website: vanderbi.lt/codegraf

QT Py RP2040 introduction

Presenter: Steve Baskauf
steve.baskauf@vanderbilt.edu

DISC DIGITAL SCHOLARSHIP AND COMMUNICATIONS

Jean & Alexander Heard LIBRARIES

CodeGraf landing page

- vanderbi.lt/codegraf

website: vanderbi.lt/codegraf

CircuitPython

DISC DIGITAL SCHOLARSHIP AND COMMUNICATIONS

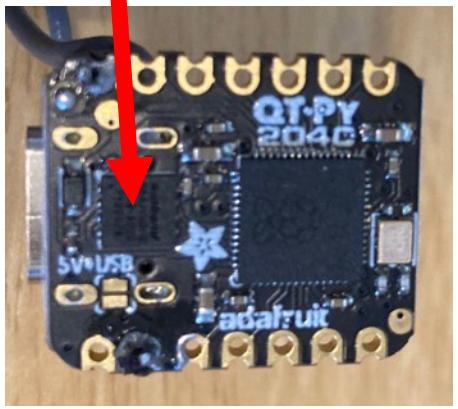
Jean & Alexander Heard LIBRARIES

Storage requirements

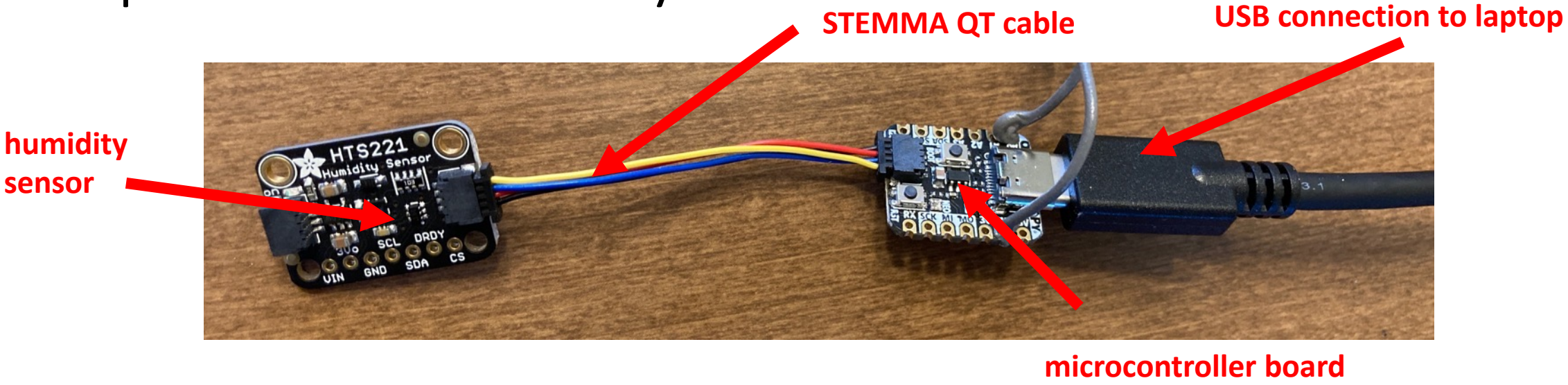
Standard cPython installation: 90 MB

Laptop: up to 1 TB (1 million MB)

