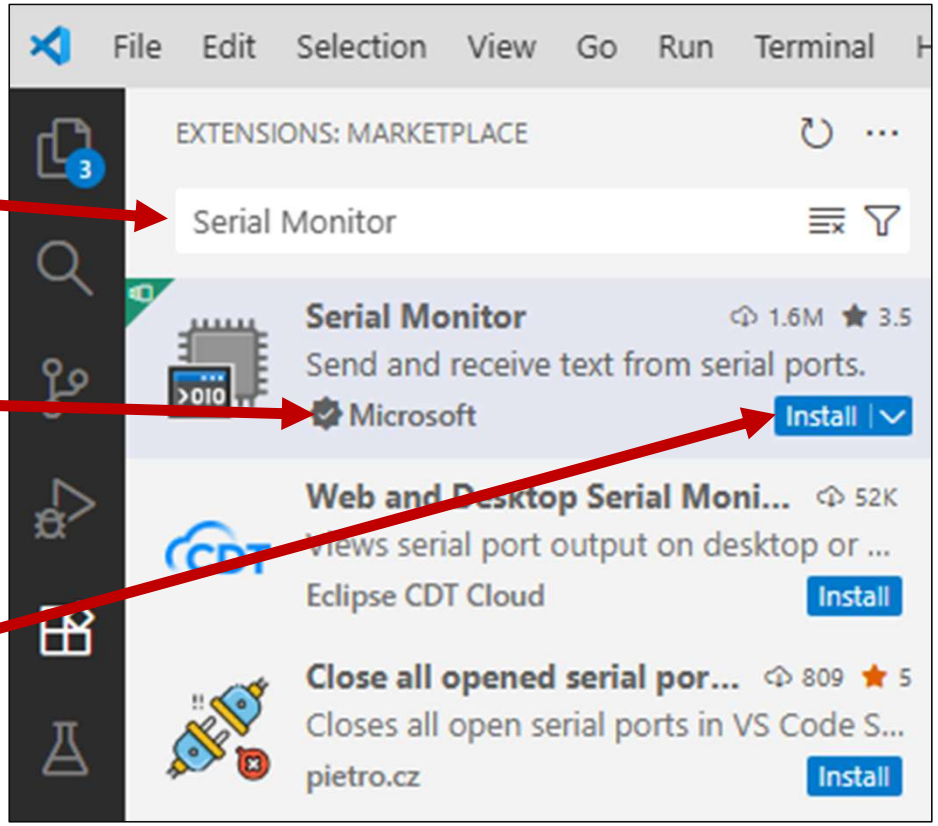


MICROCHIP

Enter “Serial Monitor” in Text Field

VS Code Extensions

- **Type in Text Field:**
  - *Serial Monitor*
- **Confirm author is *Microsoft***
- **Click *Install* button**

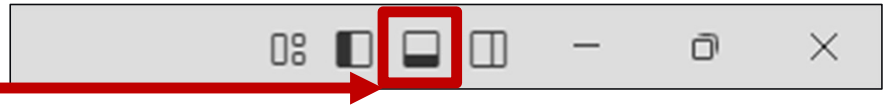


MICROCHIP

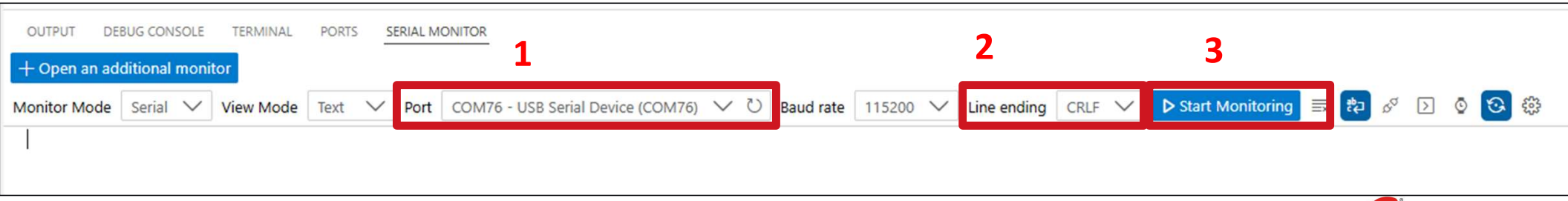
Open Serial Monitor

VS Code - Toggle Panel

- **In the top right corner**
  - Click *Toggle Panel*



1. **Select your COM port:**
   - *USB Serial Device*
2. **Line Ending:**
   - *CRLF*
3. **Click *Start Monitoring***

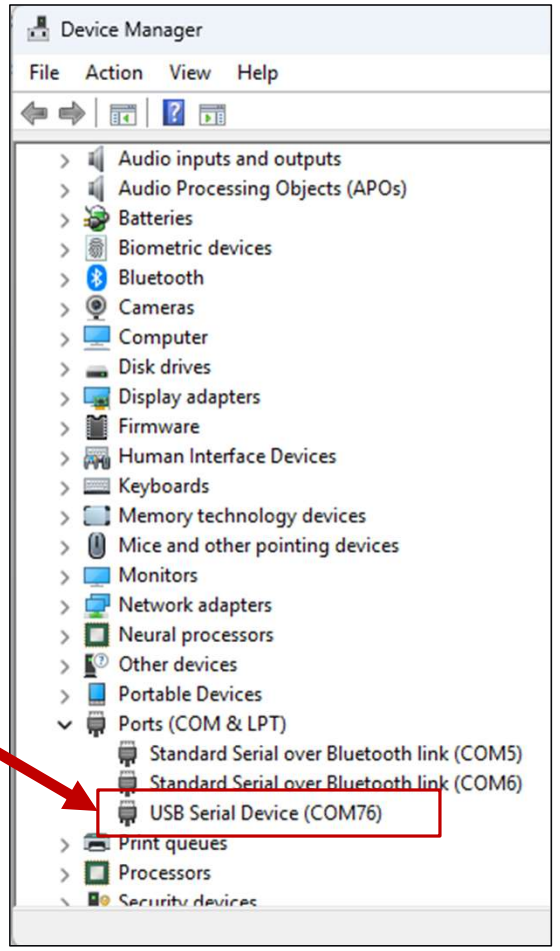


MICROCHIP

Which COM Port Do I Use?

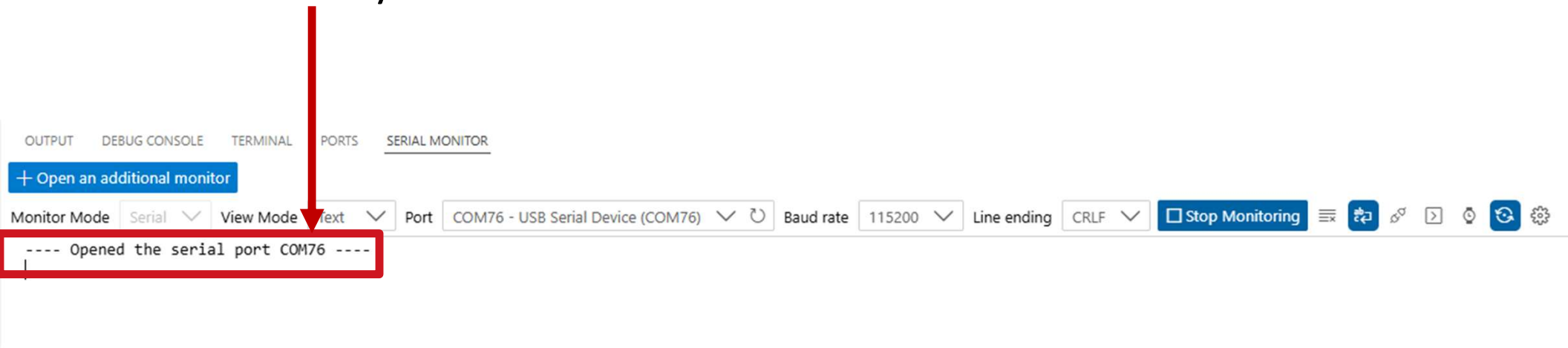
Device Manager

- **Click Windows *START* button**
  - Type *Device Manager*
- **Under: *Ports (COM & LPT)***
  - Look for *USB Serial Device*
  - This will show your COM port number – ***Your number is likely to be different!***



Serial Monitor Opened

- **You will see:**
  - *---- Opened the Serial Port COMxx ----*
  - Where: *xx* is your Serial Port Number



MICROCHIP

Can I Run Single Lines Of Code?

Like the Python Interactive Interpreter?

- **Yes!**
- It's called **REPL**
  - Read, Evaluate, Print, Loop

- **How do we enter the REPL?**
  - In Serial Terminal send *CTRL+C*

  - This stops executing code!

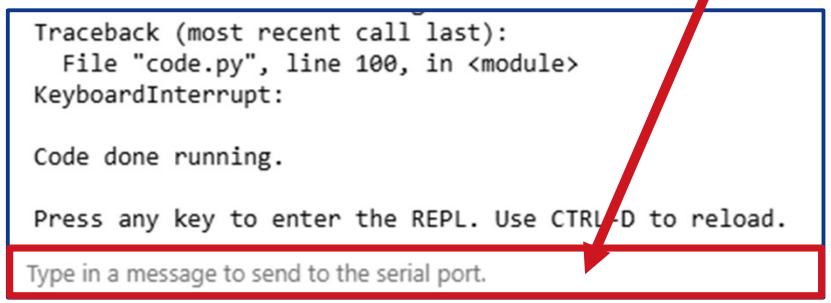
```
Traceback (most recent call last):
  File "code.py", line 100, in <module>
KeyboardInterrupt:

Code done running.

Press any key to enter the REPL. Use CTRL-D to reload.
```

Type in a message to send to the serial port.

- **Group Task: Send *CTRL+C* to your board**

Enter the REPL

- Click in the message field:



- Then press ***Enter***
  - You will then see this prompt:

```
Adafruit CircuitPython 10.1.0-alpha.0-2-g706c4b2abf on 2025-10-27; Microchip Curiosity CircuitPython with same51j20
>>>
```

  - Shows which version of CircuitPython, the board and MCU we are using

- **Group Task: Enter the REPL**

MICROCHIP

How Do I Get The Object Names?

