website: vanderbi.lt/codegraf

Microcontroller and CircuitPython background

Presenter: Steve Baskauf
steve.baskauf@vanderbilt.edu

DiSC DIGITAL SCHOLARSHIP AND COMMUNICATIONS

Jean & Alexander Heard LIBRARIES

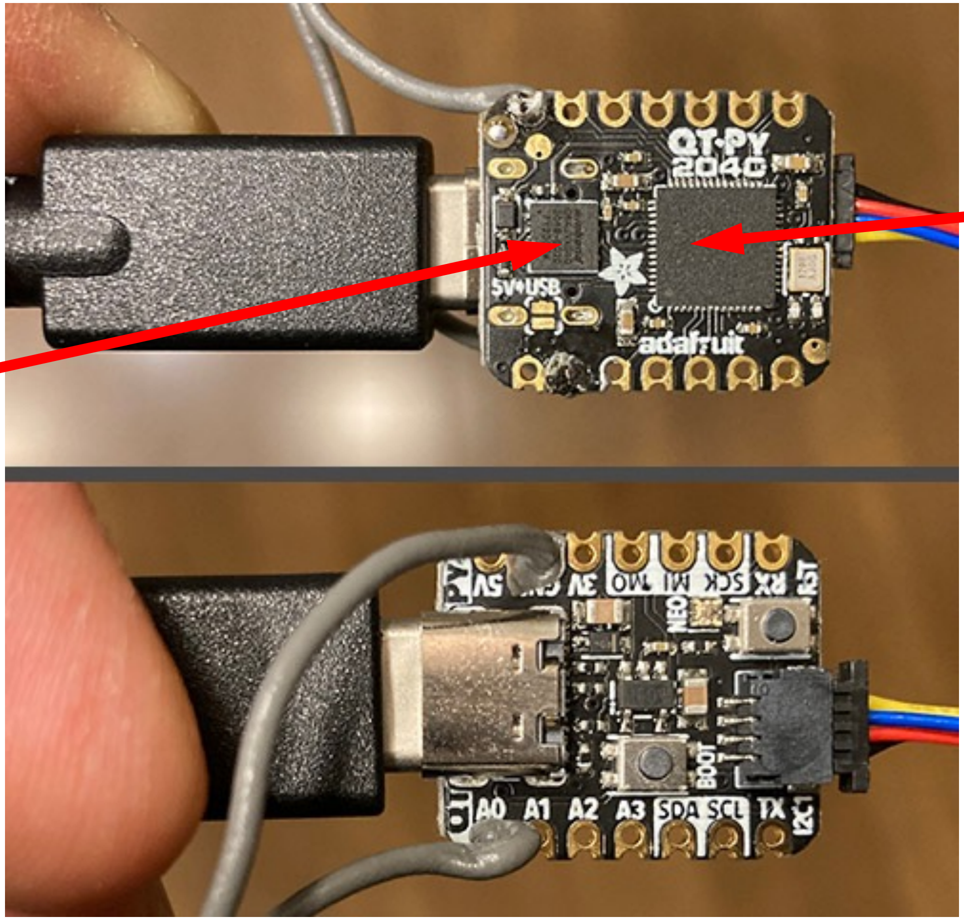
website: vanderbi.lt/codegraf

QT Py RP2040 microcontroller

DiSC DIGITAL SCHOLARSHIP AND COMMUNICATIONS

Jean & Alexander Heard LIBRARIES

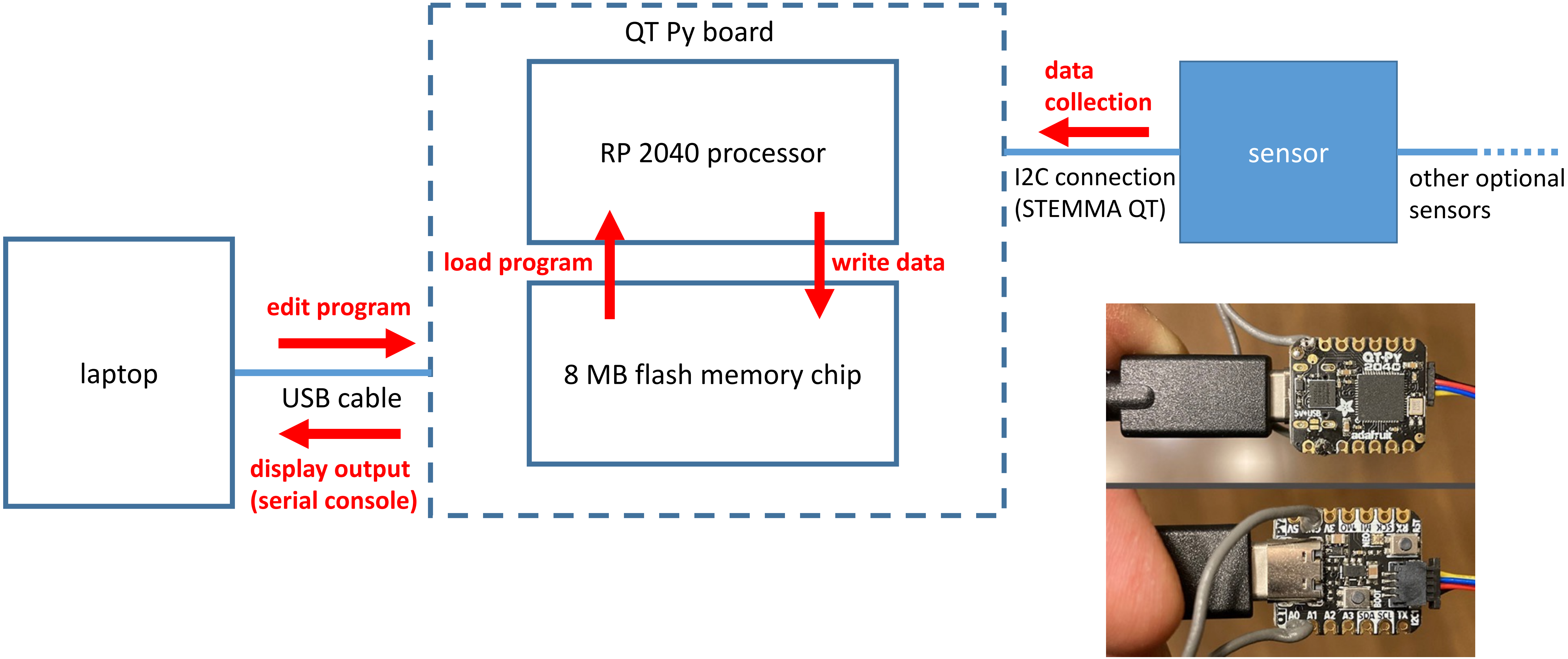
What's a microcontroller?



Computer-on-a-chip
(RP2040 chip)

additional memory
(8 MB flash)

System architecture



QT Py board

RP 2040 processor

8 MB flash memory chip

load program

write data

laptop

edit program

USB cable

display output (serial console)

data collection

sensor

I2C connection (STEMMA QT)

other optional sensors

website: vanderbi.lt/codegraf

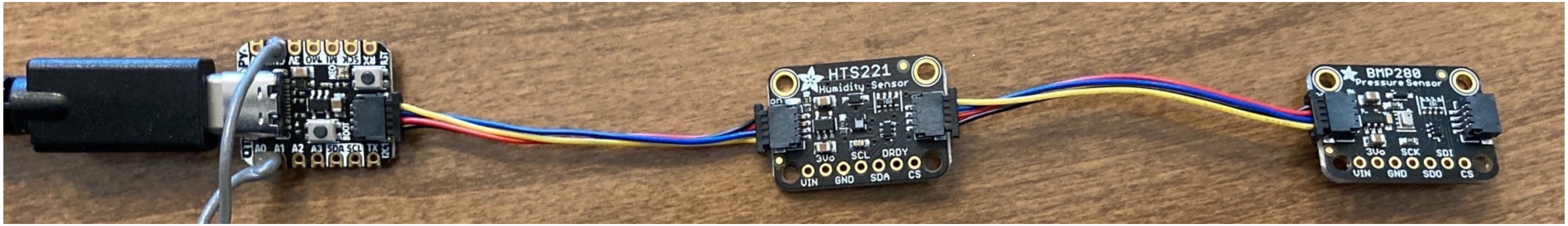
Connecting sensors

DiSC DIGITAL SCHOLARSHIP AND COMMUNICATIONS

Jean & Alexander Heard LIBRARIES

What is I2C ?