QT Py RP2040: 8 MB flash memory

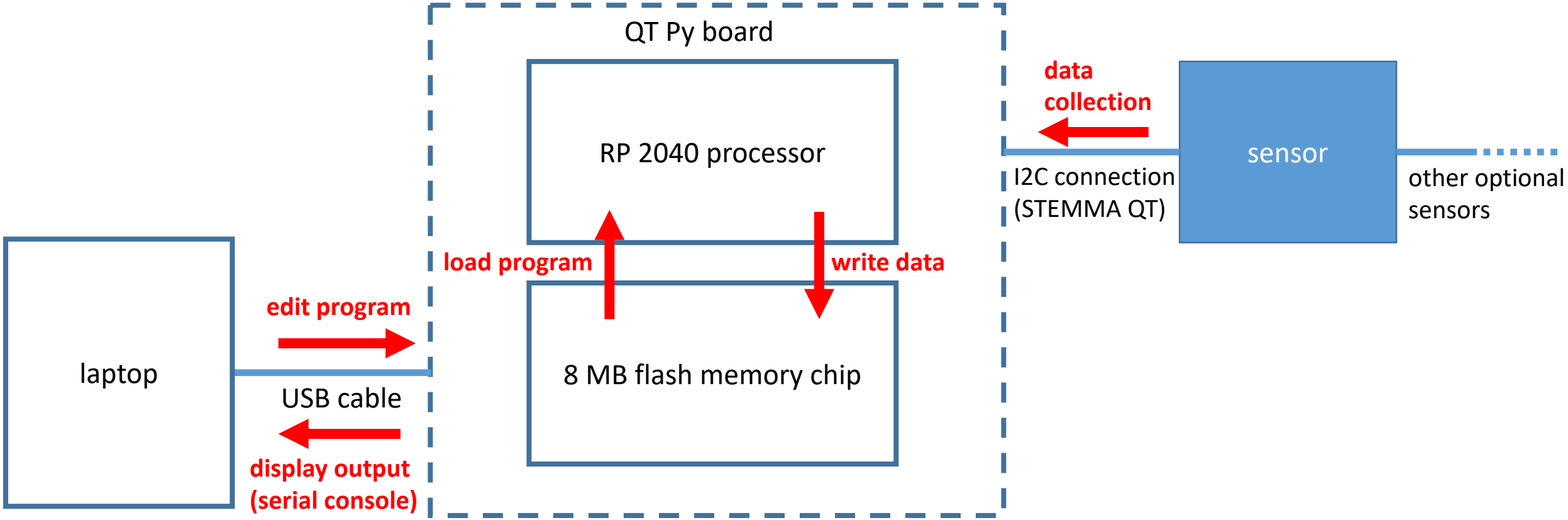


Aspects of CircuitPython



- 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

System architecture



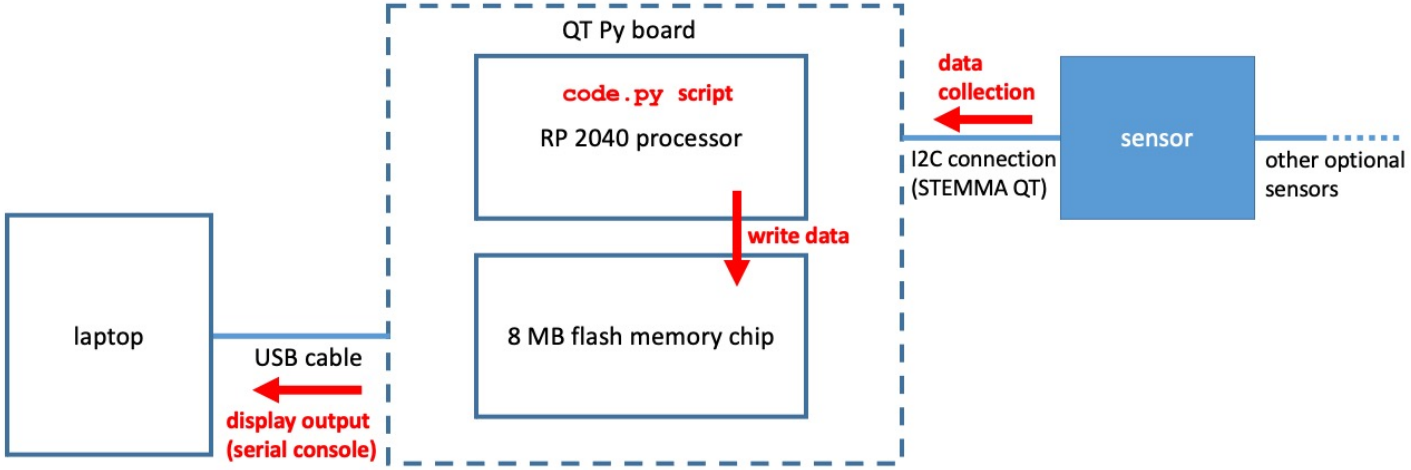
website: vanderbi.lt/codegraf

The `code.py` script

DISC DIGITAL SCHOLARSHIP AND COMMUNICATIONS

Jean & Alexander Heard LIBRARIES

What does the `code.py` script do?