From the REPL

- **In the REPL:**
  - Type: `import board`
  - `dir(board)`

```
---- Sent utf8 encoded message: "import board\r\n" ----
import board
>>>
---- Sent utf8 encoded message: "dir(board)\r\n" ----
dir(board)
['__class__', '__name__', 'A0', 'A1', 'A2', 'A3', 'A4', 'A5', 'BLE_CLR', 'BLE_RX', 'BLE_TX', 'CAN_RX', 'CAN_STDBY', 'CAN_TX', 'CAP1'
, 'CS', 'D0', 'D1', 'D10', 'D11', 'D13', 'D2', 'D3', 'D4', 'D5', 'D6', 'D7', 'D8', 'D9', 'DAC', 'DEBUG_RX', 'DEBUG_TX', 'I2C', 'IMU_
ADDR', 'IMU_INT', 'LCD_BL', 'LCD_CS', 'LCD_MOSI', 'LCD_SCK', 'LCD_SPI', 'LED', 'MISO', 'MOSI', 'NEOPIXEL', 'SCK', 'SCL', 'SDA', 'SD_
CS', 'SD_MISO', 'SD_MOSI', 'SD_SCK', 'SD_SPI', 'SPI', 'UART', 'VREF', '__dict__', 'board_id']
>>>
```

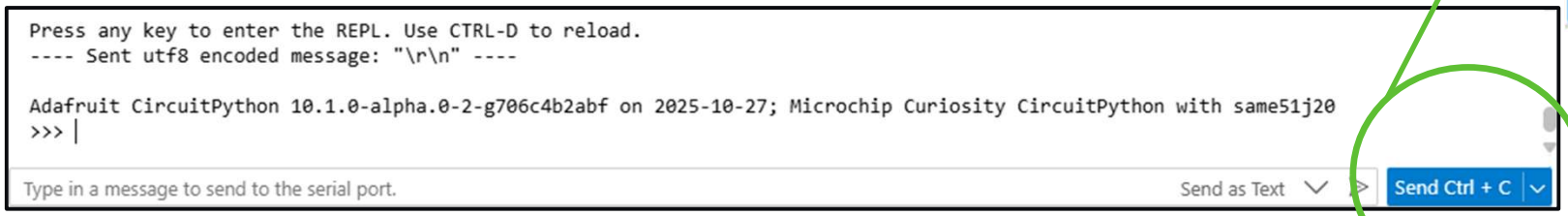
- **Notice that one of the objects is named: `LED`**
- **Group Task: Get the object names of your board**

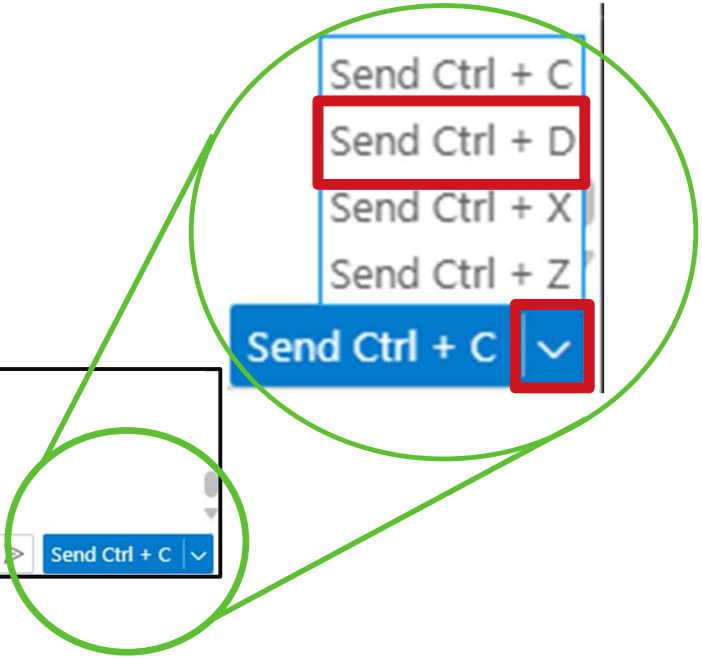
MICROCHIP

How Do I Exit The REPL?

Send CTRL-D to reload your program

- **Click on dropdown next to “Send CTRL + C”**
- **Select “Send CTRL + D”**





- **In the Serial Terminal you will see**
- **“Code done running.”**

```
soft reboot

Auto-reload is on. Simply save files over USB to run them or enter REPL to disable.
code.py output:


Hello World!


Code done running.

Press any key to enter the REPL. Use CTRL-D to reload.
```

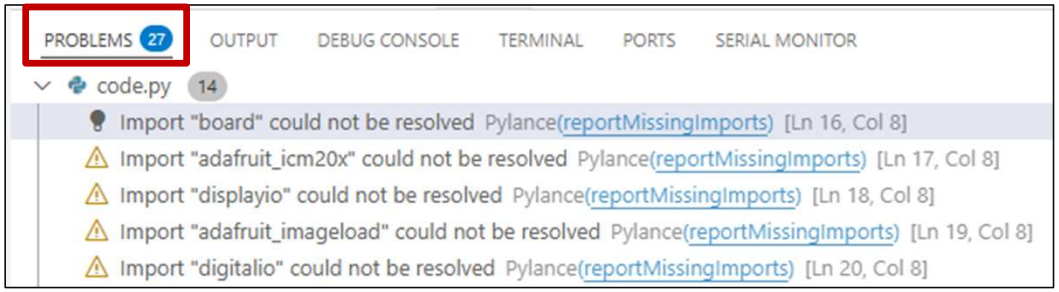
- **This means the code has now completed execution!**

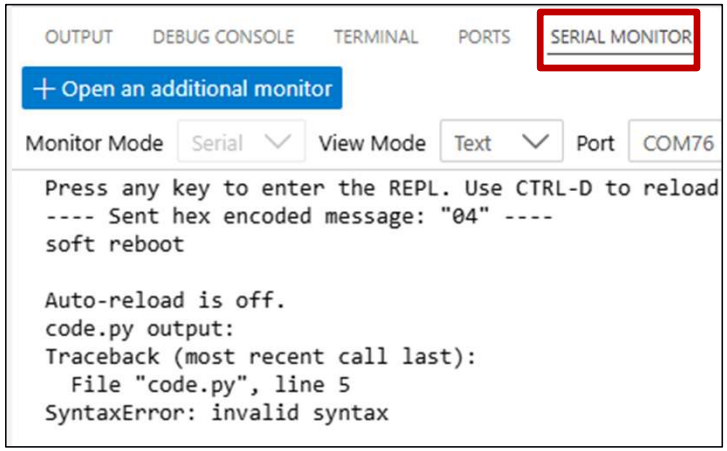
MICROCHIP

IMPORTANT NOTE ABOUT ERRORS!

Use “Serial Monitor” – Not “Problems”

- The extension ***CircuitPython v2*** does **NOT YET** support this board.
- ***Problems*** tab shows “problems” due to this extension not yet supporting this board
- Use ***Serial Monitor*** tab to view code errors





MICROCHIP

For Students Familiar With OOP

Practice Exercise

- **Enter the following text into code.py and save:**

code.py

```
x = 10  # x is an integer object
print("\nx is of type: ", type(x))  # Output: <class 'int'>
print("Memory address of x: ", id(x))    # Output: A memory address (will vary)

s = "hello"  # s is a string object
print("Print string 's' in upper case: ", s.upper()) # Output: HELLO (calling a method on the string object)

def my_function():
    pass

print("my_function is of type: ", type(my_function)) # Output: <class 'function'> (functions are also objects)

```


MICROCHIP

For Students Familiar With OOP

Practice Exercise Output

```
code.py output:

x is of type:  <class 'int'>
Memory address of x:  21
Print string 's' in upper case:  HELLO
my_function is of type:  <class 'function'>

Code done running.

Press any key to enter the REPL. Use CTRL-D to reload.
```

Let’s Get Started – 1. Blink onboard LED

The API is your friend!

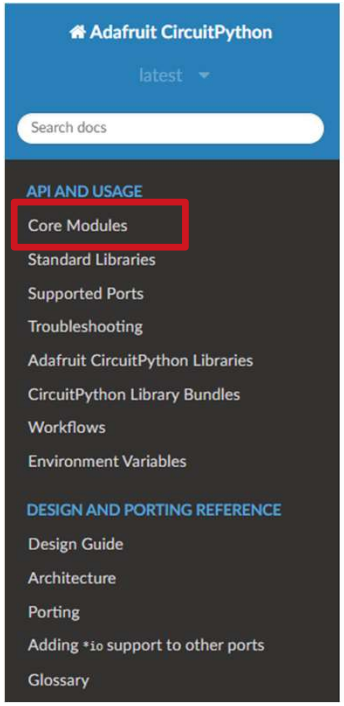
Step 1:

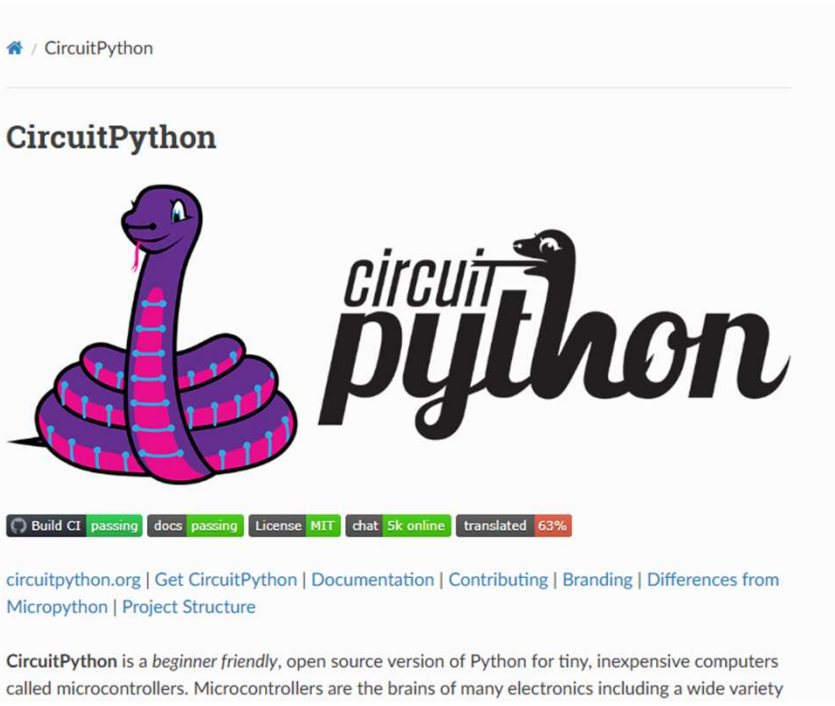
Find the documentation!

Bookmark this:

docs.circuitpython.org

Click on “Core Modules”





MICROCHIP

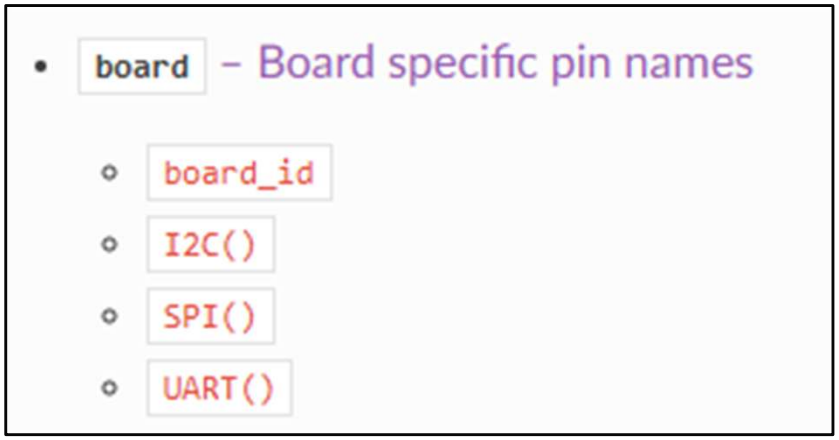
Using The Pins In A Project

Getting Started

Step 2:

EVERY embedded project with **ANY** CircuitPython board requires:

```
import board
```



board provides pin objects, bus objects (e.g. LED, D13, SPI, I2C, UART)

MICROCHIP

Using Digital Input/Output

To Blink LED we need *digitalio* library

- ***digitalio*** library is required to use GPIO pins
- **For examples:** docs.circuitpython.org
  - *Documentation -> Core Modules -> digitalio*
  - Copy this code from digitalio docs
  - Save as *code.py* in CIRCUITPY drive **root directory**

Here is blinky:

```
import time
import digitalio
import board

led = digitalio.DigitalInOut(board.LED)
led.direction = digitalio.Direction.OUTPUT
while True:
    led.value = True
    time.sleep(0.1)
    led.value = False
    time.sleep(0.1)
```


MICROCHIP

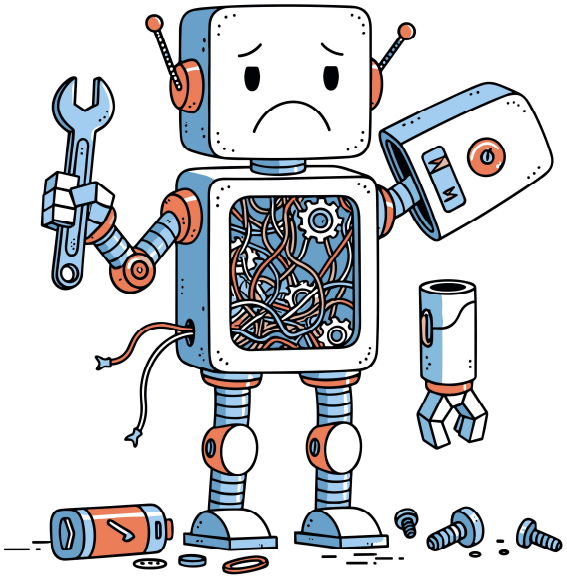
Troubleshooting!