- I2C (pronounced "eye squared C") is a simple communications protocol
- It requires only 2 wires (clock and data) for communications (plus 2 power wires).
- All devices share the same 2 wires and can be chained.
- Devices are identified by a unique address (127 possible).



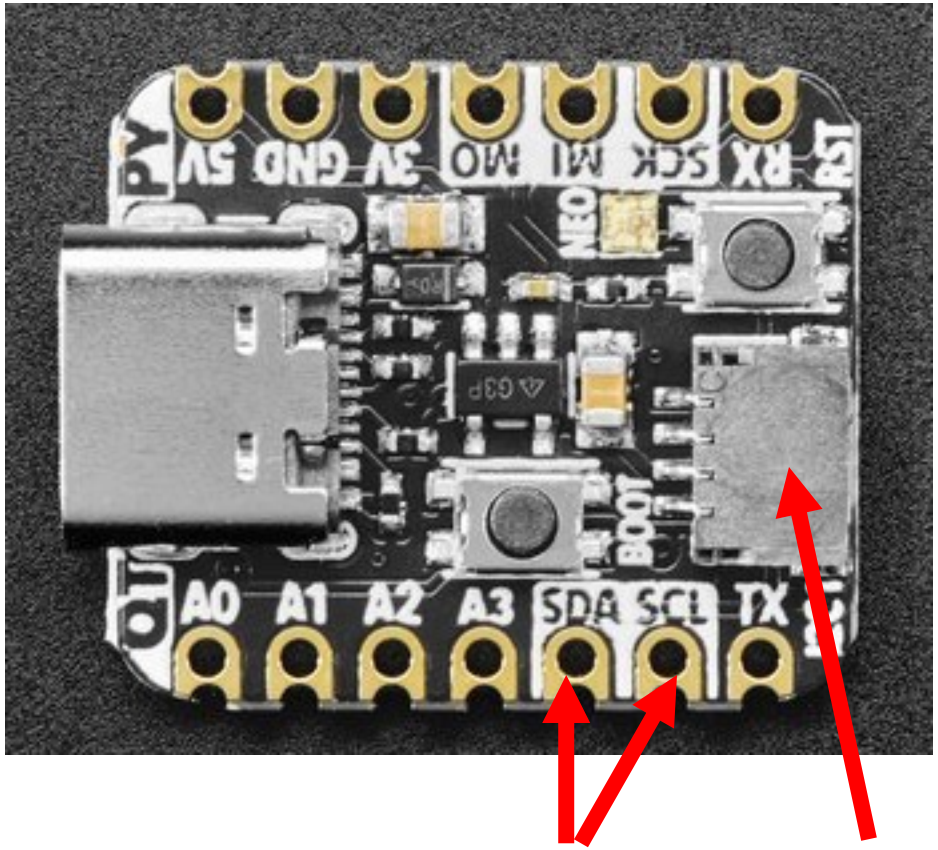
QT Py RP2040
using IC2 controller 1

wires
(only 2
needed)

HTS221
Humidity/Temp sensor
address: 0x5F

BMP280
Barometric Pressure sensor
address: 0x76

I2C connections



I2C0 connections on board pads (must be soldered)

I2C1 connections on STEMMA QT connector

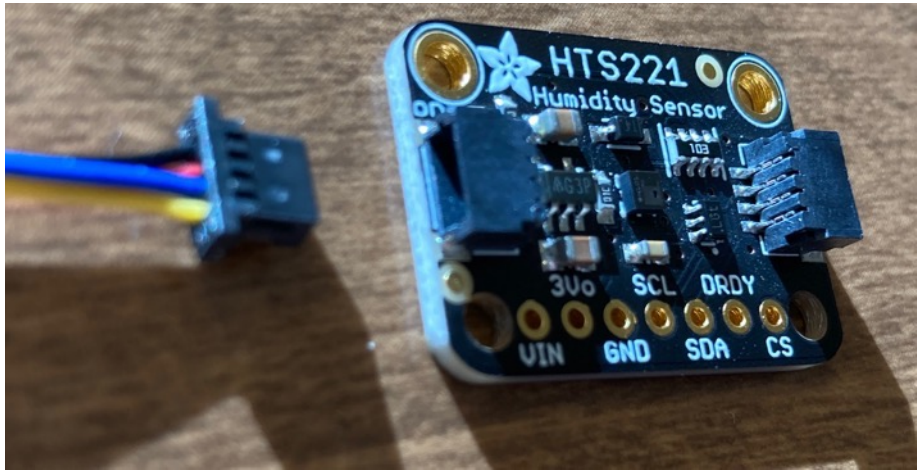
QT Py RP2040 has two I2C controllers. They are accessed via:

- **SDA** and **SCL** pads on board (**I2C0**)
- STEMMA QT connector (**I2C1**)

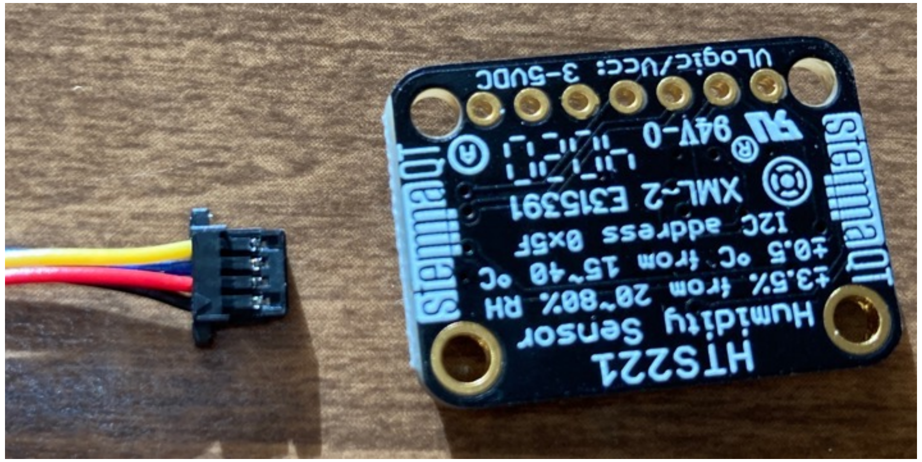
They are addressed in code as:

- **board.SCL** and **board.SDA** (**I2C0**)
- **board.SCL1** and **board.SDA1** (**I2C1**)