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

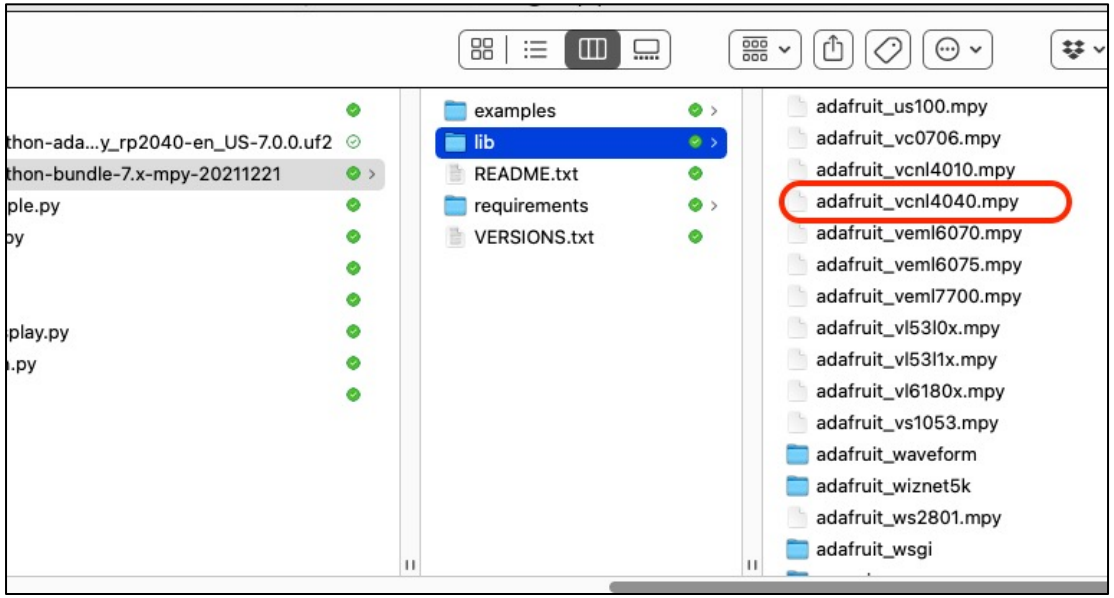
```
import time
import board
import adafruit_vcnl4040

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

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

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

module in Library Bundle (needs to be copied)



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

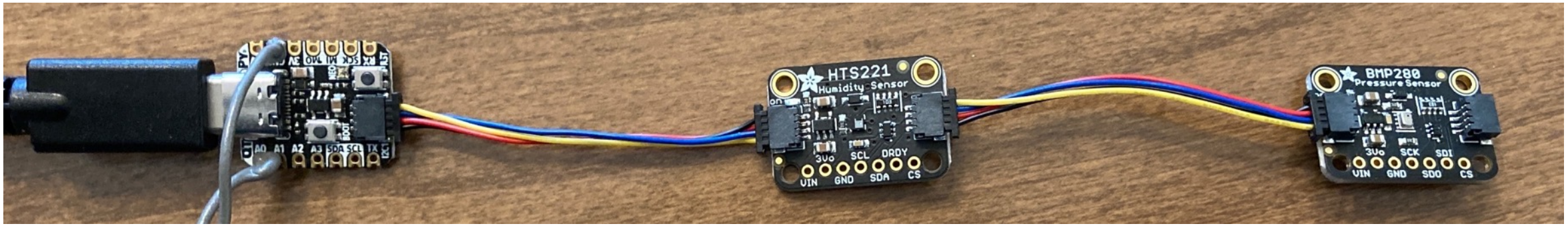
Connecting sensors

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).
- All devices share the same 2 wires and can be chained.
- Devices are identified by a unique address (127 possible).



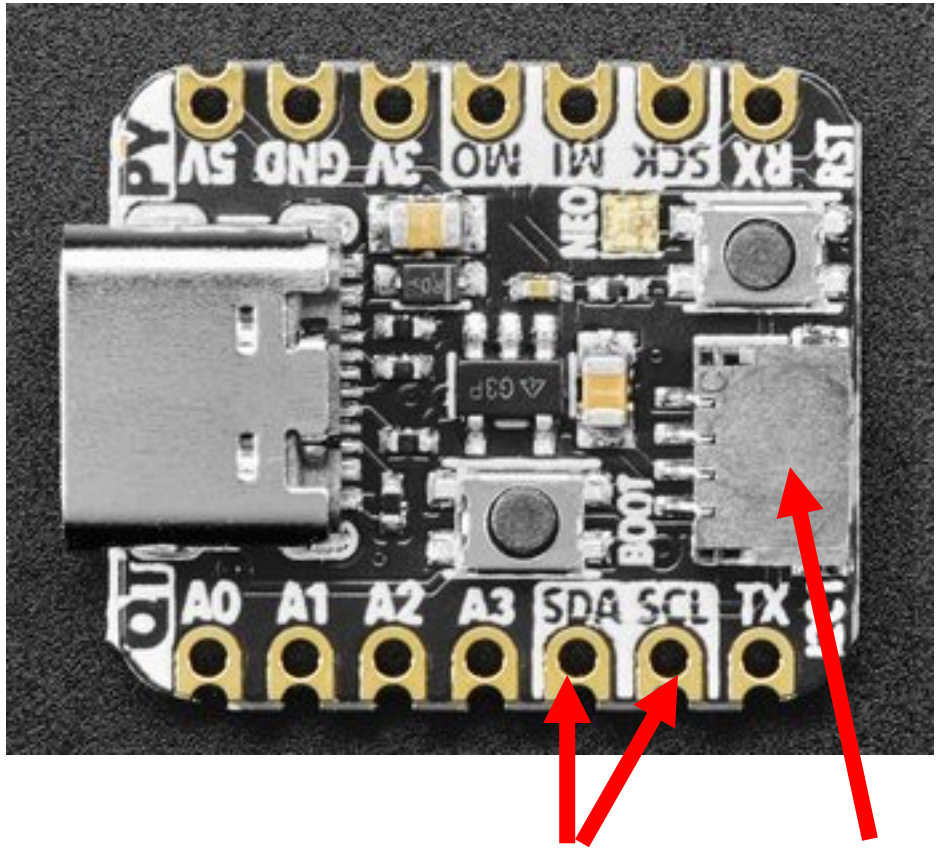
QT Py RP2040
using IC2 controller 1

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:

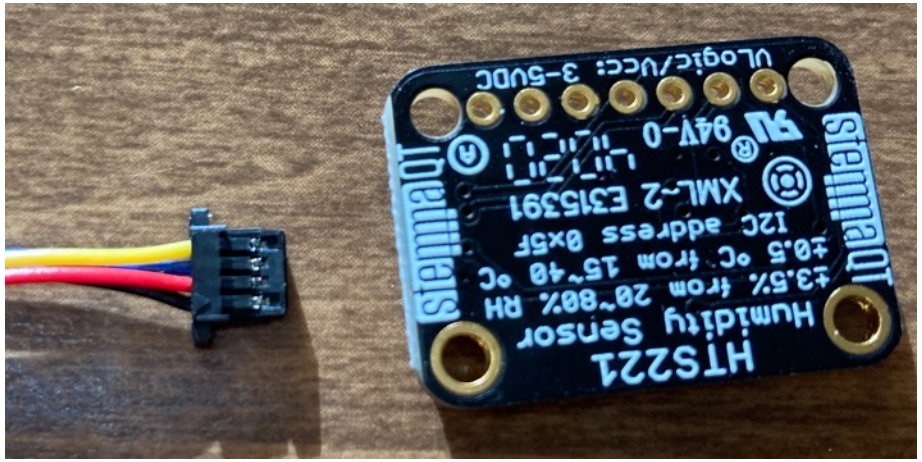
- **SDA** and **SCL** pads on board (**I2C0**)
- STEMMA QT connector (**I2C1**)

