MicroPython Introduction (cont.)

MCHE 201 – Spring 2019

Dr. Joshua Vaughan

Rougeou 225

joshua.vaughan@louisiana.edu

@Doc_Vaughan

Where can I find help?

- Full – `http://docs.micropython.org/en/latest/pyboard/`
- Quick Ref – `http://docs.micropython.org/en/latest/pyboard/pyboard/quickref.html`
- REPL specific – `http://docs.micropython.org/en/latest/pyboard/reference/repl.html`
- Links are continually added to class webpage
- If you don’t remember the syntax, look it up
- If you *really* don't understand, *ASK!*

In-class Exercise 3

- Turn on the green LED when the button is pressed

- Turn on the red LED when it is *not* pressed

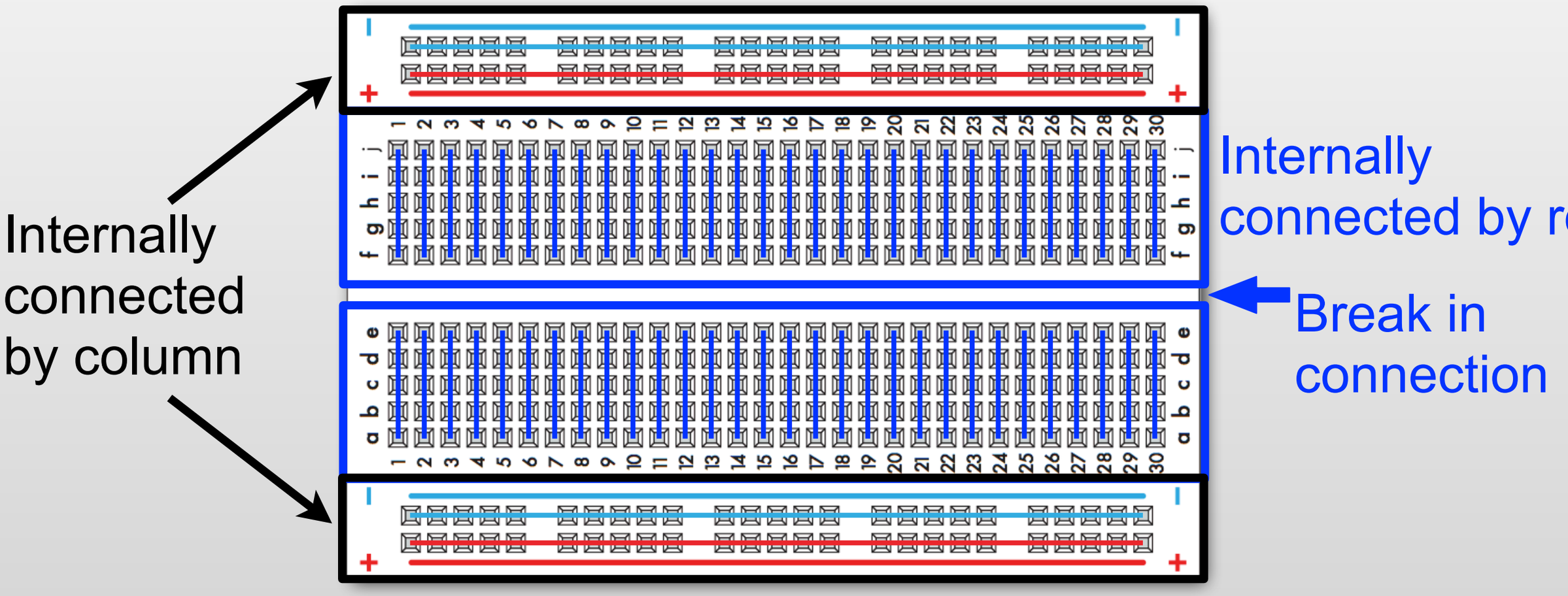
There are many ways to do this. A script with some is at:

https://github.com/DocVaughan/MCHE201---Intro-to-Eng-Design/tree/Spring-2019/MicroPython/MCHE201%20-%20In-class%20Exercise%203%20-%2003:12:19