Read Only File System Error When Saving

- **If student cannot save file and board gives Read Only File System error:**
  - First: Try Ejecting the drive, then press Reset button, then try saving file again
  - If that fails:
    - Filesystem may be corrupted!
    - Most likely source:
      - **Board has been reset/disconnected while performing write operations**
  - Before starting solution – save files from CIRCUITPY to host computer!!
- **Solution:**
  - Enter REPL
  - Type:

```
import storage
storage.erase_filesystem()
```

Then copy saved files back to D:\CIRCUITPY

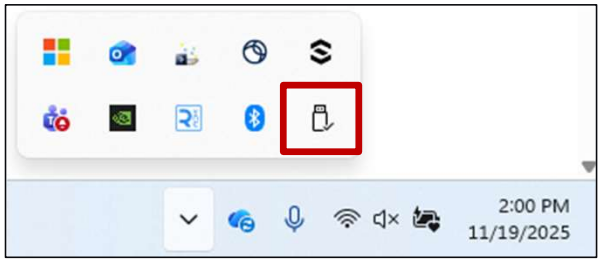


MICROCHIP

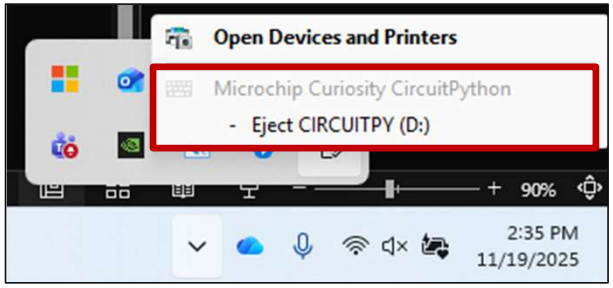
How To Eject The CIRCUITPY Drive

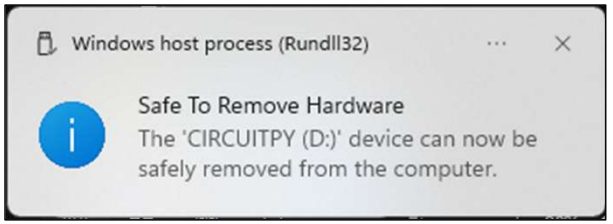
Always eject before disconnecting cable or pressing reset button

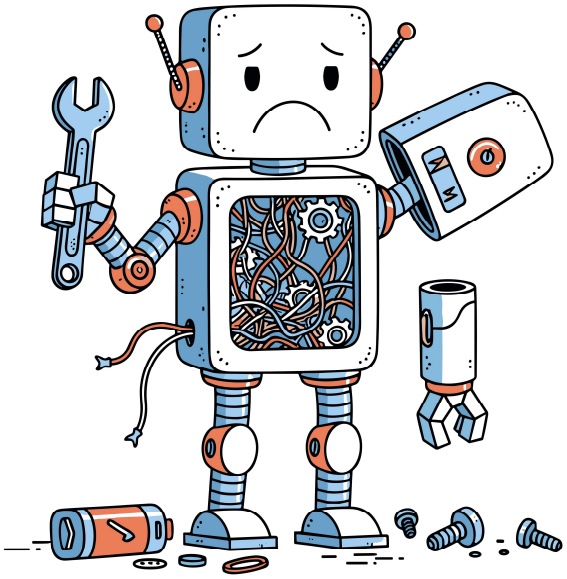
- Click ***Safely Remove Hardware and Eject Media*** button
  - Bottom right corner of Windows



  - Click ***Eject CIRCUITPY (D:)***



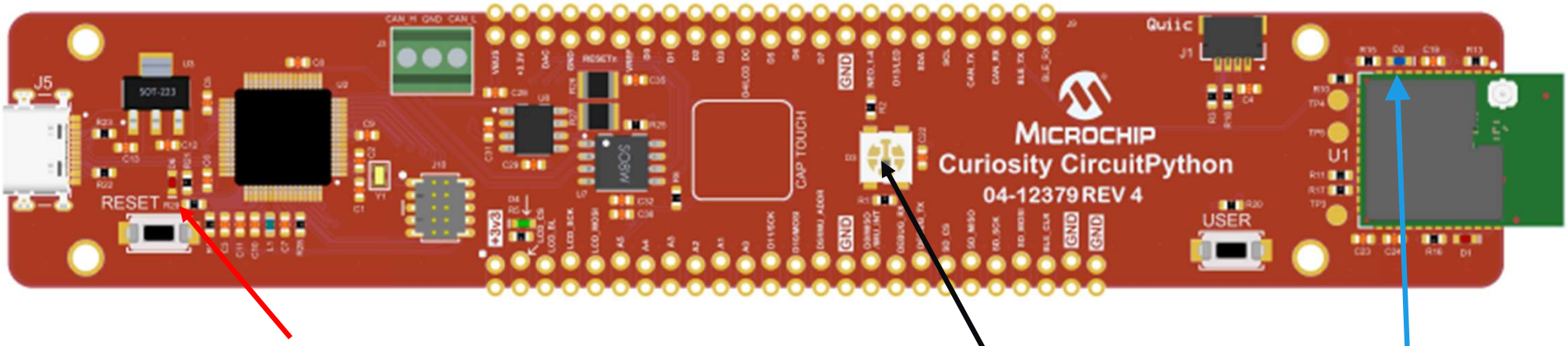




MICROCHIP

Where Is The LED?

I can’t find it!



- There is a **Red LED** near the USB connector
- Do not confuse the LED and NeoPixel. The NeoPixel is next to the Cap Touch pad
- The Bluetooth LE module has a **Blue LED** that blinks periodically (advertising mode)

MICROCHIP

Let's Get Started – 2. Read User Button

Turn on LED when button pressed

code.py

Read_Button > code.py > ...

```
1   import board
2   import digitalio
3
4   # create user button object
5   user_button = digitalio.DigitalInOut(board.D3)
6   # set up user button as an input
7   user_button.direction = digitalio.Direction.INPUT
8
9   # create led object
10  led = digitalio.DigitalInOut(board.LED)
11  # set up led as an output
12  led.direction = digitalio.Direction.OUTPUT
13
14
15  while True:     # loop forever
16      # get the value of the button - returns True or False
17      button = user_button.value
18
19      # if button is True turn on led
20      if (button):
21          led.value = True
22      # otherwise turn off led
23      else:
24          led.value = False
25
```

- **User button is on pin D3**
  - Refer to the **Schematics** folder in D:\CIRCUITPY
  - Schematics are PDF files
    - Open in a PDF viewer or browser

Enter this code:

- Save as ***code.py*** in **root** directory of D:\CIRCUITPY

- **What happens?**

MICROCHIP

Let's Get Started – 2. Read User Button

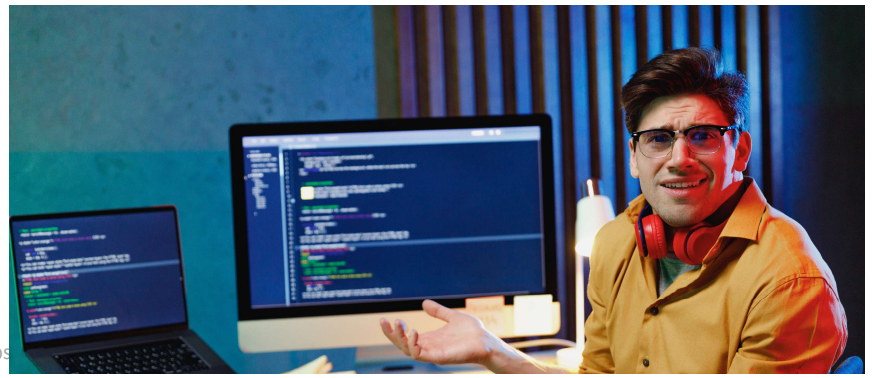
Turn on LED when button pressed

code.py

Read_Button > code.py > ...

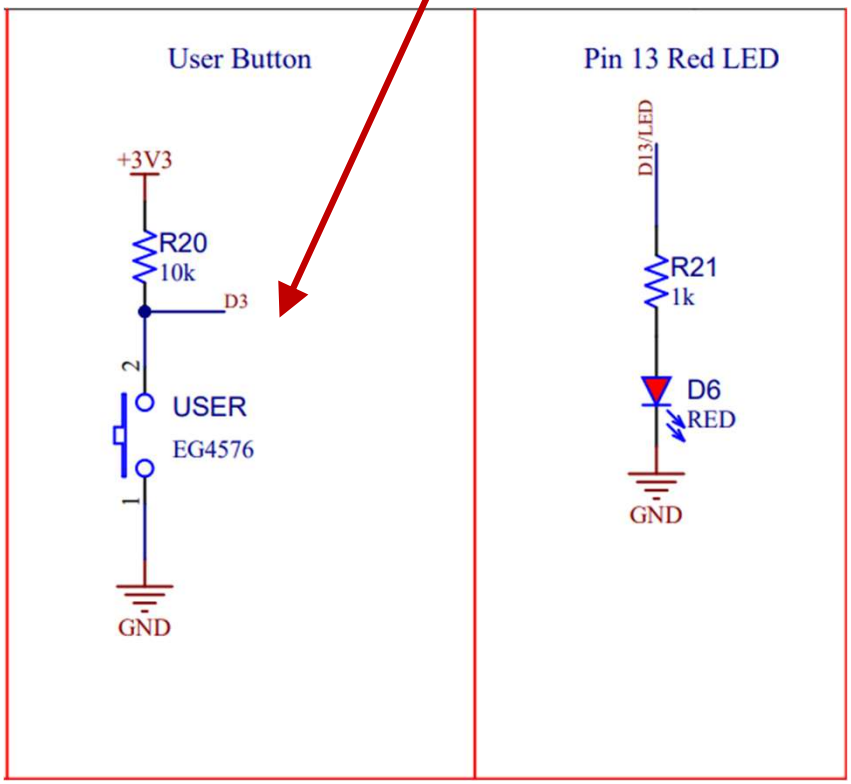
```
1   import board
2   import digitalio
3
4   # create user button object
5   user_button = digitalio.DigitalInOut(board.D3)
6   # set up user button as an input
7   user_button.direction = digitalio.Direction.INPUT
8
9   # create led object
10  led = digitalio.DigitalInOut(board.LED)
11  # set up led as an output
12  led.direction = digitalio.Direction.OUTPUT
13
14
15  while True:     # loop forever
16      # get the value of the button - returns True or False
17      button = user_button.value
18
19      # if button is True turn on led
20      if (button):
21          led.value = True
22      # otherwise turn off led
23      else:
24          led.value = False
25
```

- **User button is on pin D3**
- **LED is:**
  - OFF when button is pressed
  - ON at all other times.
  - Any ideas why?