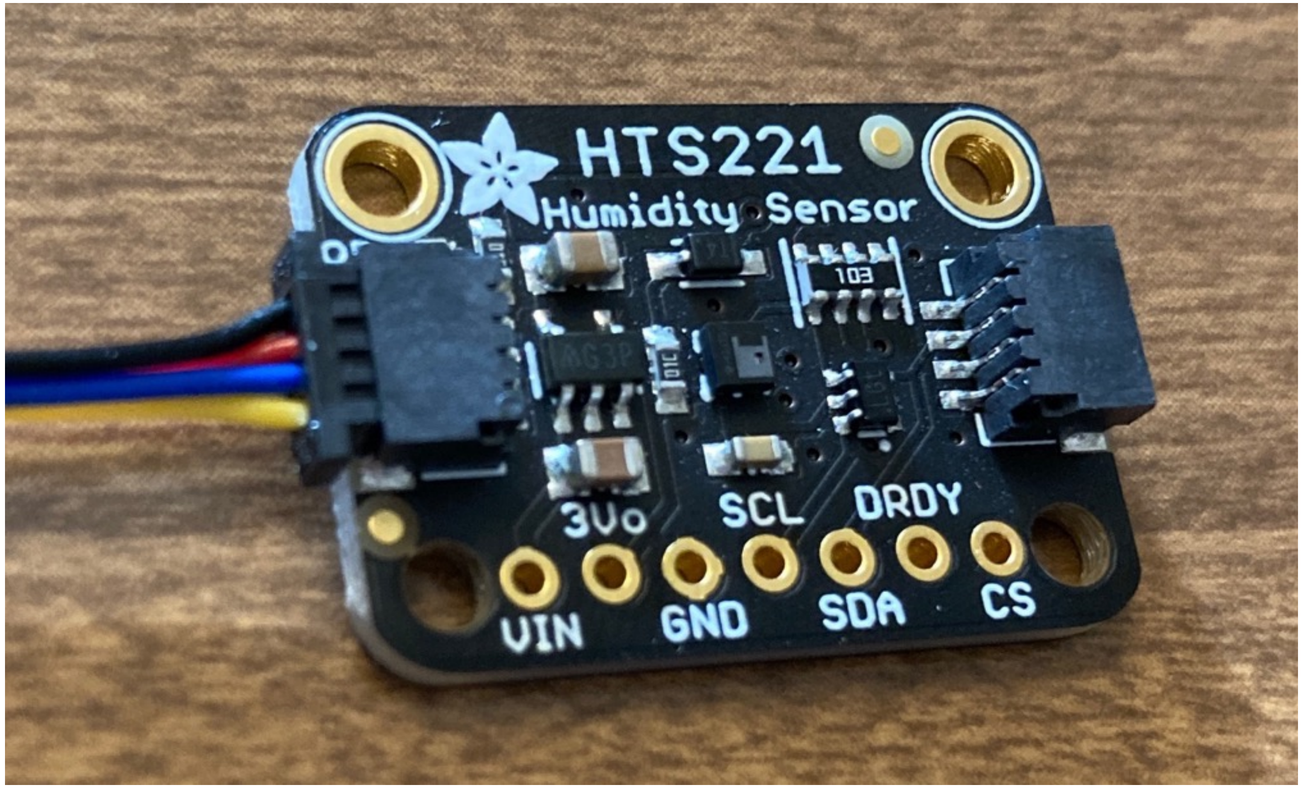
STEMMA QT connectors



top view



bottom view



cable firmly seated in connector
(either connector can be used)

website: vanderbi.lt/codegraf

CircuitPython

DiSC DIGITAL SCHOLARSHIP AND COMMUNICATIONS

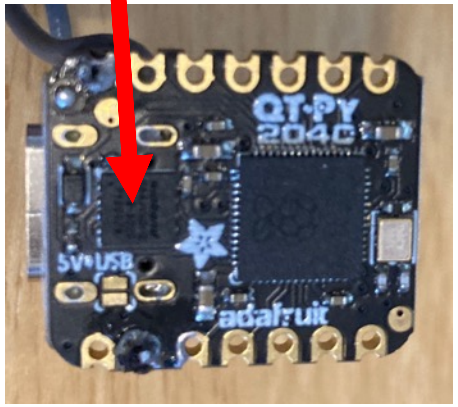
Jean & Alexander Heard LIBRARIES

Storage requirements

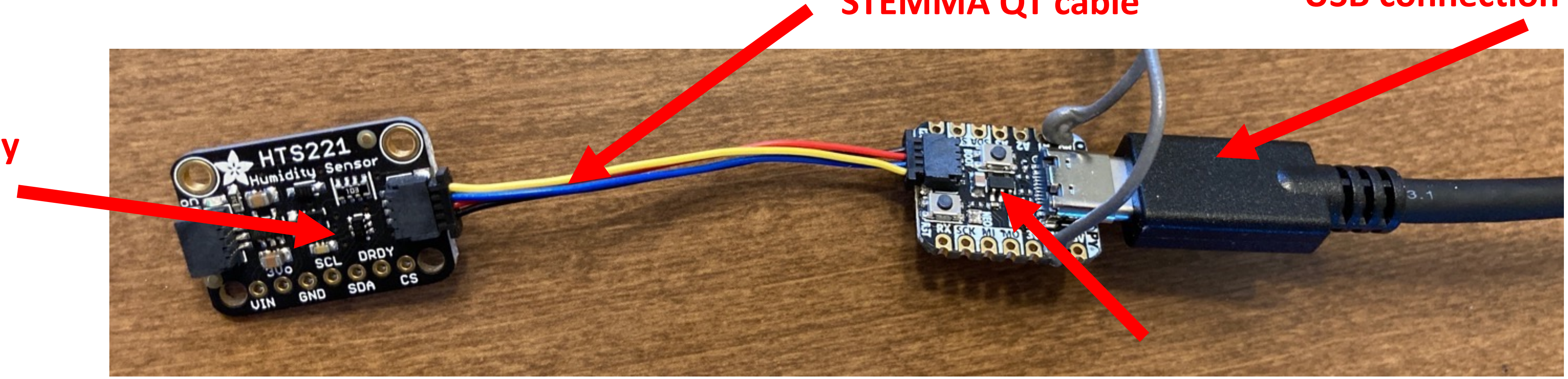
Standard cPython installation: 90 MB

Laptop: up to 1 TB (1 million MB)