Exercise 3 – One Solution

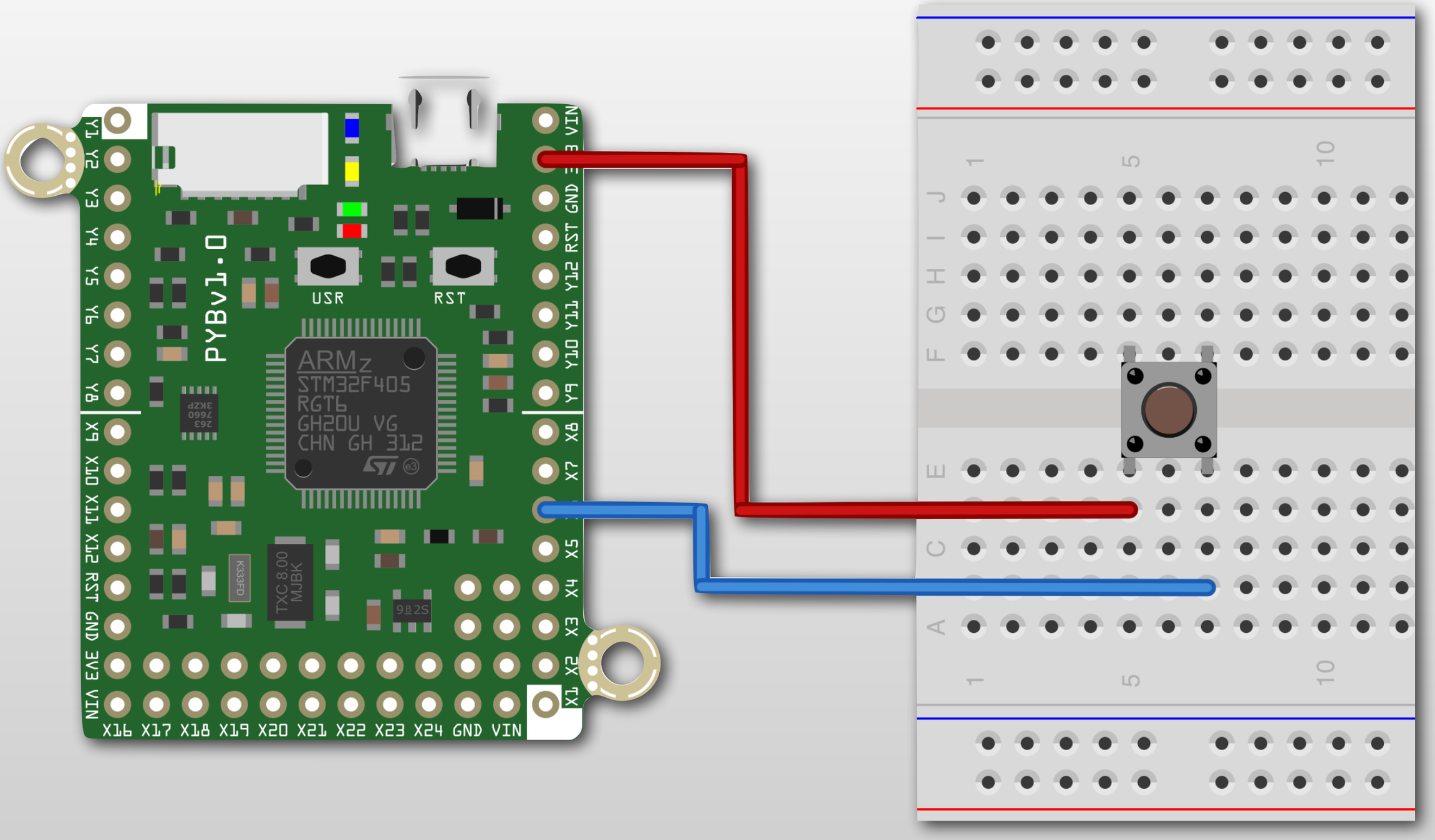
```
import pyb  # import the pyboard module
import time # import the time module

# Assign variable names for the LEDs
RED_LED = pyb.LED(1)
GREEN_LED = pyb.LED(2)

# Assign the Switch object for the onboard button to button
button = pyb.Switch()

# The condition for this while is always True, so it runs forever
while (True):
    if (button()): # button() is True if the button is pressed
        print("Button Pressed!")
        GREEN_LED.on()
        RED_LED.off()
    else:
        print("Button not pressed.")
        GREEN_LED.off()
        RED_LED.on()

   time.sleep_ms(100) # Sleep 100ms between readings
```