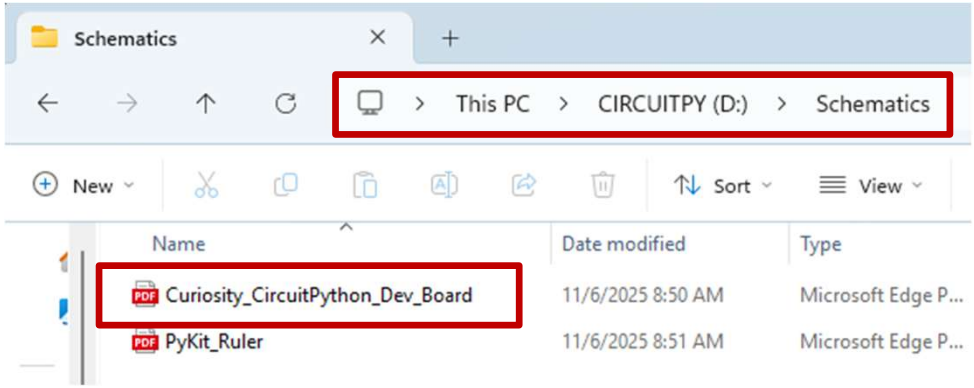
Let’s look at the Curiosity CircuitPython schematics!

User Button (D3) is Active-LOW!



- **Schematics are saved on the board!**
- D:\CIRCUITPY\Schematics

Open schematic file in your **browser**:



Let's Get Started – 2. Read User Button

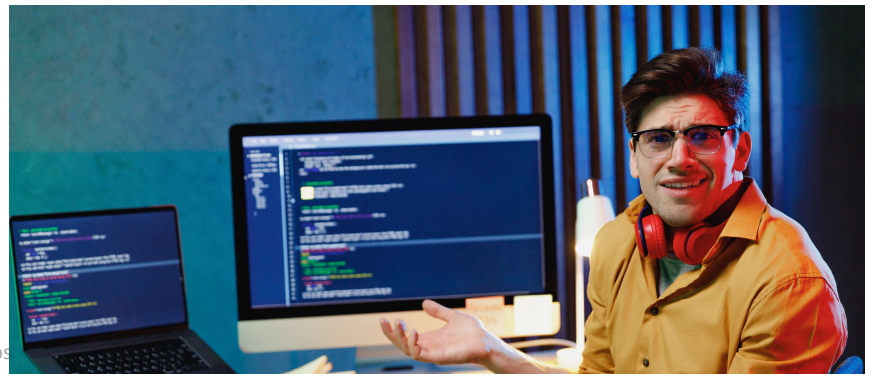
Turn on LED when button pressed

code.py

Read_Button > code.py > ...

```
1   import board
2   import digitalio
3
4   # create user button object
5   user_button = digitalio.DigitalInOut(board.D3)
6   # set up user button as an input
7   user_button.direction = digitalio.Direction.INPUT
8
9   # create led object
10  led = digitalio.DigitalInOut(board.LED)
11  # set up led as an output
12  led.direction = digitalio.Direction.OUTPUT
13
14
15  while True:     # loop forever
16      # get the value of the button - returns True or False
17      button = user_button.value
18
19      # if button is True turn on led
20      if (button):
21          led.value = True
22      # otherwise turn off led
23      else:
24          led.value = False
25
```

- **Group Task: Figure out how to make LED turn ON when button pressed, and OFF when button released**
- What needs to change in this code?



Let’s Get Started – 2. Read User Button - Solution

Invert the Logic!

```
1   import board
2   import digitalio
3
4   # create user button object
5   user_button = digitalio.DigitalInOut(board.D3)
6   # set up user button as an input
7   user_button.direction = digitalio.Direction.INPUT
8
9   # create led object
10  led = digitalio.DigitalInOut(board.LED)
11  # set up led as an output
12  led.direction = digitalio.Direction.OUTPUT
13
14
15  while True:     # loop forever
16      # get the value of the button - returns True or False
17      button = user_button.value
18
19      # if button is False (LOW) turn on led
20      if (not button):
21          led.value = True
22      # otherwise turn off led
23      else:
24          led.value = False
25
```

MICROCHIP

Let's Get Started – 2. Toggle LED on User Button Press

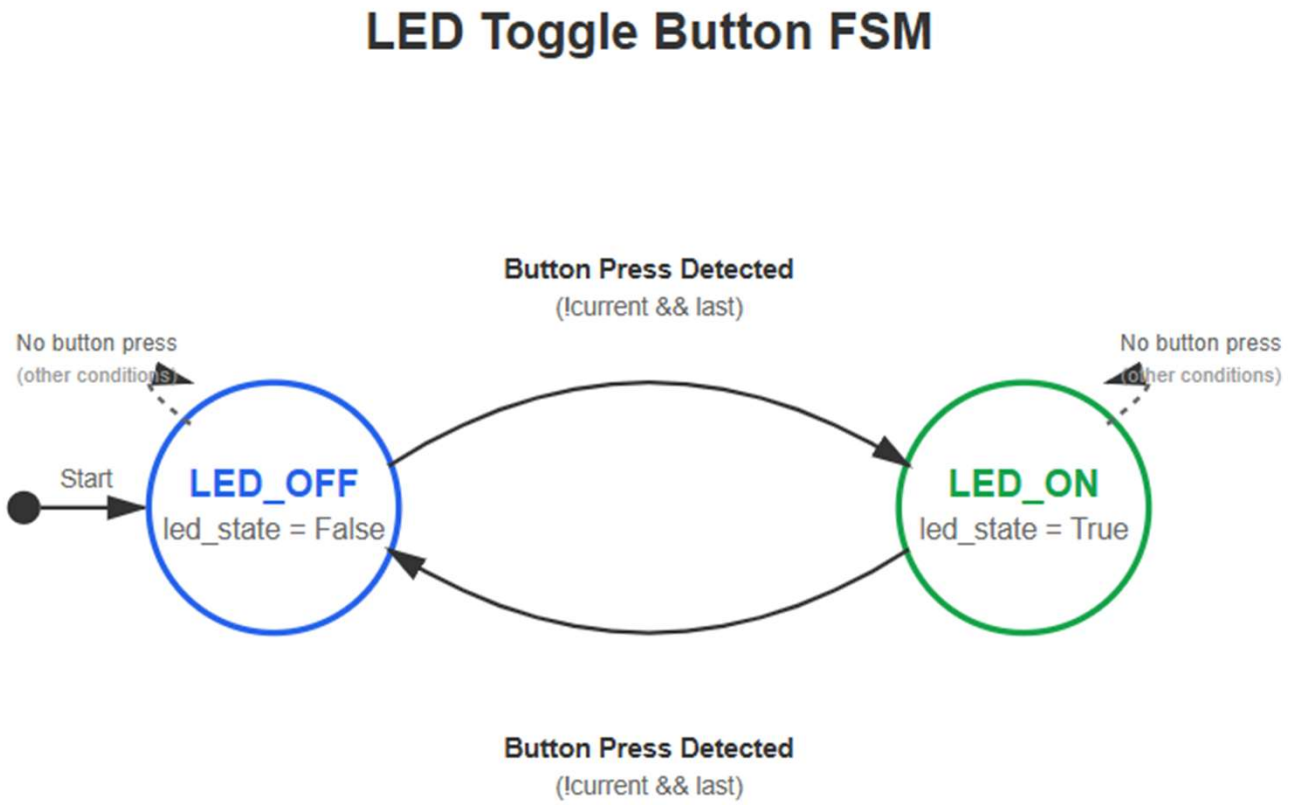
More Complex Example – State Machines

- **User button is on pin D3**
- **Enter this code**
- **Students to write the logic to toggle LED on each button press here**

Toggle_button > code.py > ...

```
1   import board
2   import digitalio
3
4   # configure User Button as digital input
5   # NOTE: button is Active LOW
6   user_button = digitalio.DigitalInOut(board.D3)
7   user_button.direction = digitalio.Direction.INPUT
8
9   # configure onboard LED as output
10  led = digitalio.DigitalInOut(board.LED)
11  led.direction = digitalio.Direction.OUTPUT
12
13  # set initial states for the button
14  current_button_state = False
15  last_button_state = False
16
17  # set initial state for LED - start with LED off
18  led_state = False
19
20  while True:
21      # read the button and save to current state
22      current_button_state = user_button.value
23
24      # Students to write the logic to toggle LED on button press
25
26
```

FSM Diagram



Legend:

- Button Press Detected: Falling edge (button goes from unpressed to pressed)
- Condition: !current = button pressed (LOW), last = button was unpressed (HIGH)

MICROCHIP

Toggle LED: Solution

```
1   import board
2   import digitalio
3
4   # configure User Button as digital input
5   # NOTE: button is Active LOW
6   user_button = digitalio.DigitalInOut(board.D3)
7   user_button.direction = digitalio.Direction.INPUT
8
9   # configure onboard LED as output
10  led = digitalio.DigitalInOut(board.LED)
11  led.direction = digitalio.Direction.OUTPUT
12
13  # set initial states for the button
14  current_button_state = False
15  last_button_state = False
16
17  # set initial state for LED - start with LED off
18  led_state = False
19
20  while True:
21      # read the button and save to current state
22      current_button_state = user_button.value
23
24      # if the button has been pressed, the current state will be False AND
25      # the last state will be True
26      if (not current_button_state and last_button_state):
27          print ("Button pressed!")       # write to serial monitor as a sanity check
28          led_state = not led_state       # set the LED state to be the opposite of what it was
29          led.value = led_state           # write the state to the LED
30
31      last_button_state = current_button_state    # save the button state for the next time through the loop
32
```