They are addressed in code as:

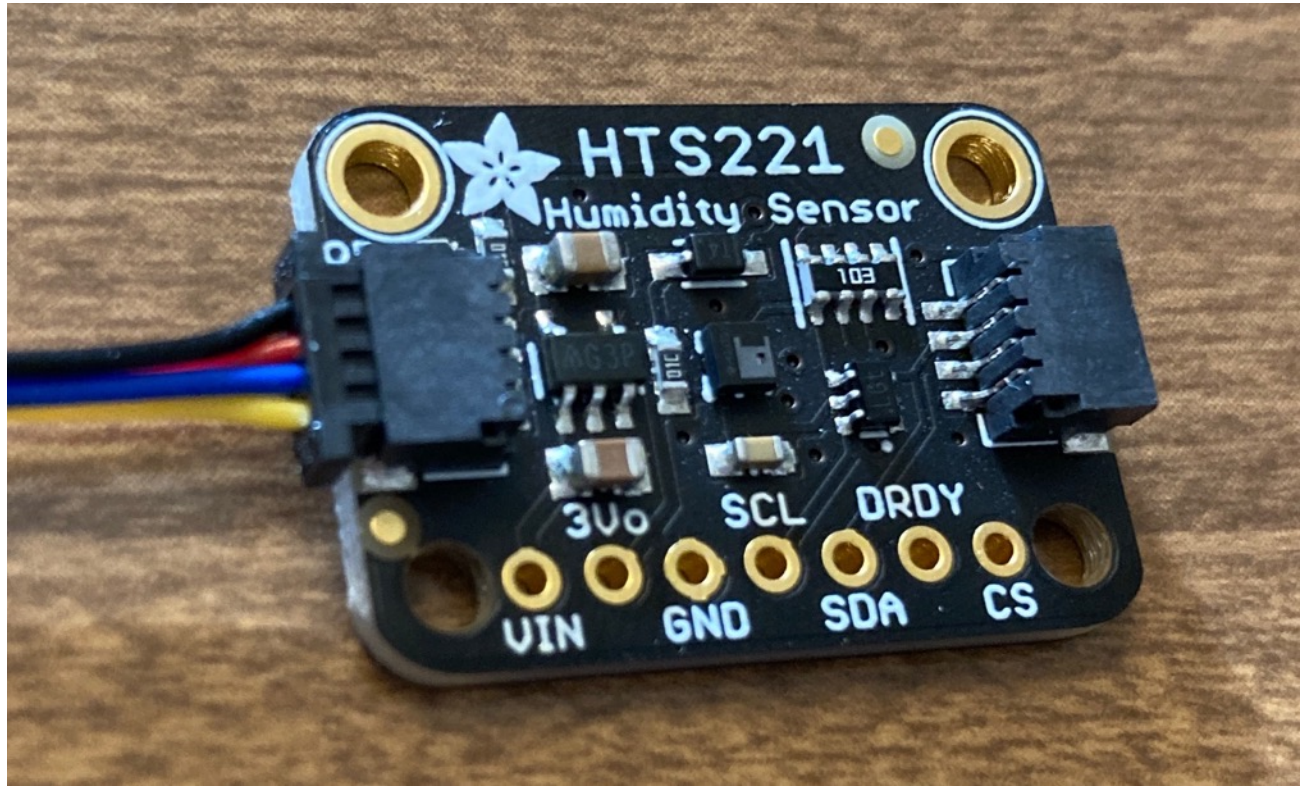
- **board.SCL** and **board.SDA** (**I2C0**)
- **board.SCL1** and **board.SDA1** (**I2C1**)

STEMMA QT connectors

top view



bottom view



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

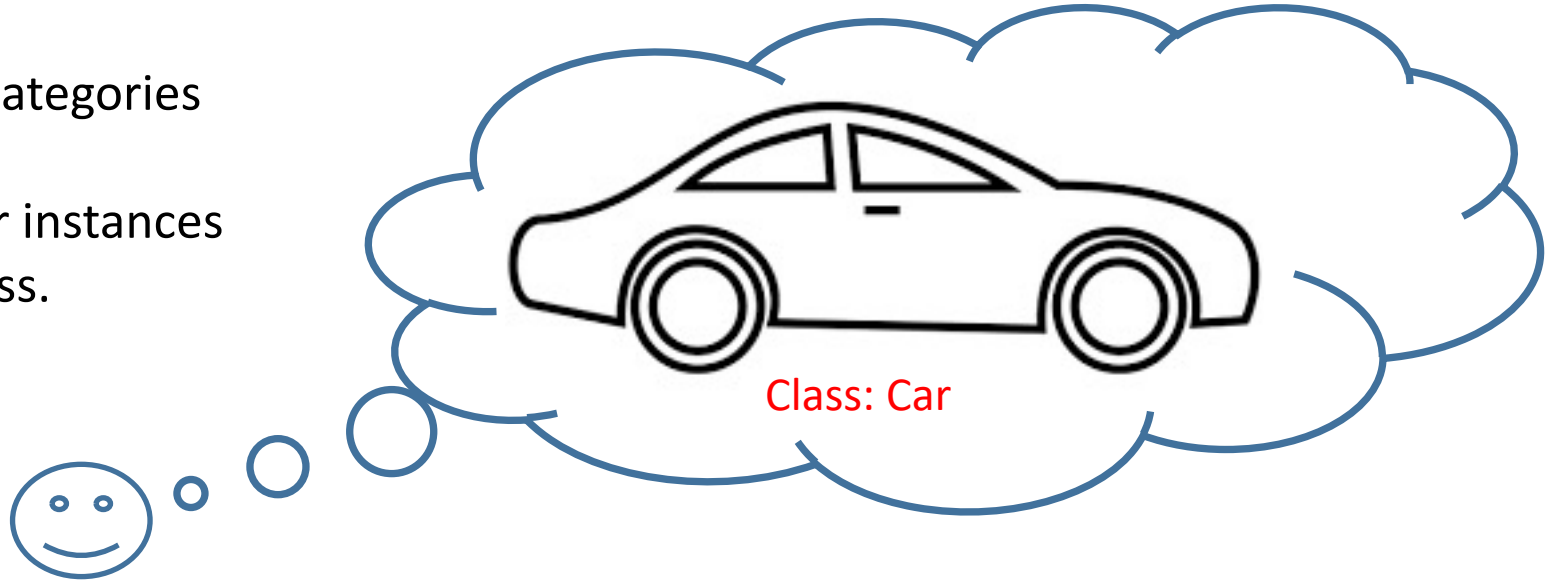
website: vanderbi.lt/codegraf

CircuitPython objects

Jean & Alexander Heard
LIBRARIES

classes and objects

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



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

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

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

normal while loop

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

infinite while loop

CircuitPython code example: indefinite number of measurements

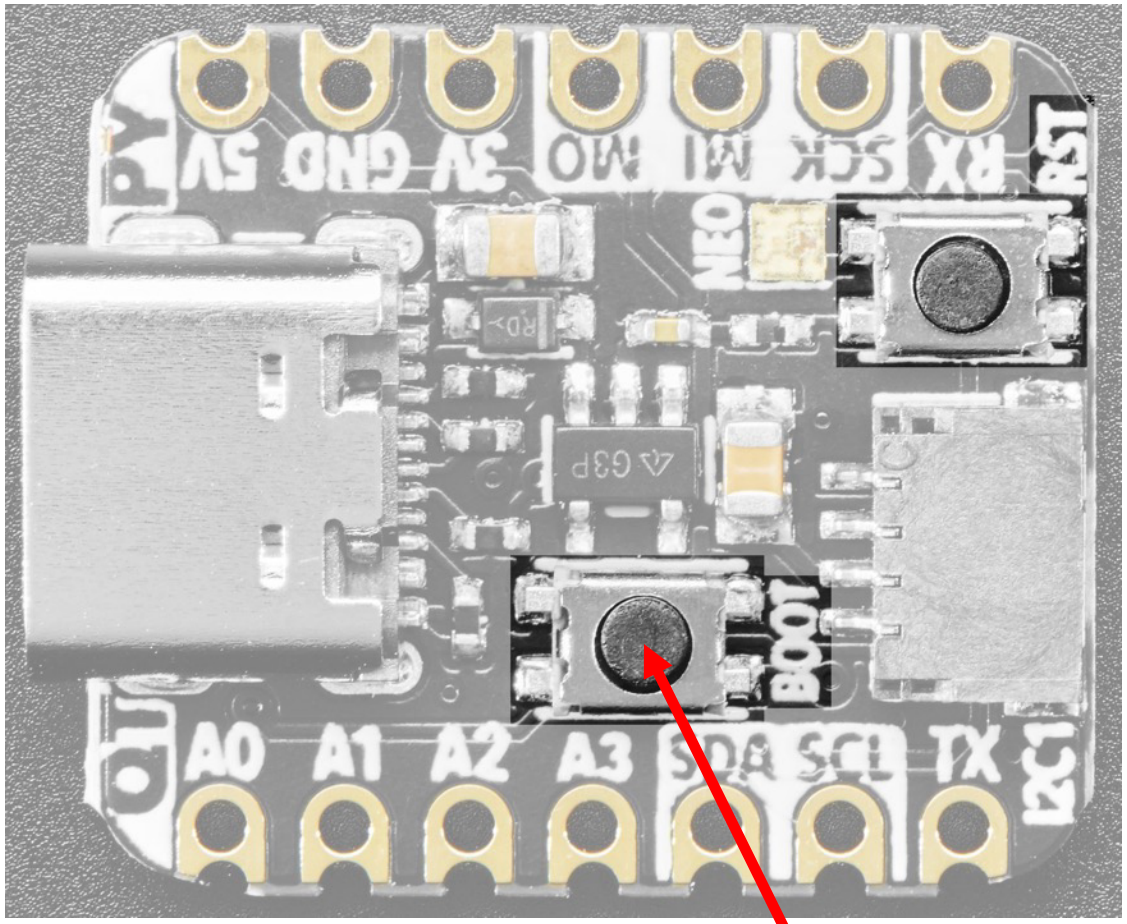
```
import time
import board
import busio
import adafruit_vcnl4040