Basic Breadboard Setup



The Pushbuttons

Pushbutton Hardware Setup



MicroPython Code for Pushbutton

```
import pyb  # import the pyboard module
import time # import the time module

# Assign the input pin to variable input_pin
# We set it up as an input with a pulldown resistor
input_pin = pyb.Pin("X6", pyb.Pin.IN, pull=pyb.Pin.PULL_DOWN)

# This will loop forever, checking the button every 100ms
while (True):
    # read the state of the input
    input_state = input_pin.value()

    if (input_state):
        print("The input is high (on).")
    else:
        print("The input is low (off).")

    # Sleep 100 milliseconds (0.1s)
    time.sleep_ms(100)
```

In-class Exercise 4

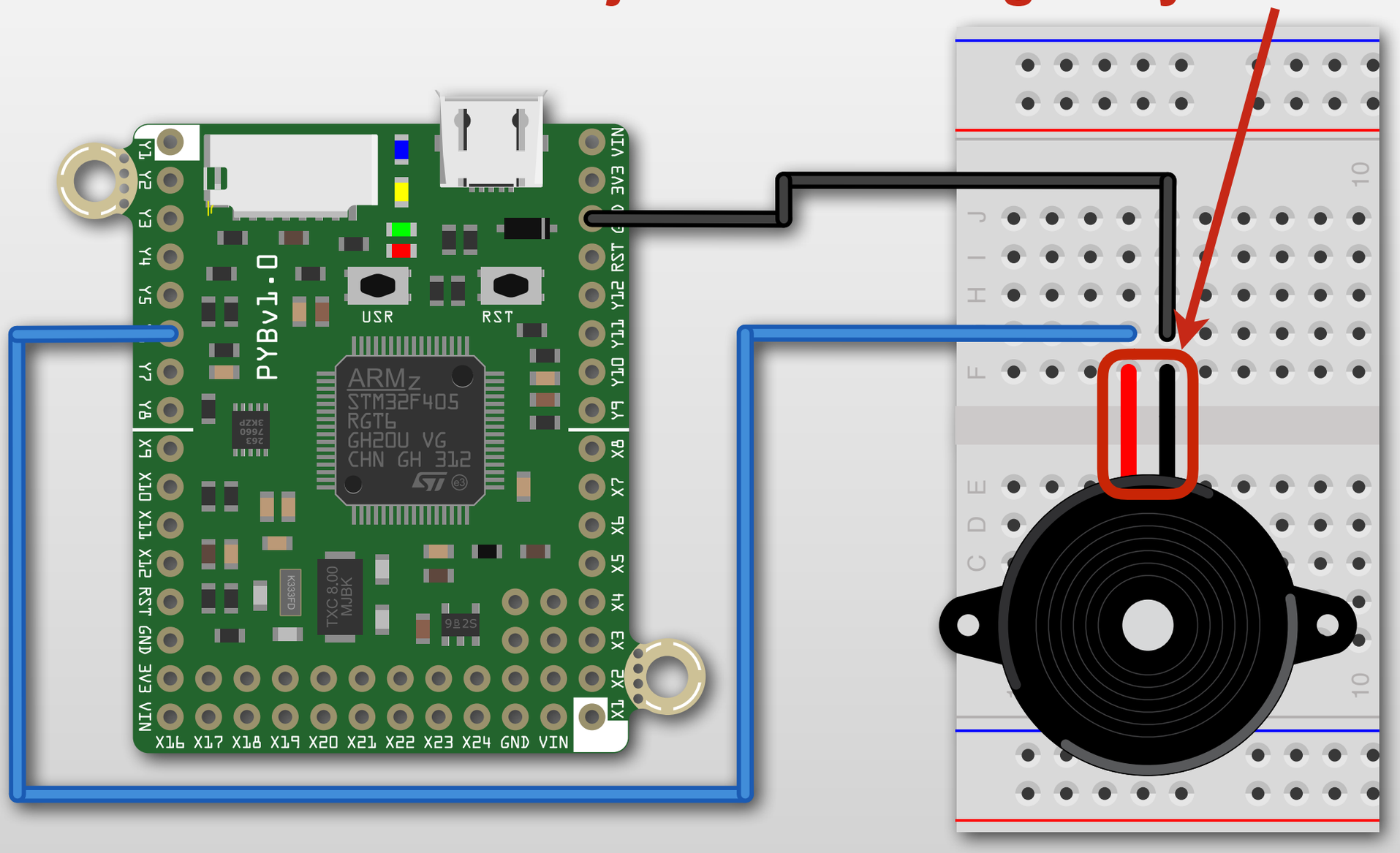
- When the external pushbutton is pressed, turn on one of the onboard LEDs. When it is not pressed, the LED should be off.