QT Py RP2040: 8 MB flash memory



Aspects of CircuitPython



STEMMA QT cable

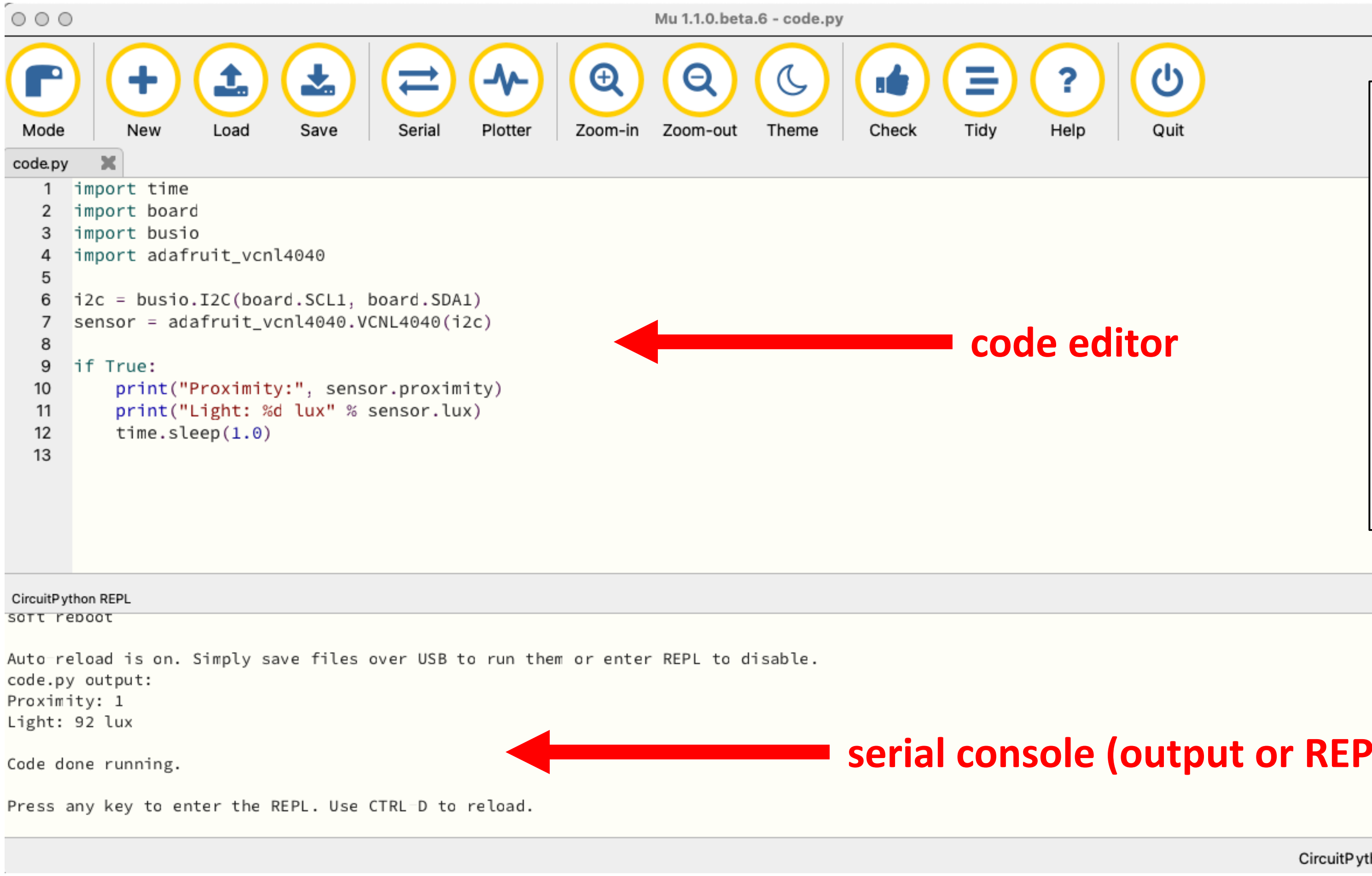
USB connection to laptop

humidity sensor

microcontroller board

- Stripped down Python language code
- Includes "operating system" for microcontroller board
- Individually installed device libraries support 300 connected devices
- Simplified communication with laptop via USB
- Automated launching of **`code.py`** file

Running CircuitPython with the Mu editor

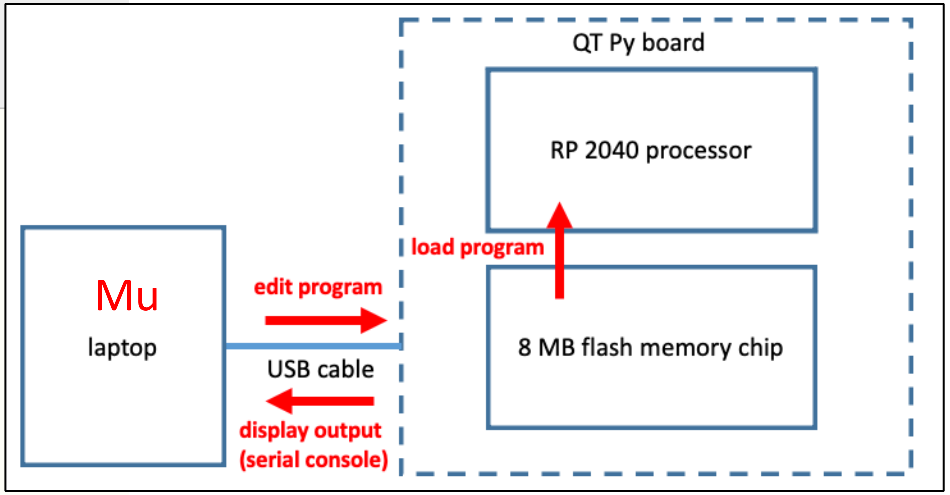


```python
import time
import board
import busio
import adafruit_vcnl4040

i2c = busio.I2C(board.SCL1, board.SDA1)
sensor = adafruit_vcnl4040.VCNL4040(i2c)

if True:
    print("Proximity:", sensor.proximity)
    print("Light: %d lux" % sensor.lux)
    time.sleep(1.0)

```

code editor

```
soft reboot

Auto-reload is on. Simply save files over USB to run them or enter REPL to disable.
code.py output:
Proximity: 1
Light: 92 lux

Code done running.

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

serial console (output or REPL)

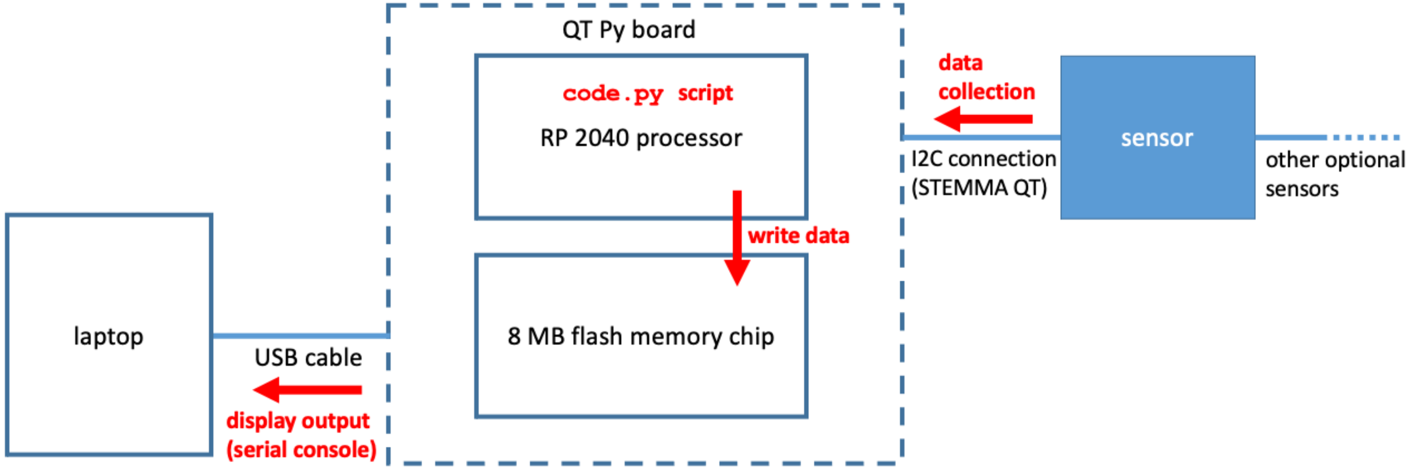


The `code.py` script

Why do we need `code.py` ?

- No peripherals (keyboard, monitor) so no sophisticated way to "talk" with the processor.
- Once we get `code.py` running properly, it can manage the microcontroller without our intervention.

What does the `code.py` script do?



QT Py board

code.py script

RP 2040 processor

write data

8 MB flash memory chip

laptop

USB cable

display output (serial console)

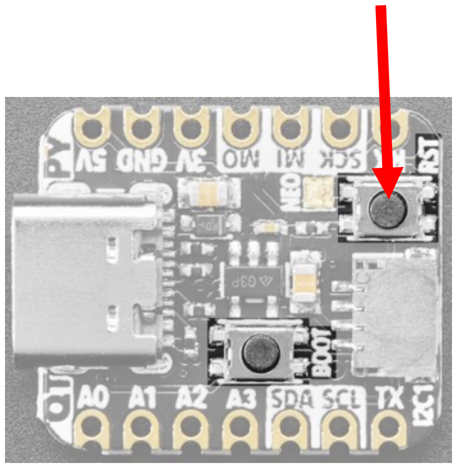
data collection

I2C connection (STEMMA QT)

sensor

other optional sensors

reset button (RST)



The **code.py** script is a special file in CircuitPython.

It controls interactions with sensors, memory, and the serial console.

It is executed when:

- the board is powered up.
- when you save something in the "drive" memory (including **code.py**).
- when you do a "hard reset" by pushing the reset button on the board
- when you do a "soft reset" (CTRL-D) from the console

Example: `code.py` for VCNL4040 proximity sensor

load code modules

```
import time
import board
import adafruit_vcnl4040
```

instantiate objects

```
i2c = board.I2C()
sensor = adafruit_vcnl4040.VCNL4040(i2c)
```

read sensors and report