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

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

website: vanderbi.lt/codegraf

Sources of information

DISC DIGITAL SCHOLARSHIP AND COMMUNICATIONS

Jean & Alexander Heard LIBRARIES

General reference documentation

- Circuit Python reference

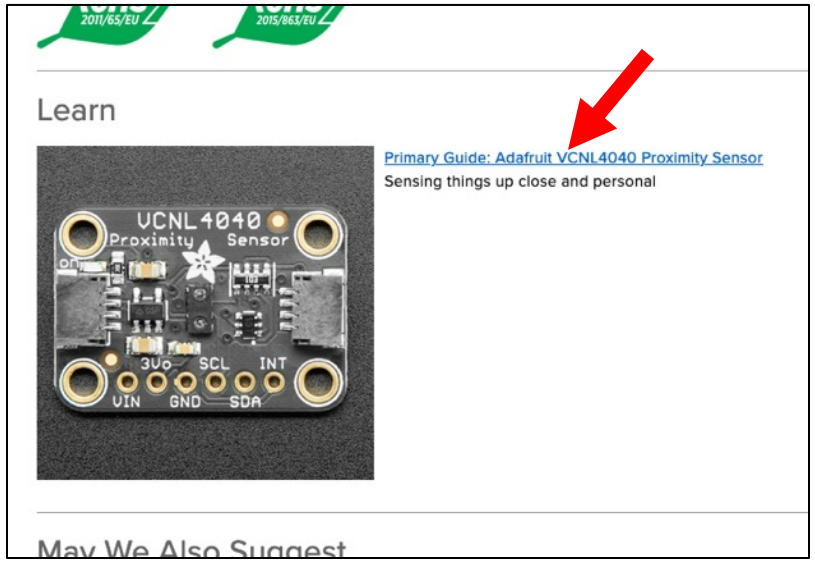
https://learn.adafruit.com/circuitpython-essentials/circuitpython-essentials