- *Hint:* The logic will be identical to In-class Exercise 3. Only the setup and method to read the button need to change.

The Buzzer

Buzzer Hardware Setup

It looks different, but these are just the two legs of your buzzer.



Buzzer MicroPython Code

https://github.com/DocVaughan/MCHE201---Intro-to-Eng-Design/blob/Spring-2019/MicroPython/pyboard%20buzzer/main.py

Try running it on your board.

In your kit…

- **Potentiometer** - changes resistance based on rotation

- **Soft Potentiometer** - changes resistance based on where it's squeezed

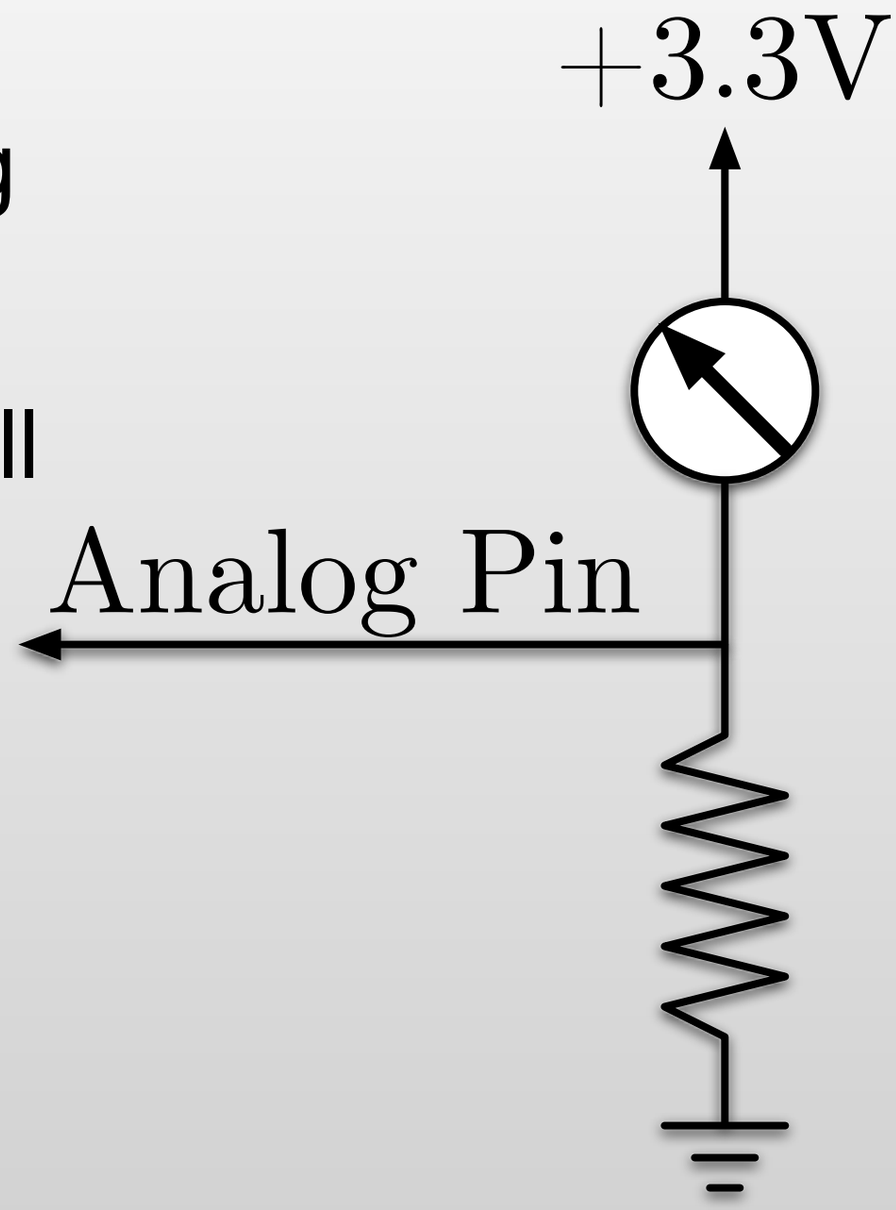
- **Flex Sensor** - changes resistance based on how far it's bent