```
while True:
    print("Proximity:", sensor.proximity)
    print("Light: %d lux" % sensor.lux)
    time.sleep(1.0)
```

website: vanderbi.lt/codegraf

Review of modules

DISC DIGITAL SCHOLARSHIP AND COMMUNICATIONS

Jean & Alexander Heard LIBRARIES

What are modules?

- **Modules** are reusable code loaded from a file. "Library" is a similar term to "module".
- They're loaded as needed using an **`import`** statement.
- Modules in the **standard library** are ready to load.
- Modules not in the standard library must be **installed**.

- In standard Python, the command line **installer application PIP** is typically used to install modules:

```
pip install requests
```

Modules in CircuitPython

- Modules are installed manually in CircuitPython
- Download the CircuitPython Library Bundle to get all the library files for a microcontroller board.
- Library files have an **`.mpy`** extension.
- Copy manually from the **`lib`** directory in the bundle to the **`lib`** directory on your board to install.
- Modules from copied library files can be imported into a script.
- Typically, modules that need to be installed have very specific code for particular sensors.

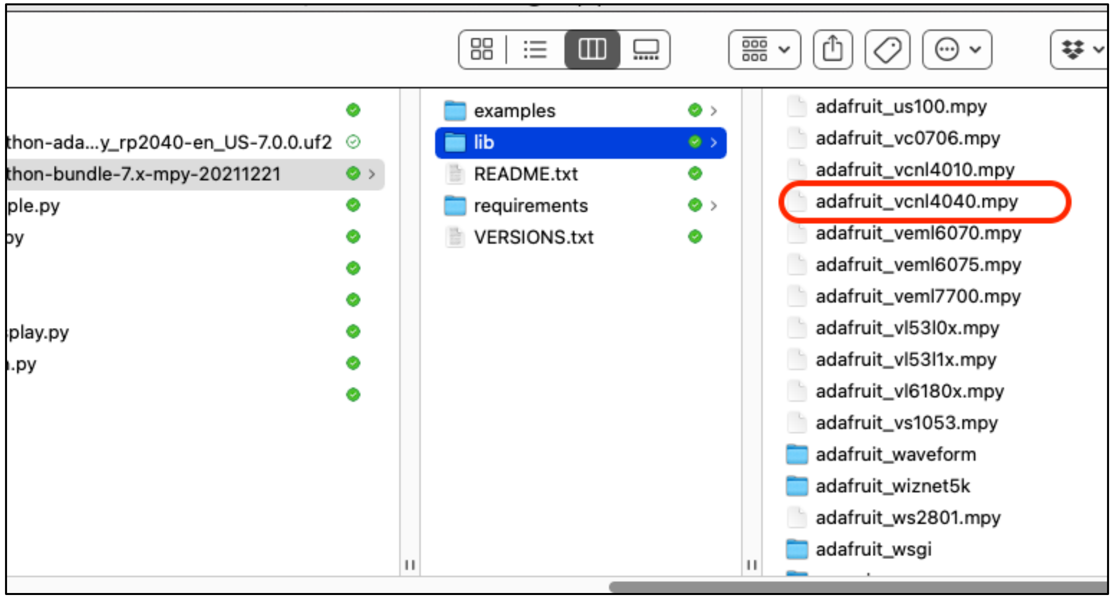
Example: `code.py` for VCNL4040 proximity sensor

```
import time
import board
import adafruit_vcnl4040
```

standard library modules (don't need to be installed)

module in Library Bundle (needs to be copied)

```
i2c = board.I2C()
sensor = adafruit_vcnl4040.VCNL4040(i2c)

while True:
    print("Proximity:", sensor.proximity)
    print("Light: %d lux" % sensor.lux)
    time.sleep(1.0)
```



thon-ada...y_rp2040-en_US-7.0.0.uf2
thon-bundle-7.x-mpy-20211221
ple.py
py
play.py
.py

examples
lib
README.txt
requirements
VERSIONS.txt

adafruit_us100.mpy
adafruit_vc0706.mpy
adafruit_vcnl4010.mpy
adafruit_vcnl4040.mpy
adafruit_veml6070.mpy
adafruit_veml6075.mpy
adafruit_veml7700.mpy
adafruit_vl53l0x.mpy
adafruit_vl53l1x.mpy
adafruit_vl6180x.mpy
adafruit_vs1053.mpy
adafruit_waveform
adafruit_wiznet5k
adafruit_ws2801.mpy
adafruit_wsgi

What is Blinka?

- Sensors can be also be run from a laptop or a single-board computer like a Raspberry Pi.
- The Blinka library can be installed using PIP in standard CPython (Linux, Mac OS, Windows)
- It contains modules that correspond to the modules that are installed manually from the CircuitPython Library Bundle.
- Modules from Blinka are imported in the normal way.

website: vanderbi.lt/codegraf

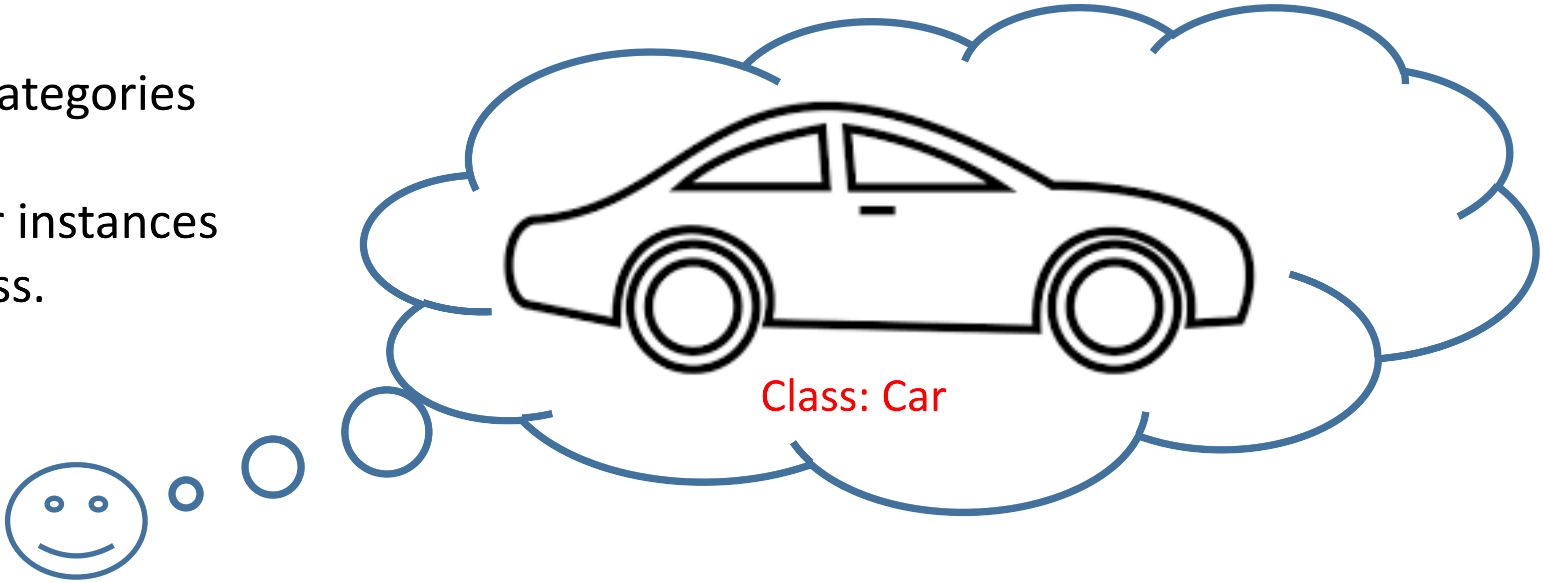
CircuitPython objects

DiSC DIGITAL SCHOLARSHIP AND COMMUNICATIONS

Jean & Alexander Heard LIBRARIES

classes and objects

Classes are abstract categories of things.
Objects are particular instances or individuals of a class.



Class: Car

object: toyotaPrius

object: ferrari

object: volkswagenBeetle

Instantiating a custom object

- Python has many built-in classes or object types (e.g. lists, dictionaries, strings)
- Programmers can define custom classes using Object Oriented Programming. Class names are usually capitalized.
- Creating an object of a particular class is called instantiating the object. Example:

  ```
  sort_button = Button()
  ```

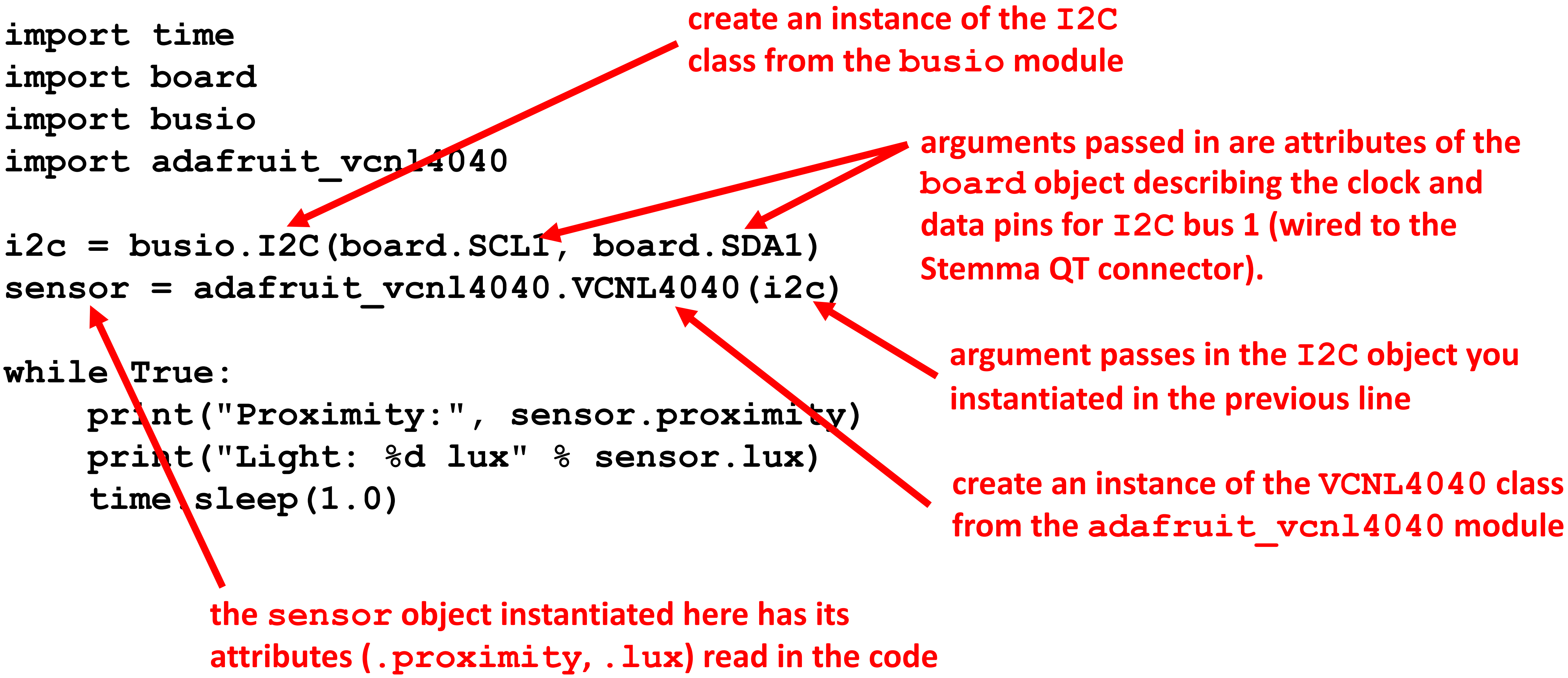
- If the class is defined in a module, the module name must be prepended to the class name when the object is created. Example:

  ```
  fitness_matrix = algebra.Matrix()
  ```

- Sometimes arguments need to be passed into the class when the object is created. Example:

  ```
  the_raven = Poem(title='The Raven', text='Quoth the Raven, nevermore!')
  ```

CircuitPython code example



```
import time
import board
import busio
import adafruit_vcnl4040

i2c = busio.I2C(board.SCL1, board.SDA1)
sensor = adafruit_vcnl4040.VCNL4040(i2c)

while True:
    print("Proximity:", sensor.proximity)
    print("Light: %d lux" % sensor.lux)
    time.sleep(1.0)
```

create an instance of the `I2C` class from the `busio` module

arguments passed in are attributes of the `board` object describing the clock and data pins for `I2C` bus 1 (wired to the Stemma QT connector).

argument passes in the `I2C` object you instantiated in the previous line

create an instance of the `VCNL4040` class from the `adafruit_vcnl4040` module

the `sensor` object instantiated here has its attributes (`.proximity`, `.lux`) read in the code

website: vanderbi.lt/codegraf

Code for reading sensors

DISC DIGITAL SCHOLARSHIP AND COMMUNICATIONS

Jean & Alexander Heard LIBRARIES

CircuitPython code example: fixed number of measurements

```
import time
import board
import busio
import adafruit_vcnl4040

i2c = busio.I2C(board.SCL1, board.SDA1)
sensor = adafruit_vcnl4040.VCNL4040(i2c)

for reading in range(10):
     print("Proximity:", sensor.proximity)
     print("Light: %d lux" % sensor.lux)
     time.sleep(1.0)
```

`for` loop executes code block 10 times

Access sensor object's attribute `.proximity`

Access sensor object's attribute `.lux`

wait 1 second between measurements

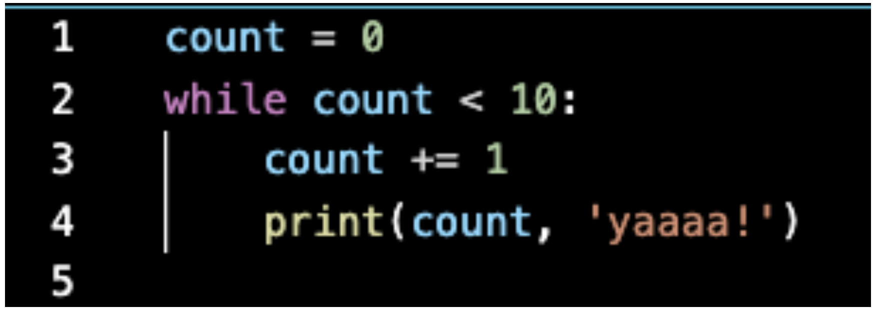
CircuitPython code example: one measurement

```
import time
import board
import busio
import adafruit_vcnl4040

i2c = busio.I2C(board.SCL1, board.SDA1)
sensor = adafruit_vcnl4040.VCNL4040(i2c)

if True:
    print("Proximity:", sensor.proximity)
    print("Light: %d lux" % sensor.lux)
    time.sleep(1.0)
```

conditional code block executes one time

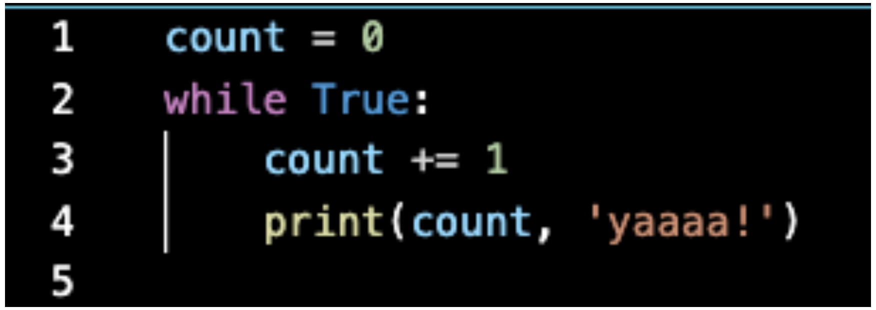
Infinite loops

- Normally we don't want code to run forever.
- **`while`** loops run until a condition becomes **`False`**
- We might want a sensor to run indefinitely.
- They will run forever if the condition is hard-coded to **`True`**
- Terminate an infinite loop using CTRL-Z or CTRL-C