Let's Get Started – 3. Control NeoPixel

Five NeoPixels Total

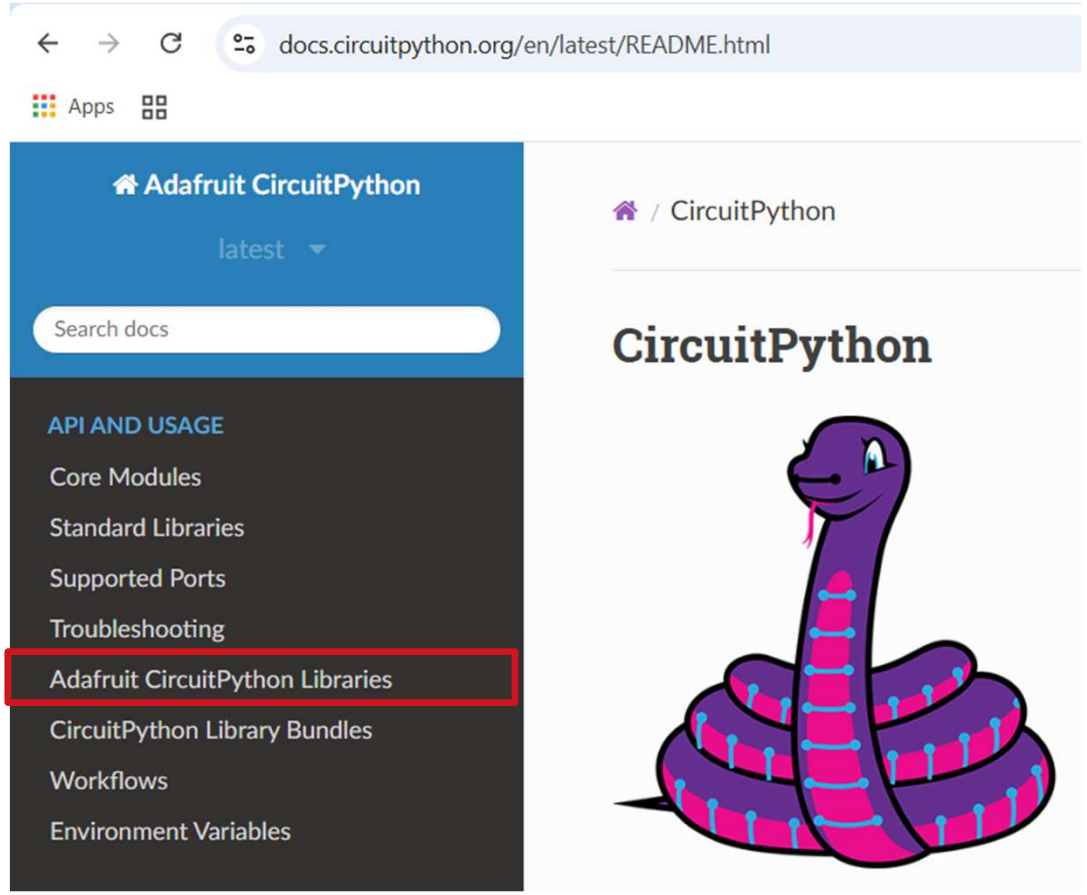
Step 1:

Find the documentation!

Remember this?

docs.circuitpython.org

Click on "Adafruit CircuitPython Libraries"



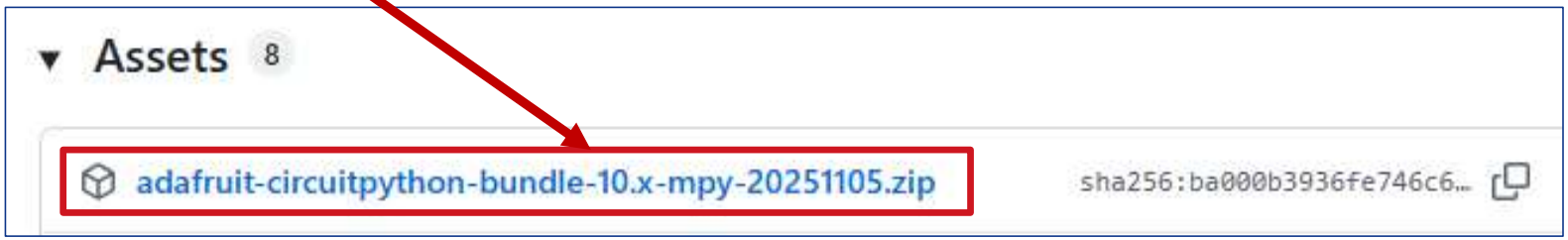
MICROCHIP

Get the Adafruit CircuitPython Library Bundle

- **Look for:**

The Adafruit bundles are available on GitHub: <https://github.com/adafruit/Adafruit_CircuitPython_Bundle/releases>. The Community bundles are available at: <https://github.com/adafruit/CircuitPython_Community_Bundle/releases>.

- **Click on link:**
  - *Adafruit_CircuitPython_Bundle/releases*
- **Click on:**



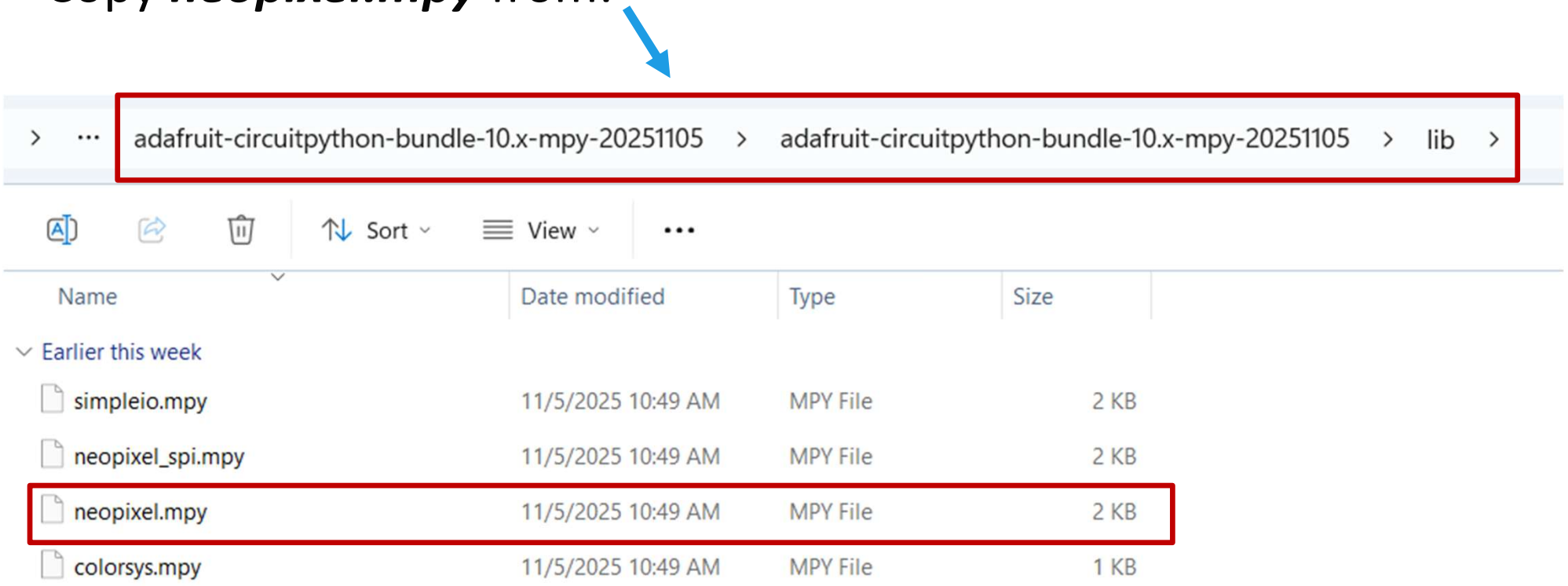
IMPORTANT: Students should become familiar with downloading the Adafruit CircuitPython Bundle, for any future projects. The bundle contains MANY very useful libraries that make life much easier!

MICROCHIP

Which neopixel.mpy file? I found 2 of them!

Look in the *lib* folder

- Copy ***neopixel.mpy*** from:



MICROCHIP

Extract the Bundle to Host Computer

Why Not To CIRCUITPY?

- **Bundle is too big to all fit on CIRCUITPY!**
- **Only copy needed libraries to:**
  - CIRCUITPY/lib/

MICROCHIP

Copy NeoPixel library to Dev Board

Where Do You Put It?

NOTE: Use Windows Explorer to copy files. Copying files using VS Code can cause problems!

Where to put ***neopixel.mpy*** file?

ALL library files go in **D:\CIRCUITPY\lib**

MICROCHIP

Now Get Project Code:

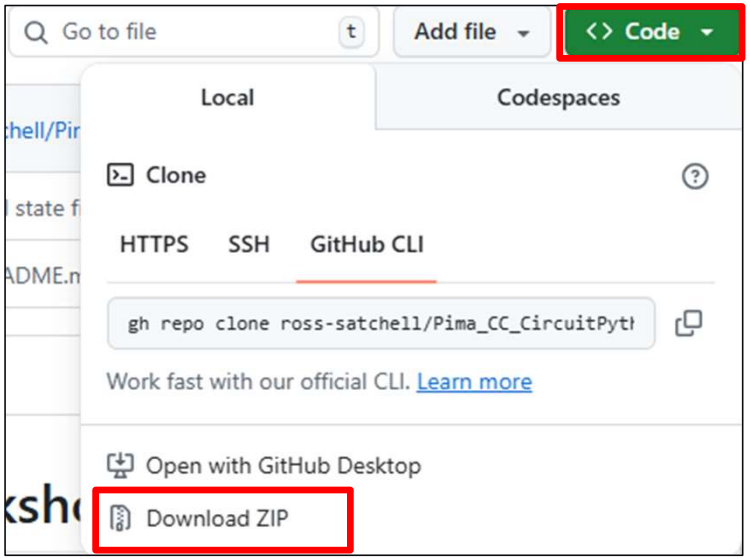
GitHub Repo

- **Direct your browser to:**

https://github.com/ross-satchell/Pima_CC_CircuitPython_Workshop

To download repo:

1. Click green ***Code*** button
2. Click ***Download ZIP***
3. Extract ZIP
4. Copy contents to D:\CIRCUITPY
   1. If asked to replace ***lib*** folder -> **YES**



MICROCHIP

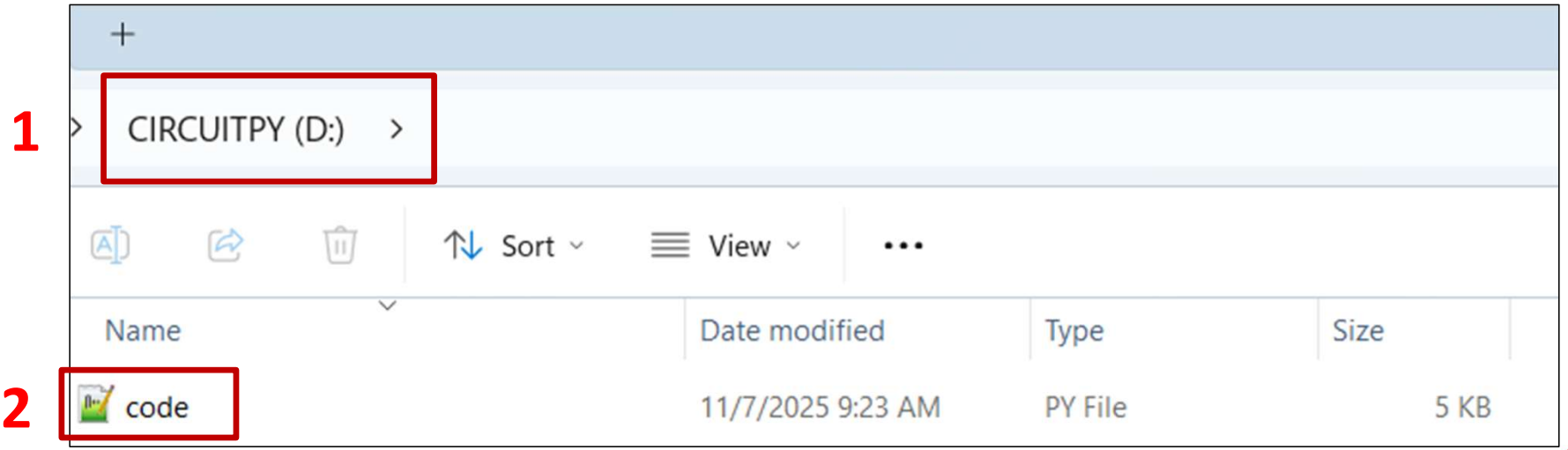
To Use GitHub Repo Code

Using GitHub Repo code

- **Recall:**

  To run a CircuitPython program, it **must** be:

  1. In the **root** directory of **D:\CIRCUITPY** and
  2. Named ***code.py***

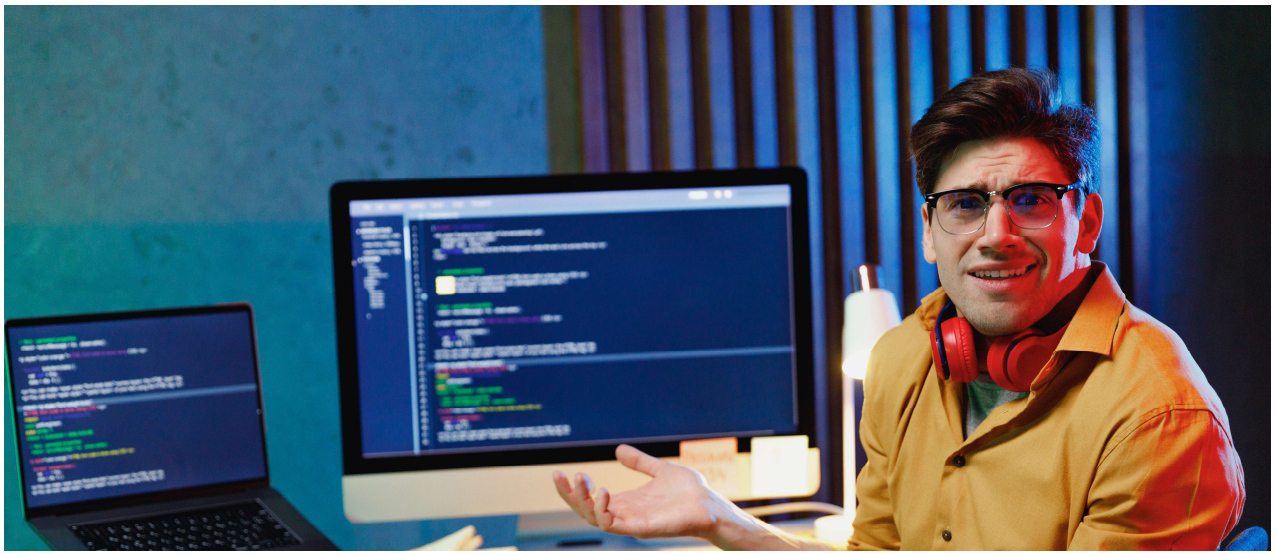


MICROCHIP

Open: 0-4_NeoPixel_Primary_Color_Mode

Copy code.py to CIRCUITPY

- **Run the code:**
  - What happens?
  - Why?
- **How do we fix it?**



MICROCHIP

Modulus Operator To The Rescue!

Which line do we change?

- **The error message in Serial Monitor shows the error is on line 39:**

```
Traceback (most recent call last):
  File "code.py", line 39, in <module>
IndexError: index out of range

Code done running.
```

```
34
35      # Solid color mode: detect button press to advance color
36      if (not last_button_state) and (current_button_state):
37          # Is there a problem with this?
38          color_i = (color_i + 1)
39          pixels[0] = colors[color_i]
40          print(f"Changed to {colors_text[color_i]}")
41
42      last_button_state = current_button_state
43
```

MICROCHIP

Modulus Operator To The Rescue!

What does this do? Hint: Remember Integer Math?

Current `color_i`	`color_i + 1`	`% 3` (remainder)	New `color_i`	Color
0	1	**1**	1	Green
1	2	**2**	2	Red
2	3	**0**	0	Blue ← wraps!
0	1	**1**	1	Green

```
color_i = (color_i + 1) % len(colors)
          └──────┬─────┘   └────┬────┘
          add 1 first     divide by 3, keep remainder
```

MICROCHIP

Modulus Operator To The Rescue!

Solution: Add the Modulus operator

```
35      # Solid color mode: detect button press to advance color
36      if (not last_button_state) and (current_button_state):
37          # Is there a problem with this?
38          color_i = (color_i + 1) % len(colors)
39          pixels[0] = colors[color_i]
40          print(f"Changed to {colors_text[color_i]}")
```

MICROCHIP

Open: 1-1_ST7789_Stationary_Meatball

Copy *code.py* and */lib/* to CIRCUITPY

- **If asked to merge *lib* folder**
  -> YES

- **Run the code:**
  - What happens?

- **Let's walk through the code**

MICROCHIP

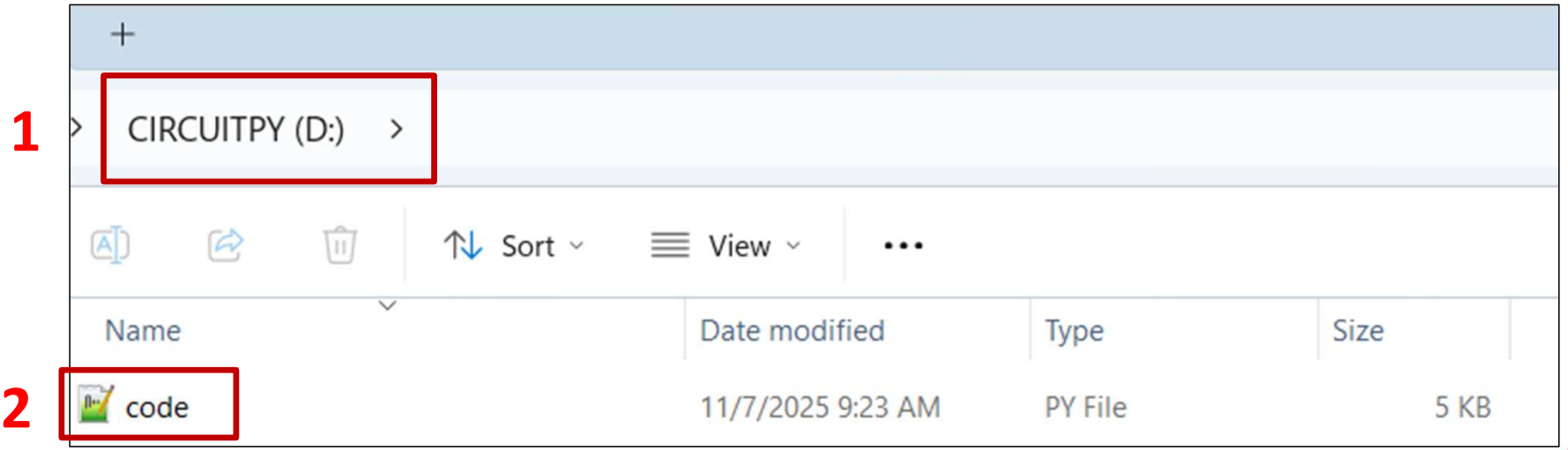
To Use GitHub Repo Code

Using GitHub Repo code

- **Recall:**

  To run a CircuitPython program, it must be:

  1. In the **root** directory of **D:\CIRCUITPY**
  2. Named ***code.py*** and



MICROCHIP

1-1_ST7789_Stationary_Meatball

Import libraries

code.py > ...

```
1   import time
2   import board
3   import displayio
4   import digitalio
5   import microcontroller
6   from fourwire import FourWire
7   from adafruit_st7789 import ST7789
```

ST7789 is a driver for many different TFT displays

MICROCHIP

1-1_ST7789_Stationary_Meatball

Debug, Set Up Backlight, Release Resources

```
9   # Change this to False to hide debug print statements
10  Debug = True
11
12  if Debug:
13      print("Create pin called 'backlight' for LCD backlight on PA06")
14  backlight = digitalio.DigitalInOut(microcontroller.pin.PA06)
15  backlight.direction = digitalio.Direction.OUTPUT
16
17  if Debug:
18      print("Turn TFT Backlight On")
19  backlight.value = False # Backlight control is Active LOW
20
21  # Release any resources that may have been previously in use for the displays
22  if Debug:
23      print("Release displays")
24  displayio.release_displays()
25
```

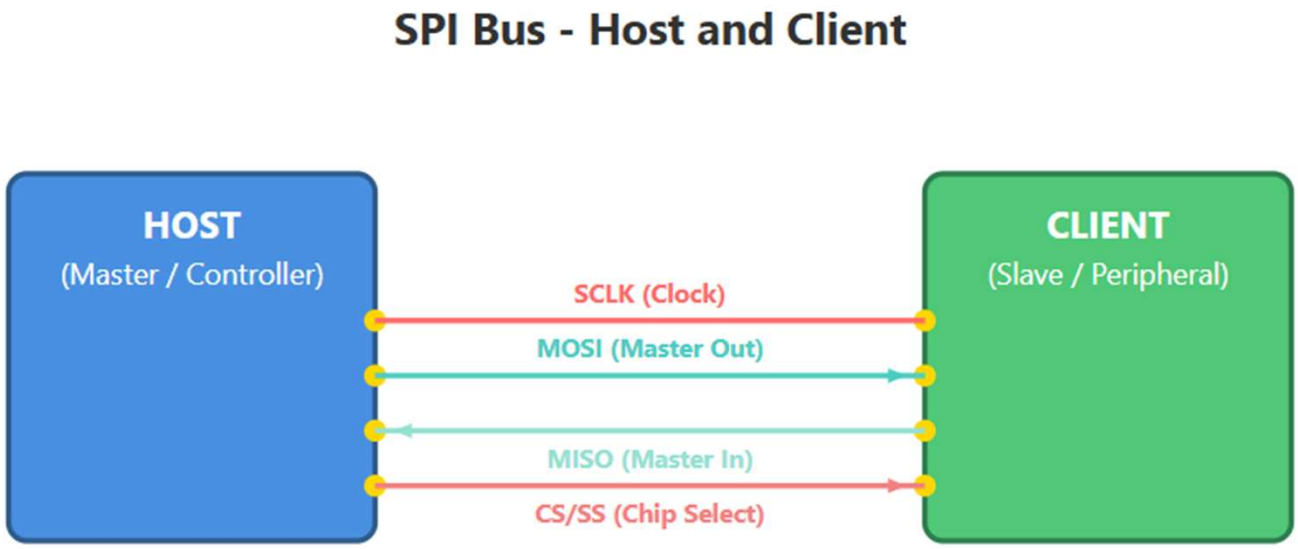
- **Q: Why do we Release Displays?**

MICROCHIP

SPI Primer

Serial Peripheral Interface

- **Key characteristics:**
- **Full-duplex:**
  - Data can flow both directions simultaneously
- **Host-controlled:**
  - Only the host can initiate communication
- **Synchronous:**
  - All transfers are synchronized to the clock
- **Multiple clients:**
  - One host can control many clients using separate CS lines for each



MICROCHIP

1-1_ST7789_Stationary_Meatball

Create SPI Object, Define TFT CS and DC pins

```
26    if Debug:
27        print("Create SPI Object for display")
28    spi = board.LCD_SPI()
29    tft_cs = board.LCD_CS
30    tft_dc = board.D4
```