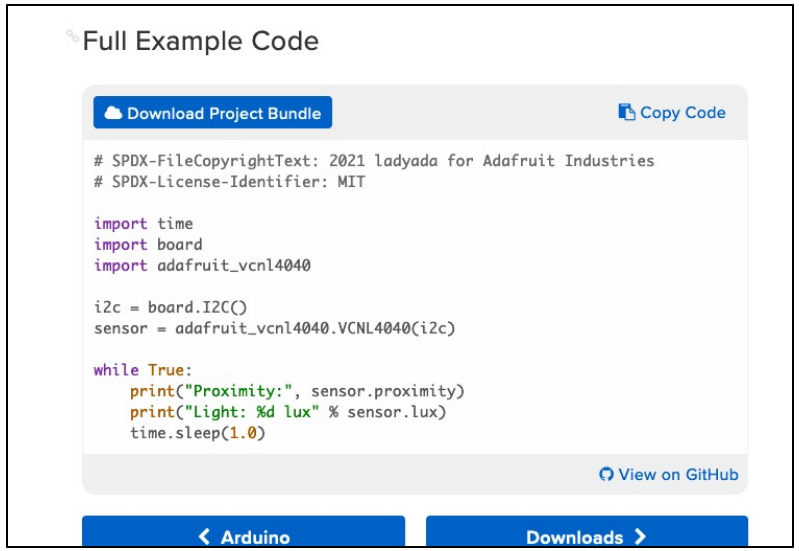
- QT Py RP2040 microcontroller

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

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.





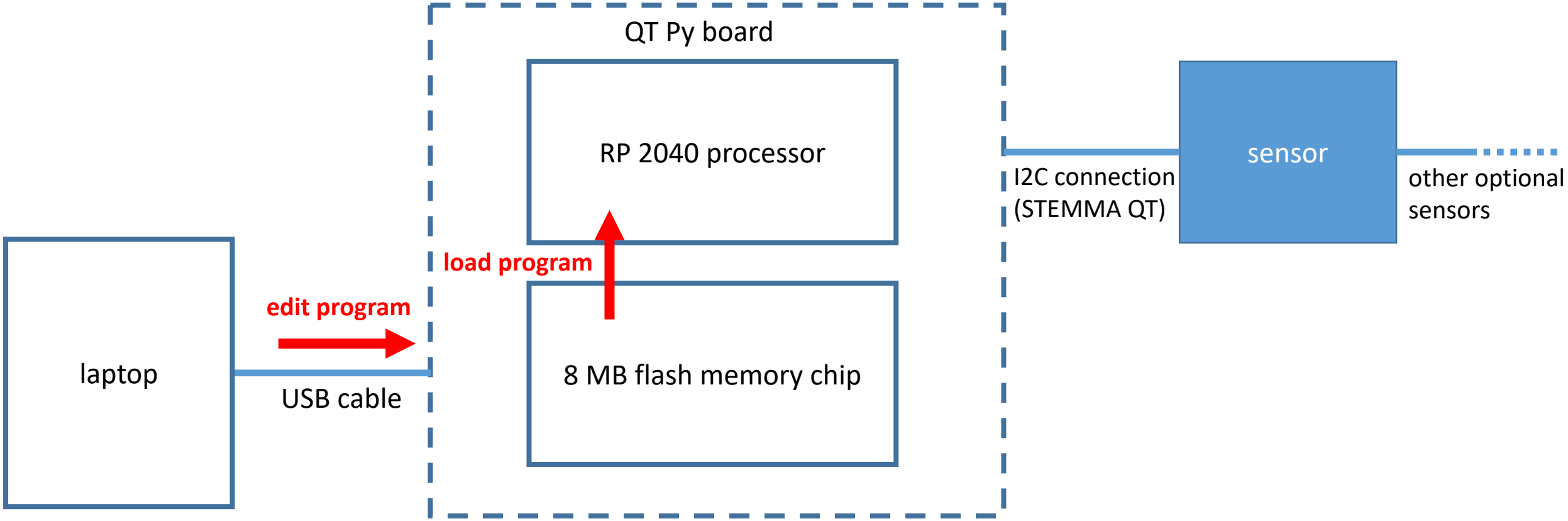
website: vanderbi.lt/codegraf

QT Py RP2040 memory limitations

DISC DIGITAL SCHOLARSHIP AND COMMUNICATIONS

Jean & Alexander Heard LIBRARIES

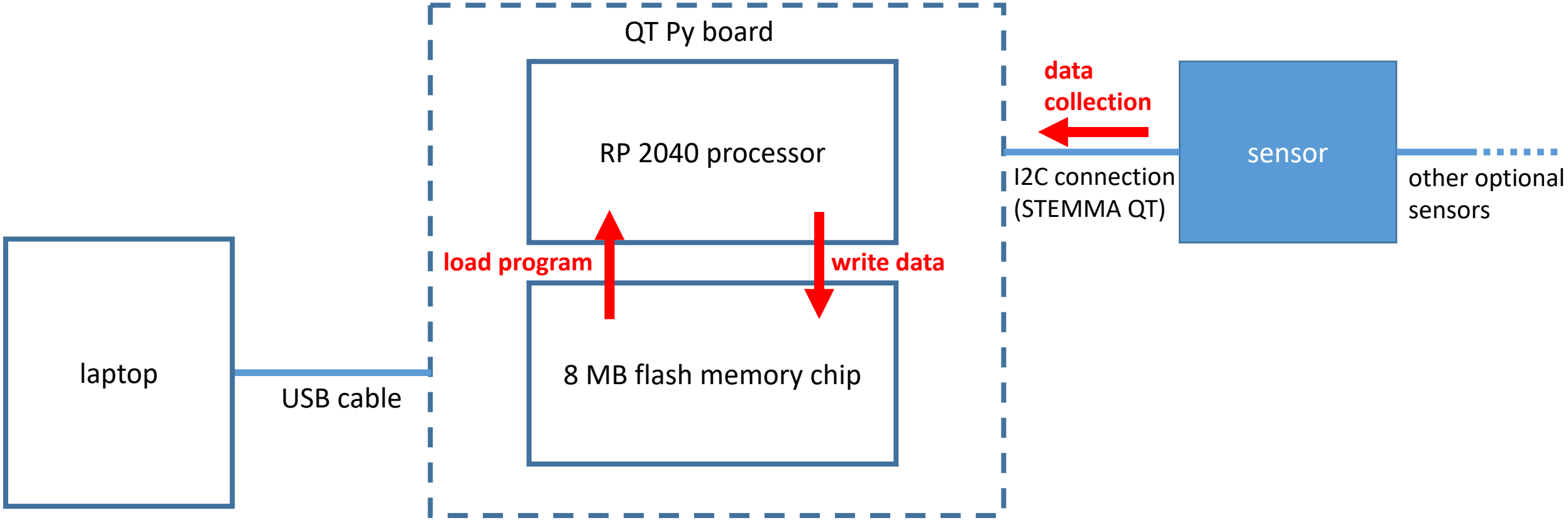
Memory in read only mode (default)



The processor can only read from the memory chip.