- **Force Sensitive Resistor (FSR)** - changes resistance based on how hard it's squeezed

Measuring Changes in Resistance

Measuring Changes in Resistance

- Make the component part of a voltage divider
- Measure voltage change resulting from resistance change
- To measure an analog signal, we'll need an Analog-to-Digital Converter (ADC)
- The pyboard ADCs are 12 bit, meaning that they return a value of 4095 for 3.3V and a value of 0 for 0V



On the pyboard

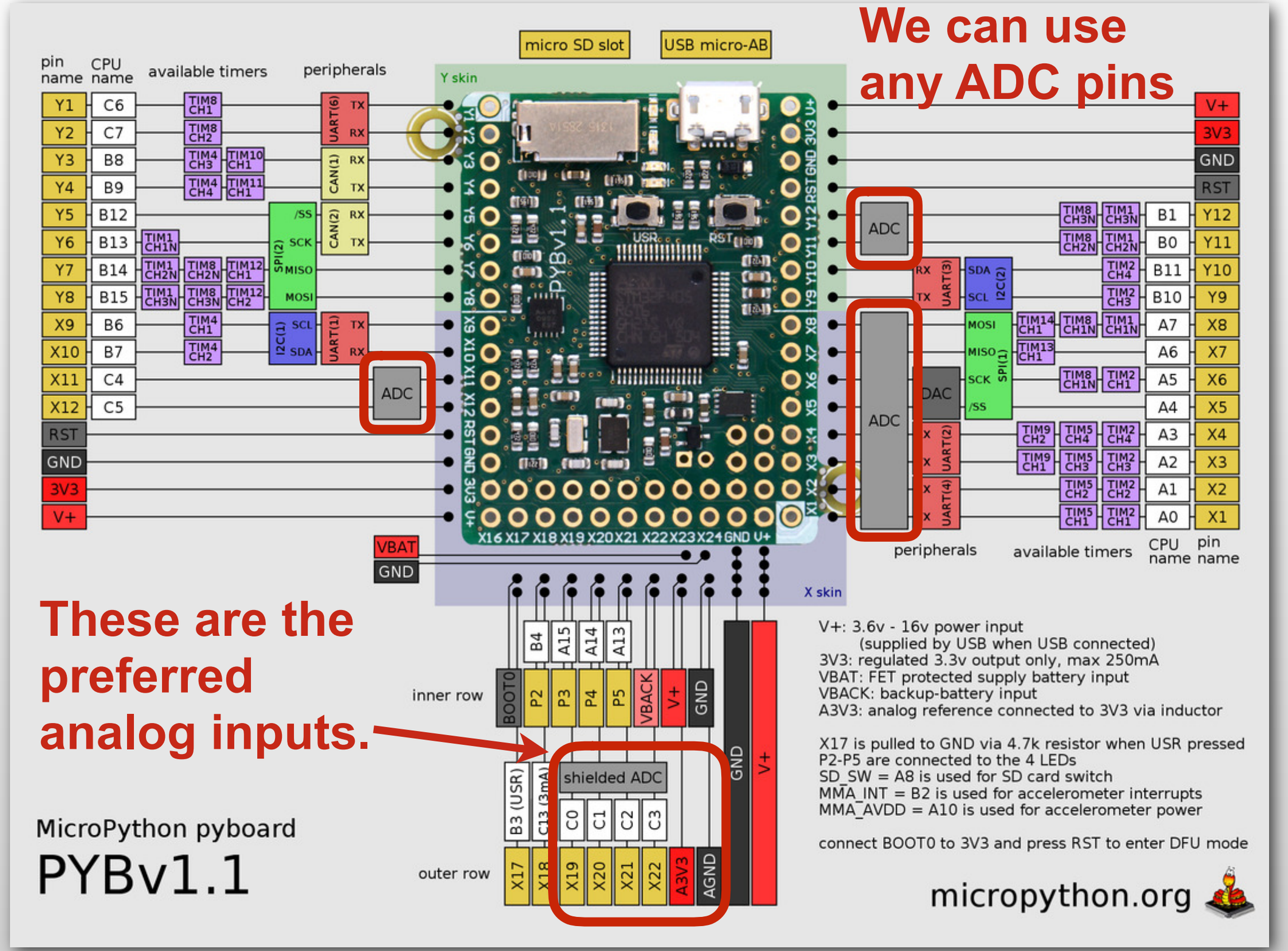
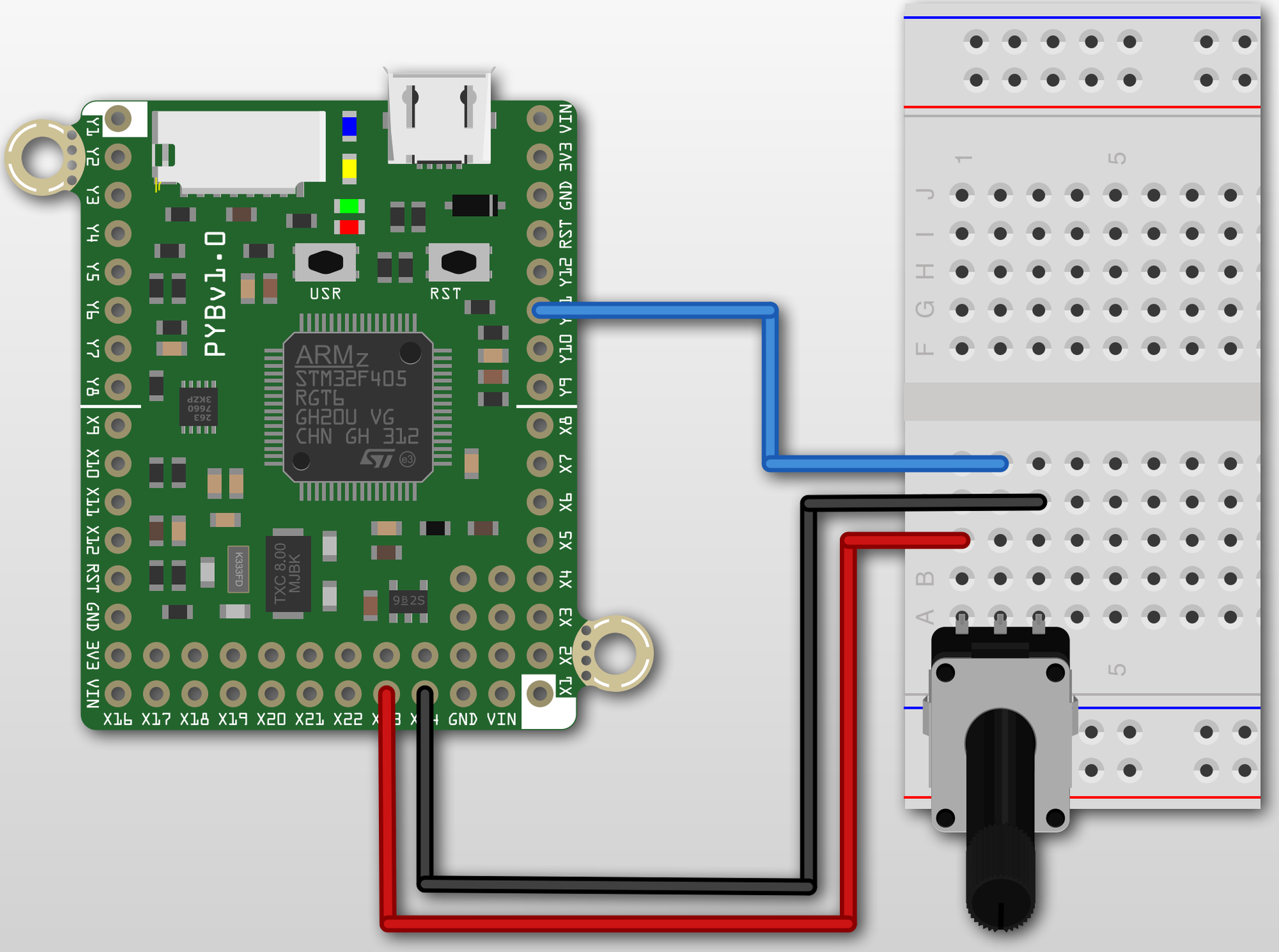