- **Q: What is CS and DC?**

MICROCHIP

1-1_ST7789_Stationary_Meatball

Create FourWire Object – name it display_bus

```
35    if Debug:
36        print("Create DisplayBus")
37    display_bus = FourWire(spi, command=tft_dc, chip_select=tft_cs)
```

- **What is FourWire?**

MICROCHIP

1-1_ST7789_Stationary_Meatball

Create FourWire Object – name it display_bus

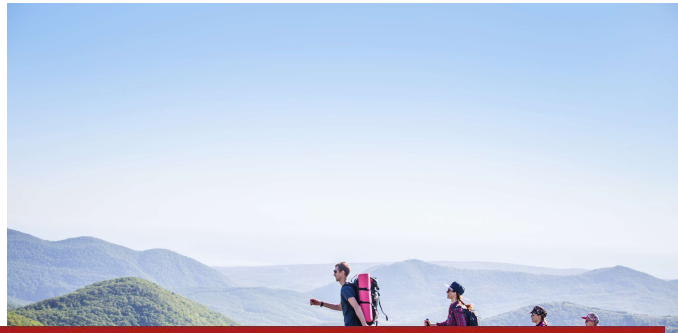
```
35  if Debug:
36      print("Create DisplayBus")
37  display_bus = FourWire(spi, command=tft_dc, chip_select=tft_cs)
```

- **Why "FourWire"?**
  - The name is a bit confusing because it actually uses MORE than 4 wires! It's called FourWire because it uses the 4 standard SPI wires PLUS additional control pins:
- **Standard SPI (4 wires):**
  - SCLK - Clock
  - MOSI - Data out (sometimes called SDA)
  - MISO - Data in (often unused for displays)
  - CS - Chip Select
- **Extra Display Control Pins:**
  - DC (Data/Command) - Tells the display whether you're sending commands or pixel data
  - RESET (optional) – We use a Hardware Reset controller

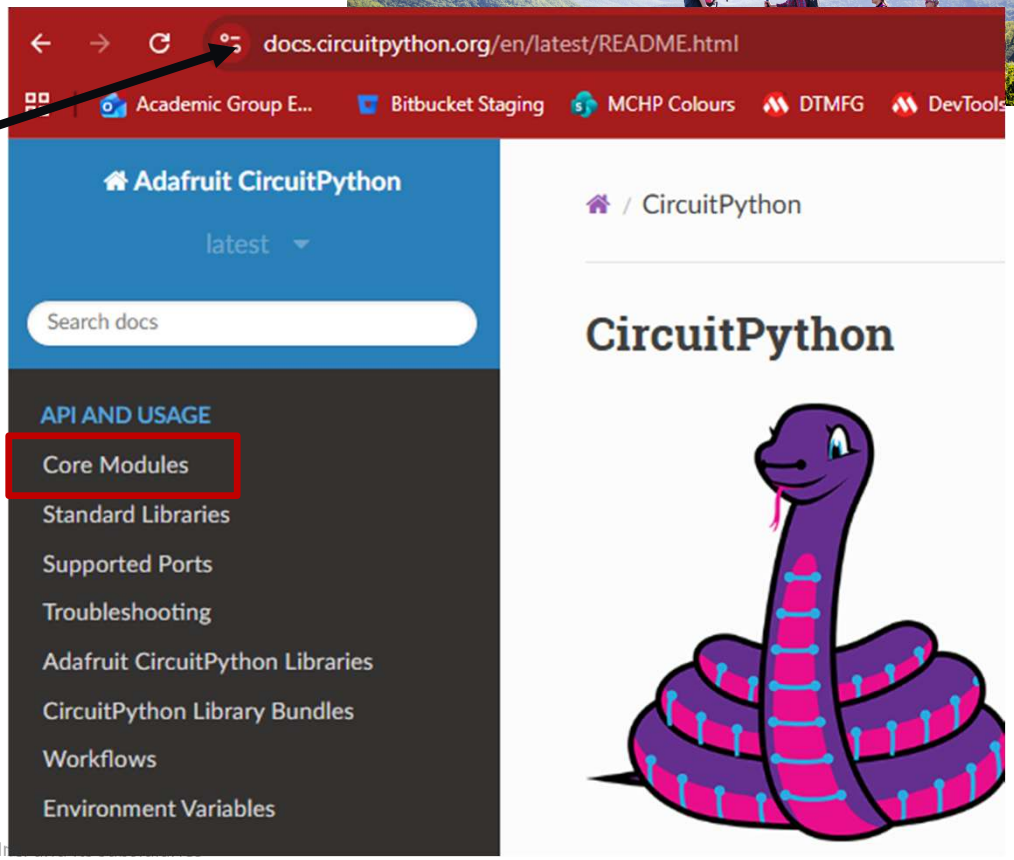
MICROCHIP

1-1_ST7789_Stationary_Meatball

Create FourWire Object – name it display_bus



- **Where do we find FourWire initialization?**
  - Let’s check the docs!
  - It's under *Core Modules*

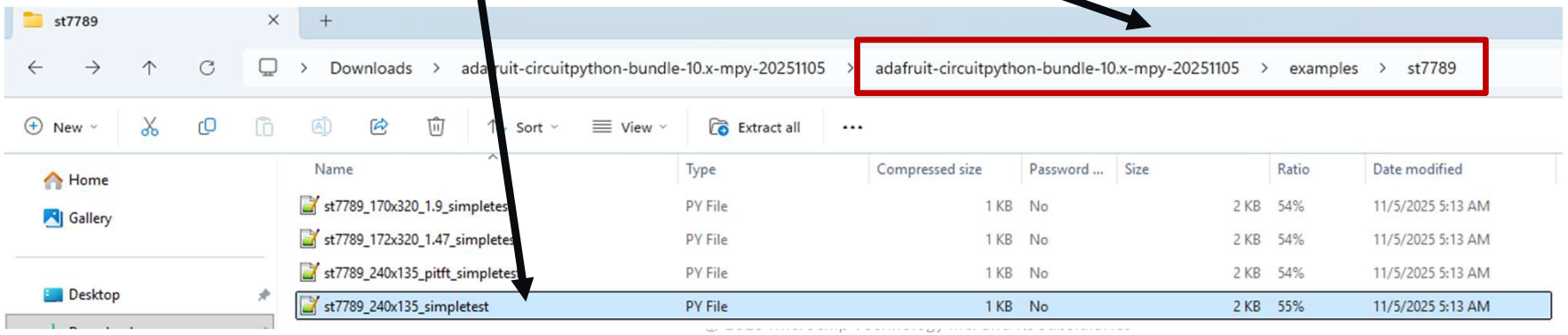


1-1_ST7789_Stationary_Meatball

Create ST7789 – name it display

```
38  display = ST7789(
39      display_bus, rotation=90, width=DISPLAY_WIDTH, height=DISPLAY_HEIGHT, rowstart=40, colstart=53)
```

- **What is ST7789?**
  - Let's check the docs – part of the Adafruit Bundle you downloaded and unzipped earlier
  - Under ***examples\st7789***
  - st7789_240x135_simpletest.py



MICROCHIP

Let's Move The Meatball Using User Button!

Simple Animation

- **We just need to add the button driver from earlier!**
- ***Before*** **the loop:**

```
# configure User Button as digital input
# NOTE: button is Active LOW
user_button = digitalio.DigitalInOut(board.D3)
user_button.direction = digitalio.Direction.INPUT
```

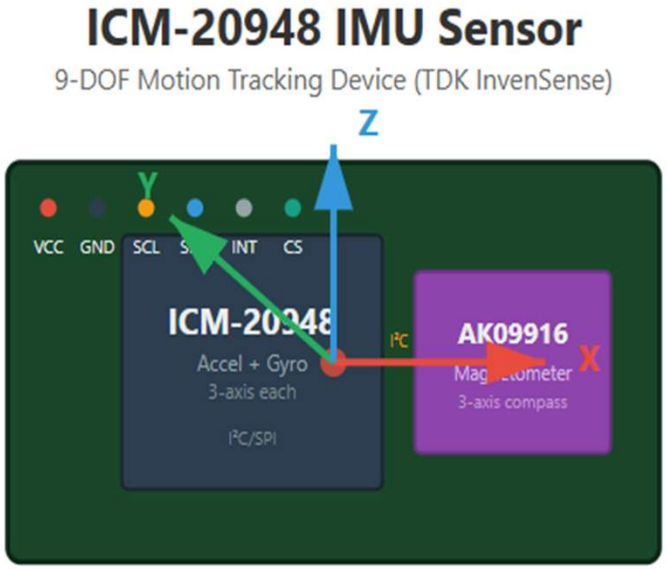
- **Then in the loop:**

```
while True:
    # read the button state
    button_state = user_button.value
    # if button pressed, move left
    if not button_state:
        group.x -= 1
    else:
    # otherwise move right
        group.x += 1
    time.sleep(0.05)
```

MICROCHIP

1-3_ICM20948_IMU

What’s an IMU (Inertial Measurement Unit) ?



ACCELEROMETER

(3-axis)

Measures:
Linear Acceleration
(±2, ±4, ±8, ±16g)

Applications:
- Tilt detection
- Free-fall detection
- Motion tracking
- Vibration analysis

16-bit ADC

GYROSCOPE

(3-axis)

Measures:
Angular Velocity
(±250, ±500, ±1000, ±2000°/s)

Applications:
- Rotation rate
- Stabilization
- Gaming controllers
- Drone navigation

16-bit ADC

MAGNETOMETER

(3-axis)

N

S

Measures:
Magnetic Field
(±4900μT)

Applications:
- Compass heading
- Absolute orientation
- Navigation
- AR/VR tracking

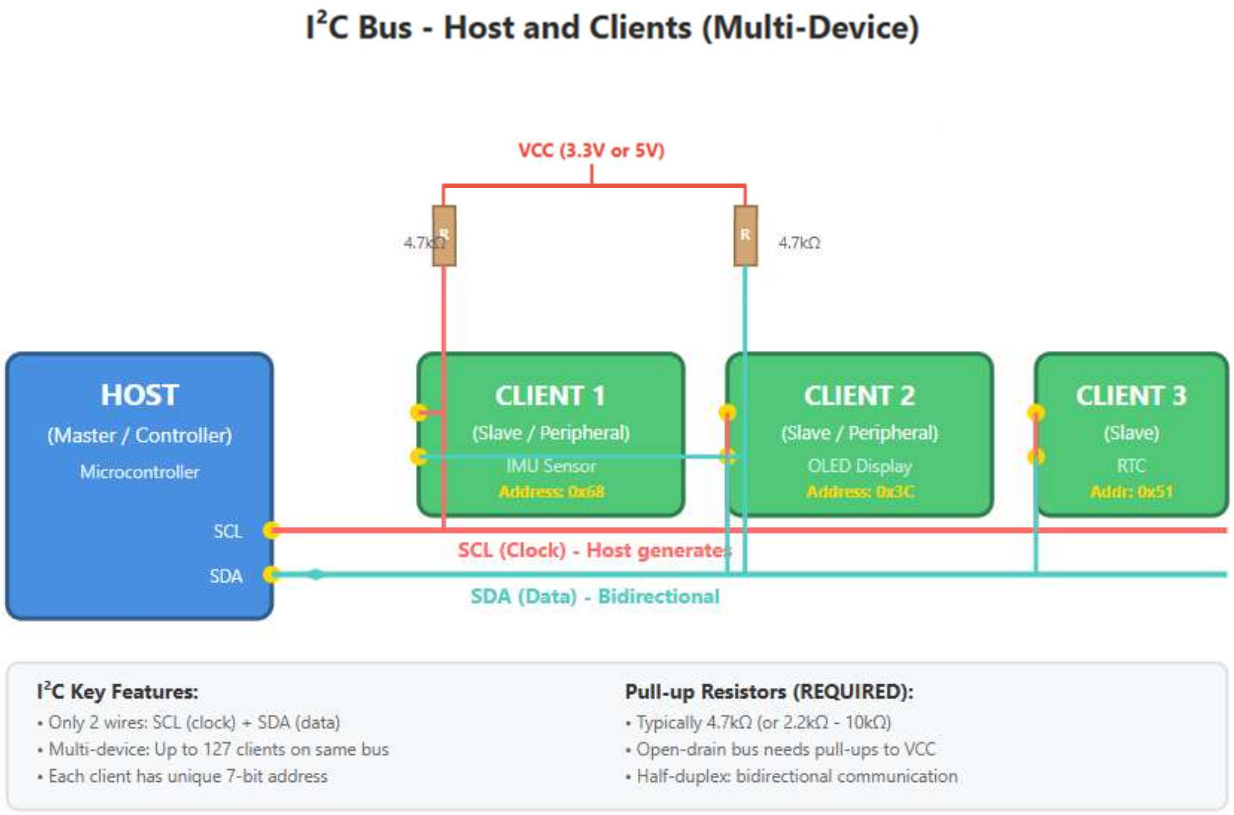
AK09916 chip

9-DOF | I²C (400kHz) & SPI (7MHz) | DMP (Digital Motion Processor) | Low Power | 3.3V