```python
count = 0
while count < 10:
    count += 1
    print(count, 'yaaaa!')

```

normal while loop



```python
count = 0
while True:
    count += 1
    print(count, 'yaaaa!')

```

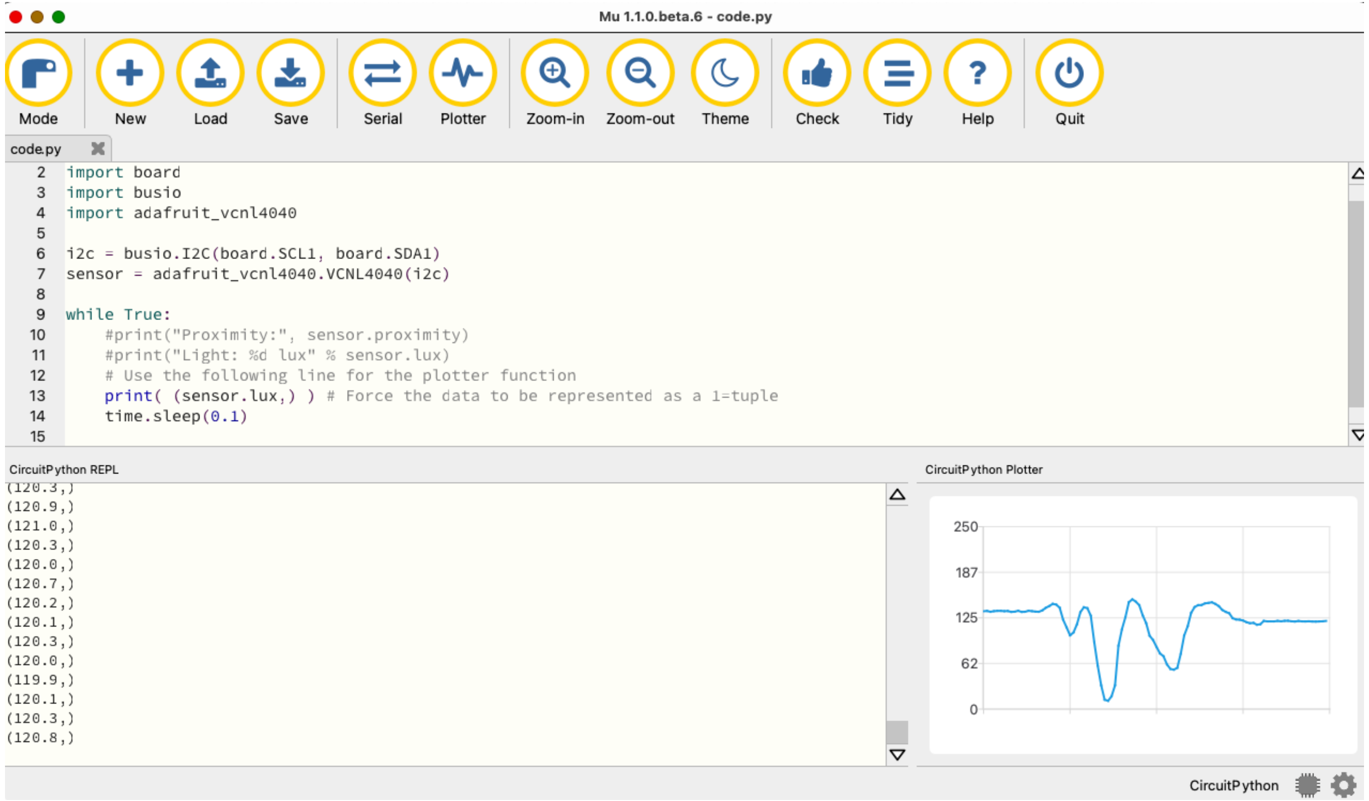
infinite while loop

CircuitPython code example: indefinite number of measurements

```
import time
import board
import busio
import adafruit_vcnl4040

i2c = busio.I2C(board.SCL1, board.SDA1)
sensor = adafruit_vcnl4040.VCNL4040(i2c)

while True:
    print("Proximity:", sensor.proximity)
    print("Light: %d lux" % sensor.lux)
    time.sleep(1.0)
```

while loop executes indefinitely

Using the Mu plotter

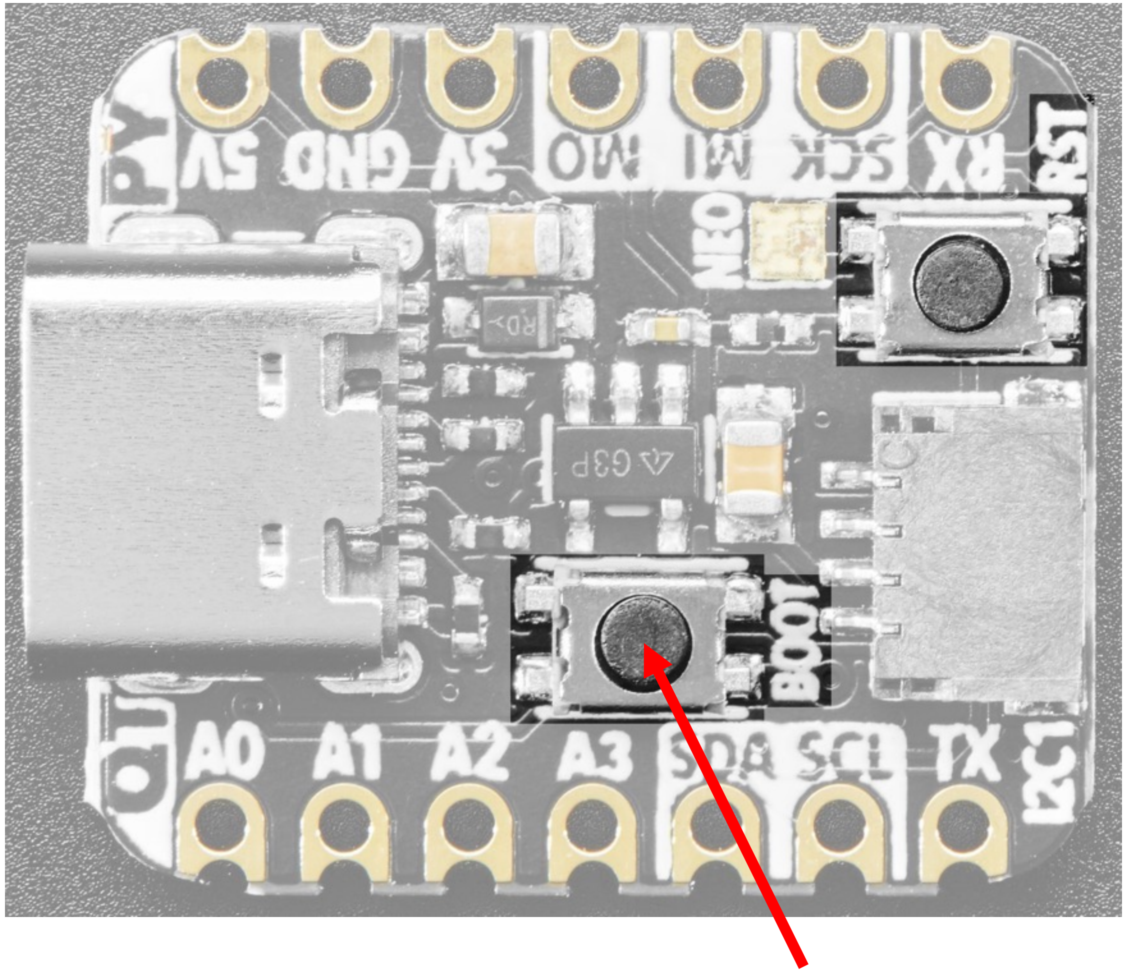