The laptap can write to the memory through the USB.

Memory in write only mode



The processor can write to the memory chip.

The laptap cannot access the memory, although it can monitor what's going on through the serial console.

try … **except** … code blocks

- Normally, when an error is thrown, code execution is terminated.
- If errors are handled, the script can continue to operate.
- A **try** code block specifies the code that might throw an error.
- An **except** code block specifies the code to be run if an error is thrown.

```
number_string = input('Enter a number: ')
number = int(number_string)
try:
    print(10/number)
except:
    print('Division by zero is undefined.')

```

CircuitPython error trapping code example

define code block where error might occur

trying to open a file in read-only mode will throw an error

indented code block for `try`

capture the type of error

error handling code

error code for read-only

```
try:
    with open("/temperature.txt", "w") as file_object:
        for count in range(10):
            print("Temp: %.2f C" % hts.temperature)
            temp = hts.temperature
            file_object.write(str(temp) + '\n')
            file_object.flush() # writes the file buffer after write
            time.sleep(delay_time)
except OSError as e:
    if e.args[0] == 30:
        print('read-only error')
    else:
        print('error', e.args[0])
```

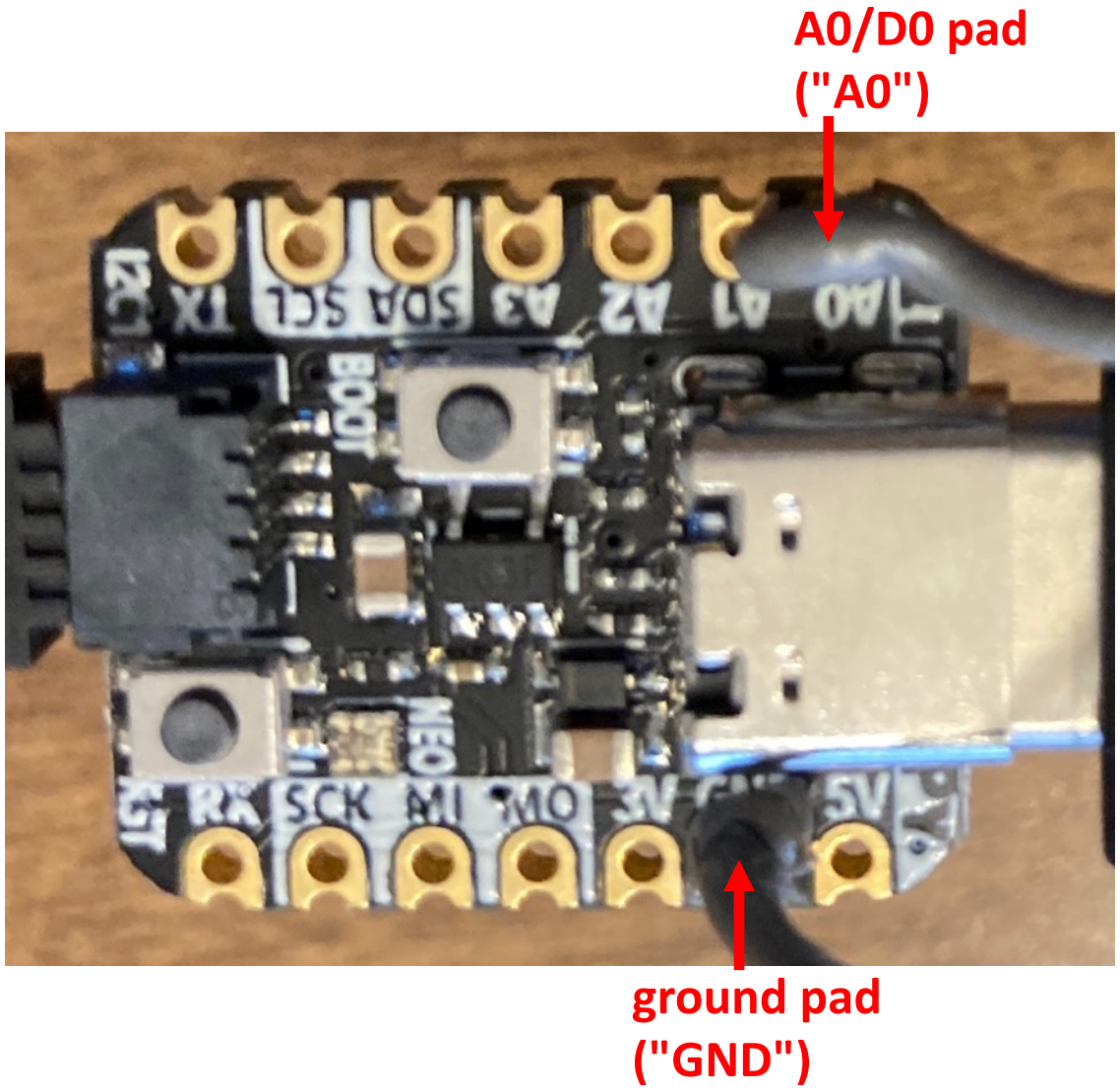
website: vanderbi.lt/codegraf

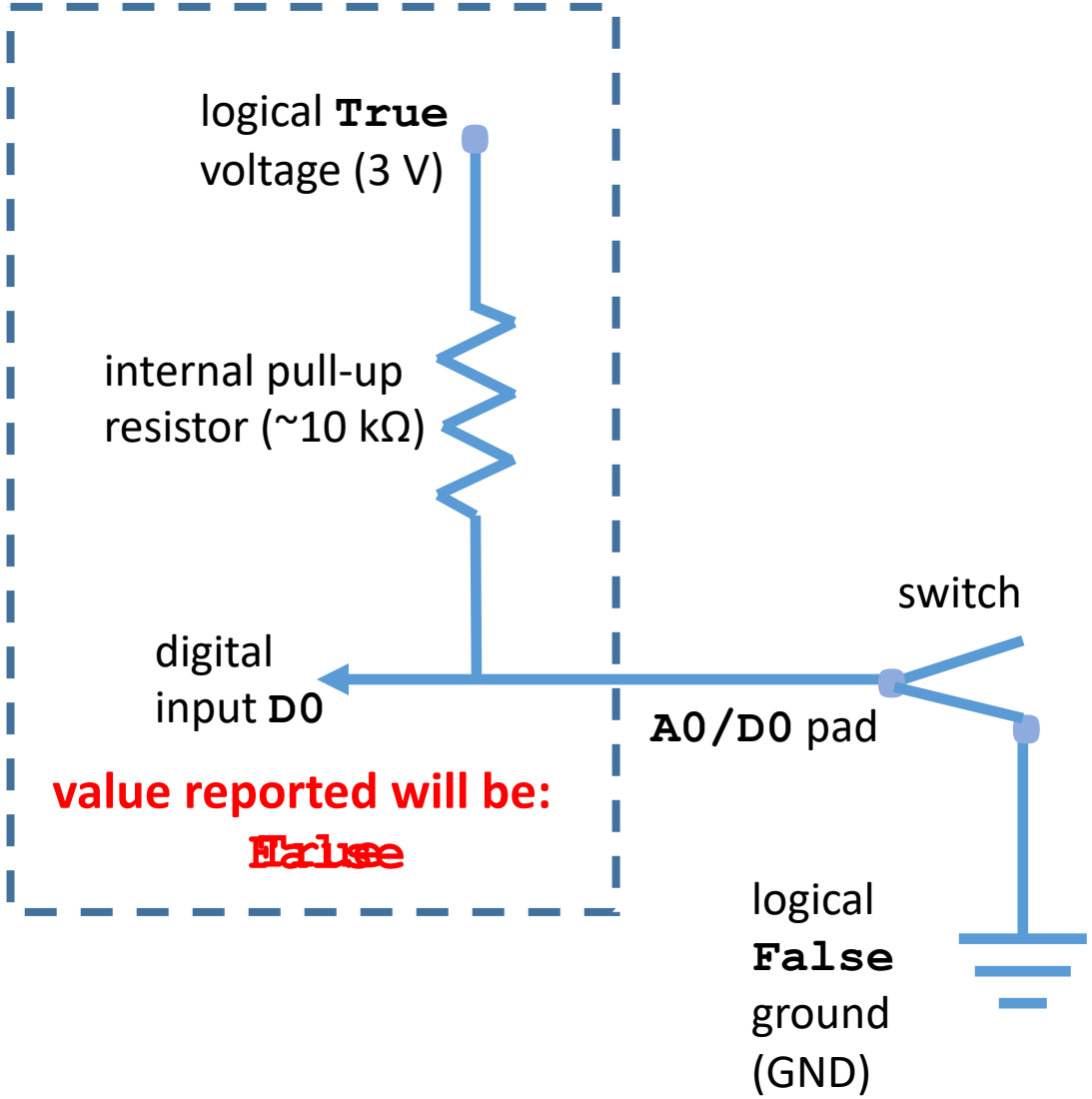
Controlling read and write state

DISC DIGITAL SCHOLARSHIP AND COMMUNICATIONS

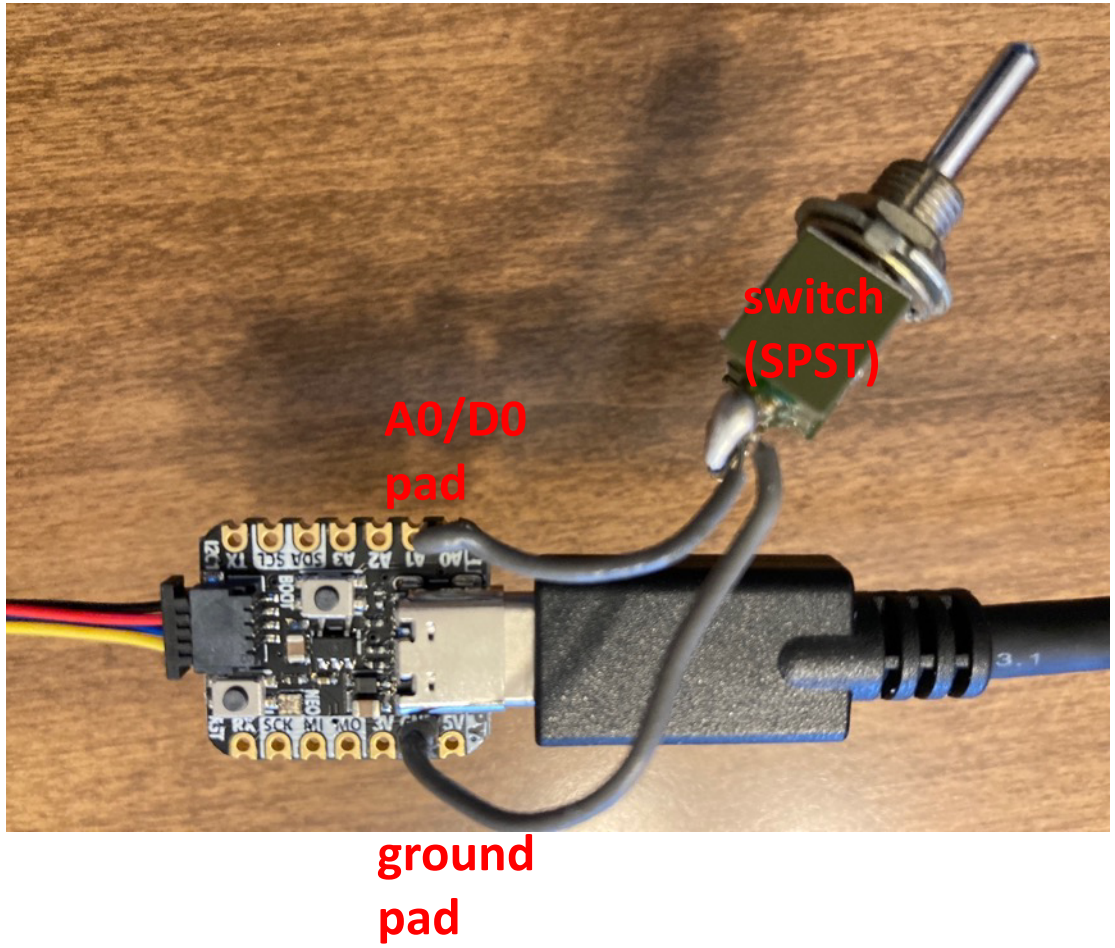
Jean & Alexander Heard LIBRARIES

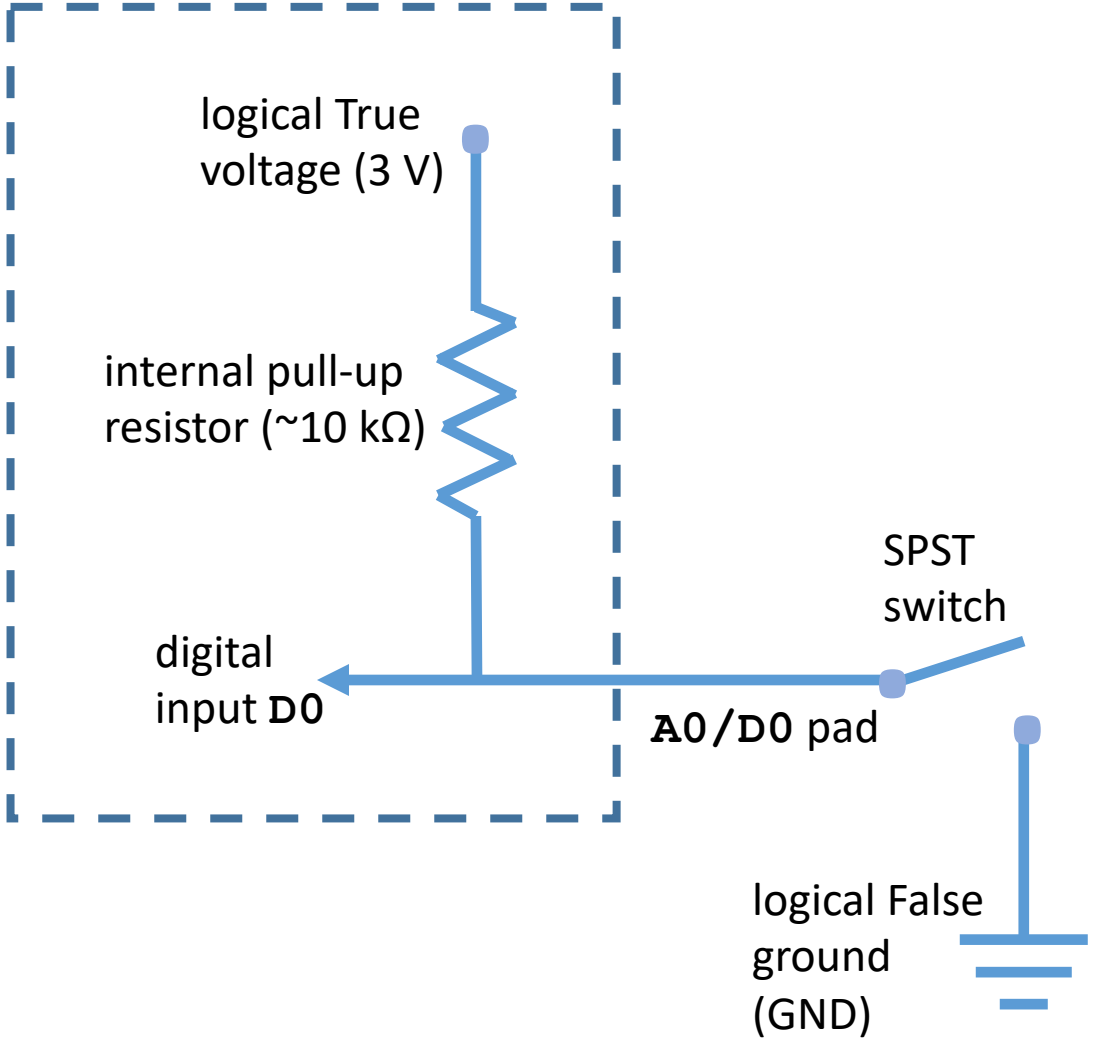
Controlling the input voltage of digital input



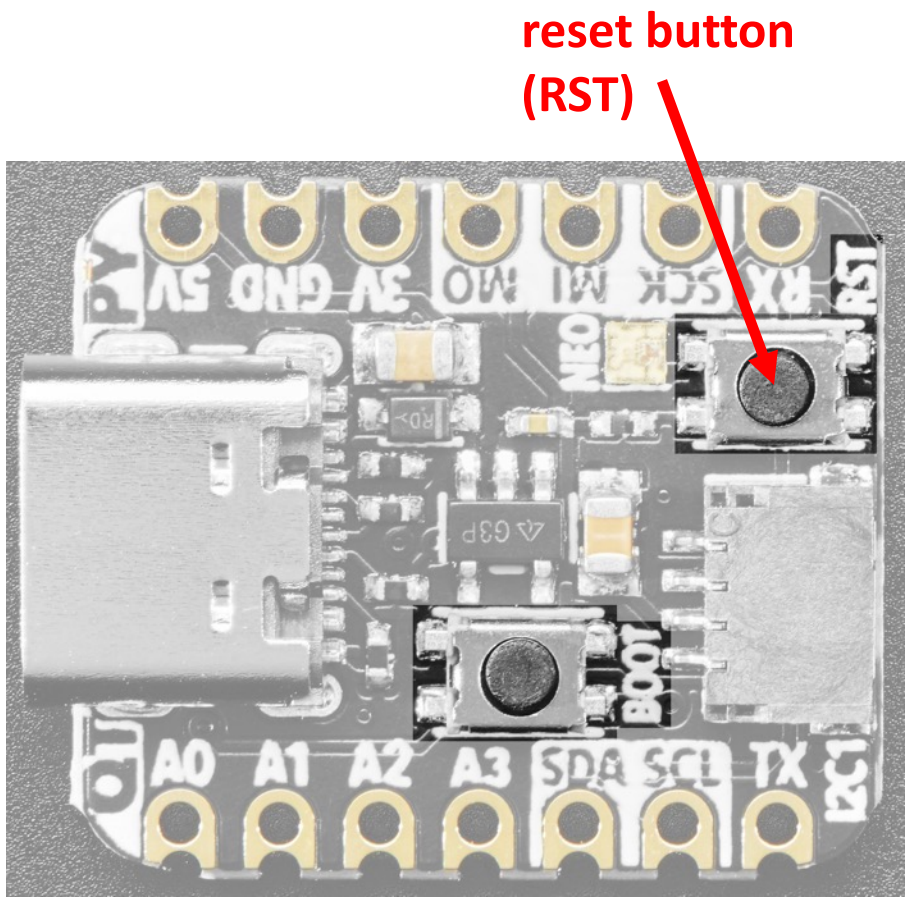


External switch wiring (soldering required)





What does the `boot.py` script do?



The **`boot.py`** script is a special file in CircuitPython

It is executed when:

- the board is powered up.
- when you do a "hard reset" by pushing the reset button on the board

It does not run of a "soft reset" from the serial console.

It runs before **`code.py`**

`boot.py` script to control read/write

```
import board
import digitalio
import storage

switch = digitalio.DigitalInOut(board.D0)
switch.direction = digitalio.Direction.INPUT
switch.pull = digitalio.Pull.UP

# Connecting D0 to ground makes switch.value False
storage.remount("/", switch.value)
```

Set the mode of A0/D0 ADC pin to digital by instatiating a `DigitalInOut` object using the D0 pin object

Set the direction of A0/D0 ADC pin to input

Set up internal resistor to pull-up configuration

If second argument of `.remount()` is `True`, memory is read-only
If second argument of `.remount()` is `False`, CircuitPython can write