MICROCHIP

1-3_ICM20948_IMU

I²C (Inter-Integrated Circuit) Primer



- **Uses only 2 wires + GND**
  - SDA – Data
  - SCL – Clock
- **Uses device address**
  - Each device requires unique address
  - Devices often have alternate address
- **Open Drain bus:**
  - Requires pull-ups
  - Device can only pull bus LOW – not HIGH

MICROCHIP

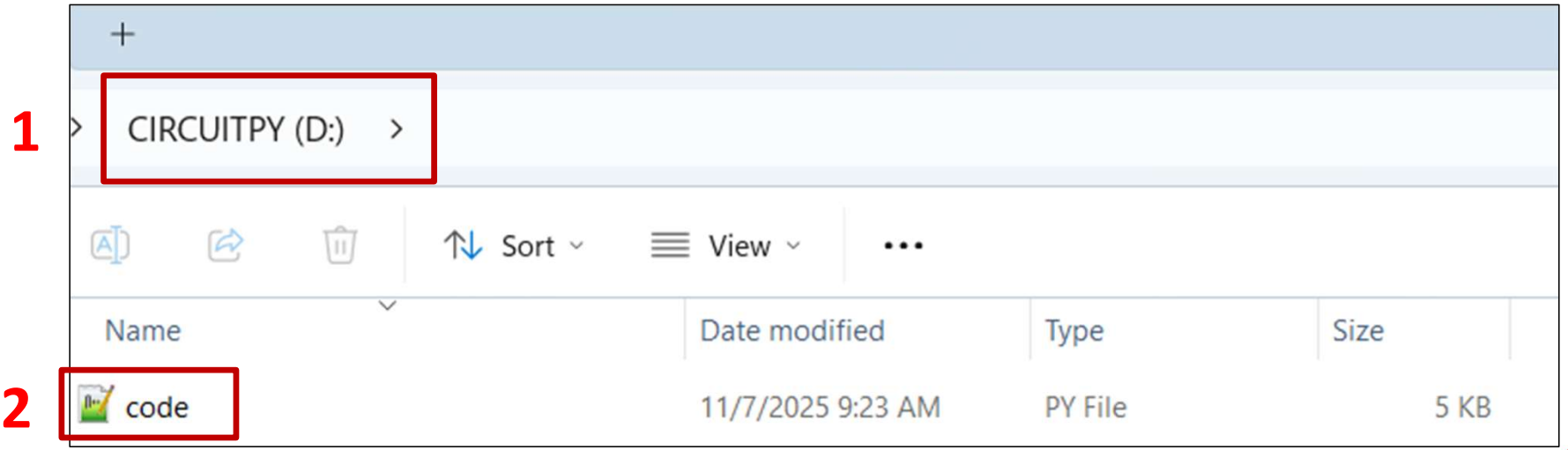
To Use GitHub Repo Code

Using GitHub Repo code

- **Recall:**

  To run a CircuitPython program, it must be:

  1. In the **root** directory of **D:\CIRCUITPY**
  2. Named ***code.py*** and



MICROCHIP

1-3_ICM20948_IMU

Copy *code.py* and */lib/* to CIRCUITPY

- **Read from the I²C sensor and display in Serial Monitor**
- **Imports:**

```
import time
import board
import adafruit_icm20x
```

- **ICM20948 has 2 possible device addresses: 0x68 and 0x69**

```
try:
    icm = adafruit_icm20x.ICM20948(i2c, 0x69)
except:
    print("No ICM20948 found at default address 0x69. Trying alternate address 0x68.")
    try:
        icm = adafruit_icm20x.ICM20948(i2c, 0x68)
    except:
        print("No ICM20948 device found! Make sure the dev board and ruler are connected properly!")
```


MICROCHIP

1-3_ICM20948_IMU

Copy *code.py* and */lib/* to CIRCUITPY

- **Read from the I²C sensor and display in Serial Monitor**
- **Read accelerometer data as X,Y,Z tuple:**

```
while True:
    X, Y, Z = icm.acceleration

    if Debug:
        print("X: {:.2f}".format(X))
        print("Y: {:.2f}".format(Y))
        print("Z: {:.2f}".format(Z))
        print("")

    time.sleep(0.1)
```

MICROCHIP

Now Let's Put It All Together!

Use IMU data to move Meatball on TFT Display

- **Use the code from:**
  - 1-1_ST7789_Stationary_Meatball
  - 1-2_Meatball_Move_Button_Press
  - 1-3_ICM20948_IMU
- **Put it all together to make the IMU tilt move the Meatball!**

MICROCHIP

What Does It Do?

Microchip “Meatball” controlled by IMU - Accelerometer

- **Try tilting the Ruler**
  - What happens?
- **How do we fix this?**
  - Hint: You will need to create a new variable to determine whether the bounds of the LCD have been breached.

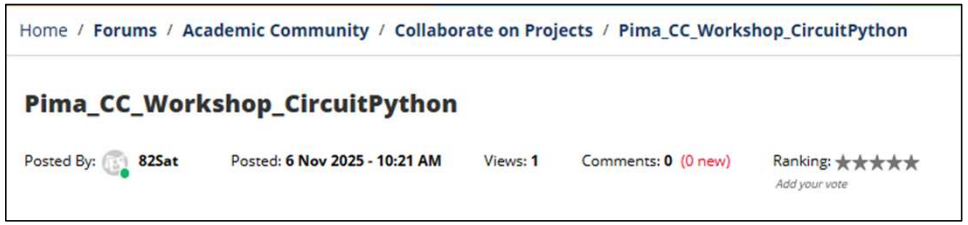
After The Workshop

Are You Stuck?

Forum Link:

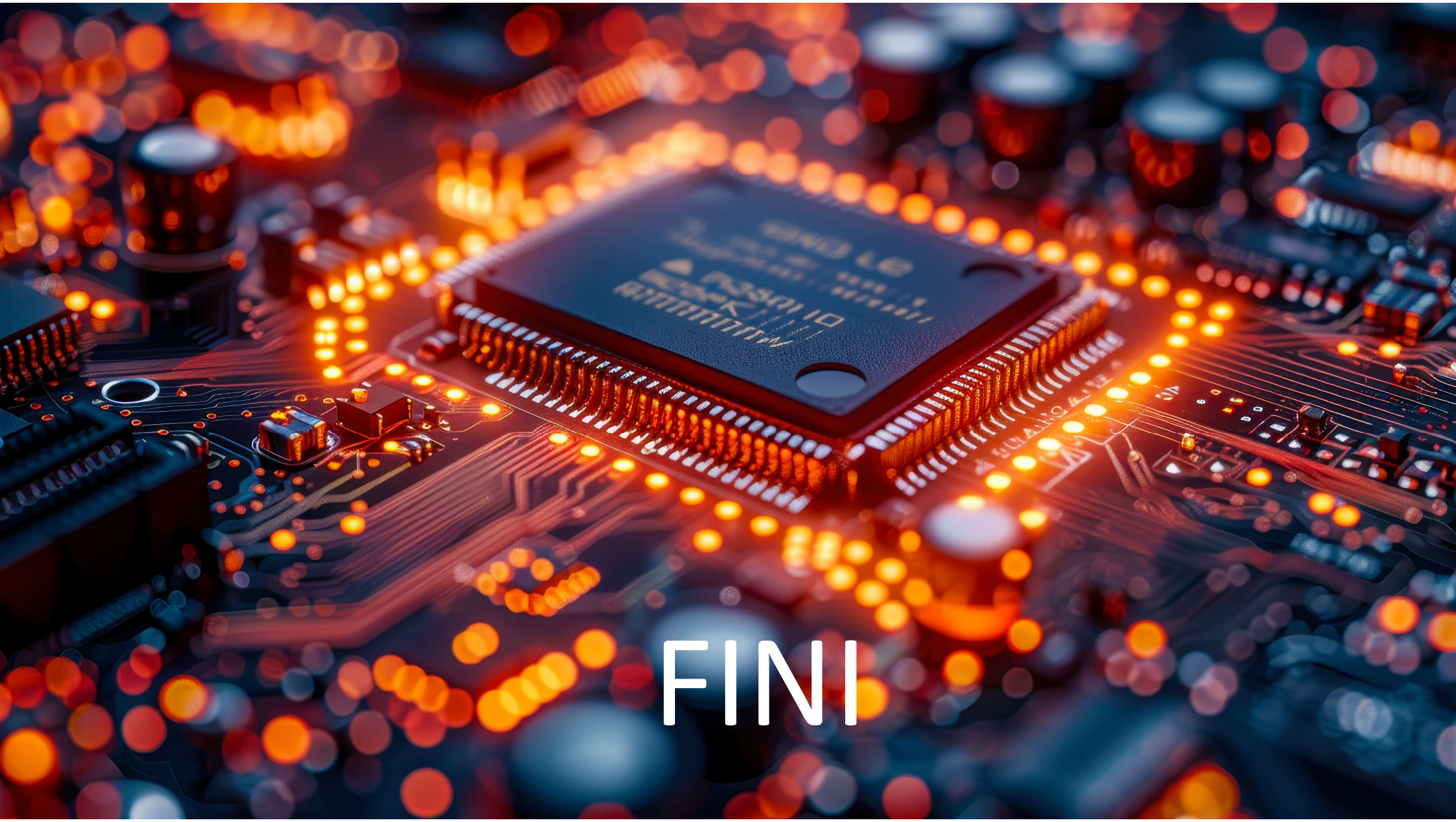
Pima_CC_Workshop_CircuitPython

- **In the Git repo open the folder:**
  - *1-x_Completed*
- **Then open:**
  - *How_To_Complete_This_Project.txt*



- **You will find a link to the forum**
  - Post your questions there
  - Try to help each other in person & on the forum
  - ***We will also be monitoring the forum to help.***

MICROCHIP


FINI