```
import board
import busio
import adafruit_vcnl4040

i2c = busio.I2C(board.SCL1, board.SDA1)
sensor = adafruit_vcnl4040.VCNL4040(i2c)

while True:
    #print("Proximity:", sensor.proximity)
    #print("Light: %d lux" % sensor.lux)
    # Use the following line for the plotter function
    print( (sensor.lux,) ) # Force the data to be represented as a 1=tuple
    time.sleep(0.1)

```

```
(120.3,)
(120.9,)
(121.0,)
(120.3,)
(120.0,)
(120.7,)
(120.2,)
(120.1,)
(120.3,)
(120.0,)
(119.9,)
(120.1,)
(120.3,)
(120.8,)
```

```
import time
import board
import busio
import adafruit_vcnl4040

i2c = busio.I2C(board.SCL1, board.SDA1)
sensor = adafruit_vcnl4040.VCNL4040(i2c)

while True:
    # Force the data to be represented as a 1=tuple
    print( (sensor.lux,) )
    time.sleep(0.1)
```

Triggering measurements with the BOOT button



user-defined button (BOOT)

- The **BOOT** button is a general-purpose button whose use can be defined in the **code.py** script
- It is identified as board.BUTTON

CircuitPython code example: triggering measurements by button

```
import time
import board
import busio
import digitalio
import adafruit_vcnl4040

i2c = busio.I2C(board.SCL1, board.SDA1)
sensor = adafruit_vcnl4040.VCNL4040(i2c)
button = digitalio.DigitalInOut(board.BUTTON)
button.switch_to_input(pull=digitalio.Pull.UP)

while button.value: # will be True when button not pressed
    pass
for reading in range(10):
    print("Proximity:", sensor.proximity)
    print("Light: %d lux" % sensor.lux)
    time.sleep(1.0)
```

module for digital input and output

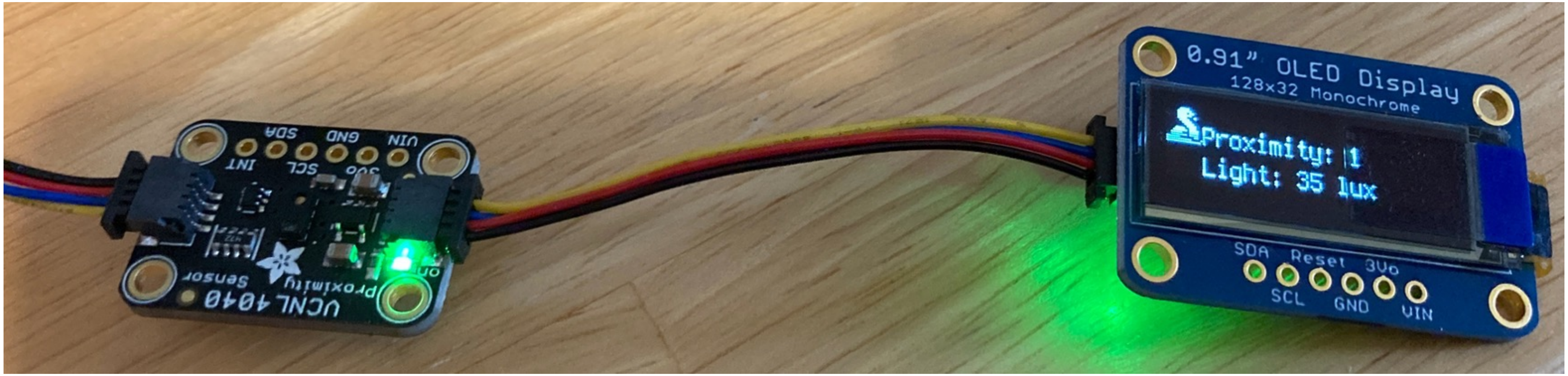
instantiate button object

define button behavior

access .value attribute of button object

code to do nothing

Sensor with display



- Code to operate display is more complicated.
- Makes it possible to monitor sensor independently of laptop (battery operation)

website: vanderbi.lt/codegraf

Sources of information

DiSC DIGITAL SCHOLARSHIP AND COMMUNICATIONS

Jean & Alexander Heard LIBRARIES

General reference documentation

- Circuit Python reference

https://learn.adafruit.com/circuitpython-essentials/circuitpython-essentials

- QT Py RP2040 microcontroller

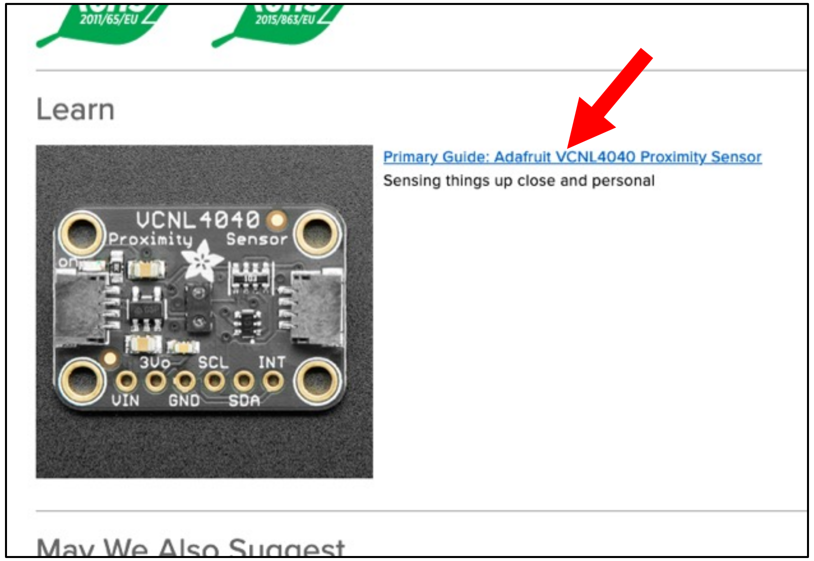
https://learn.adafruit.com/adafruit-qt-py-2040

- DiSC GitHub repository

https://github.com/HeardLibrary/digital-scholarship/tree/master/code/circuit_python

Sensor documentation on Adafruit website

- Search for device by name or part number (e.g. VCNL4040 Proximity Sensor).
- Follow link at bottom of page from "Learn" section.
- Click through to "Python & CircuitPython" page for code examples.

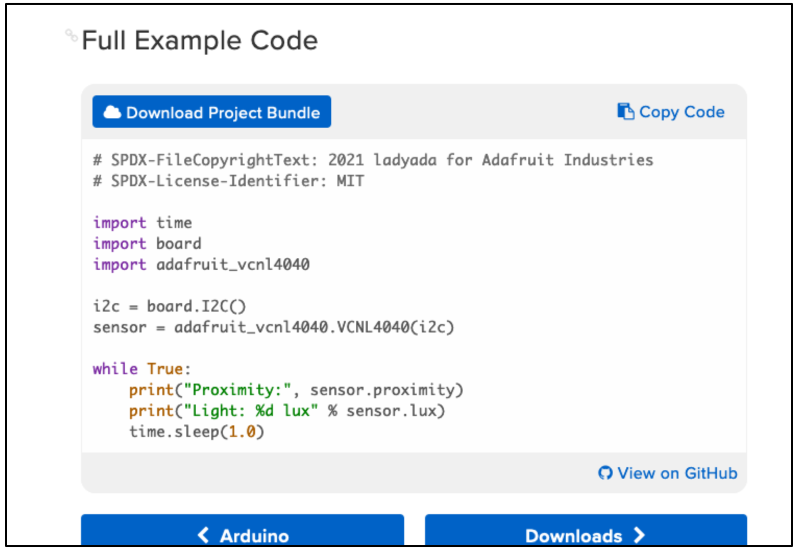


Learn

Primary Guide: Adafruit VCNL4040 Proximity Sensor

Sensing things up close and personal

May We Also Suggest



Full Example Code

```
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

import time
import board
import adafruit_vcnl4040

i2c = board.I2C()
sensor = adafruit_vcnl4040.VCNL4040(i2c)

while True:
    print("Proximity:", sensor.proximity)
    print("Light: %d lux" % sensor.lux)
    time.sleep(1.0)
```