Image via `micropython.org`

Potentiometers

Potentiometer Hardware Setup



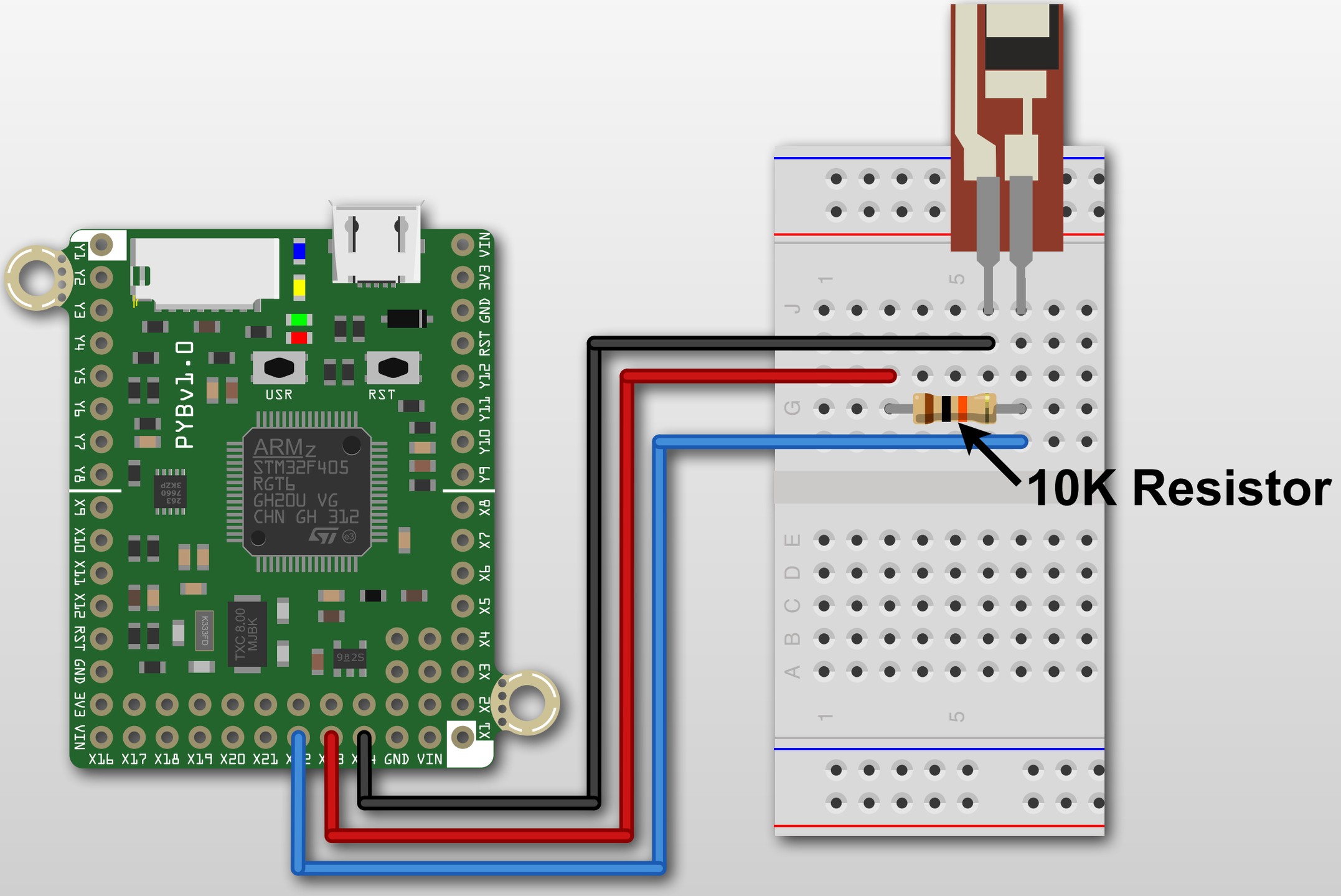
Use the small blue potentiometer from your kit. It's casually called a *trimpot*.

MicroPython Code for Potentiometer

```
import pyb          # import the pyboard module
import time         # import the time module

# Set up the analog-to-digital converter
adc = pyb.ADC(pyb.Pin("Y11"))

# Now read the pot every 500ms, forever
while (True):
    # Read the value of the potentiometer.
    # It should be in the range 0-4095
    pot_value = adc.read()

    # We can convert the value to the voltage
    # 0=0V and 4095=3.3V
    voltage = 3.3 / 4095 * pot_value

    # print out the values, nicely formatted
    print("The ADC value is {} = {:.2f}V.”.format(pot_value, voltage))

    # sleep for 500ms
    time.sleep_ms(500)
```

The Flex Sensor

Flex Sensor Hardware Setup



MicroPython Code for Flex Sensor

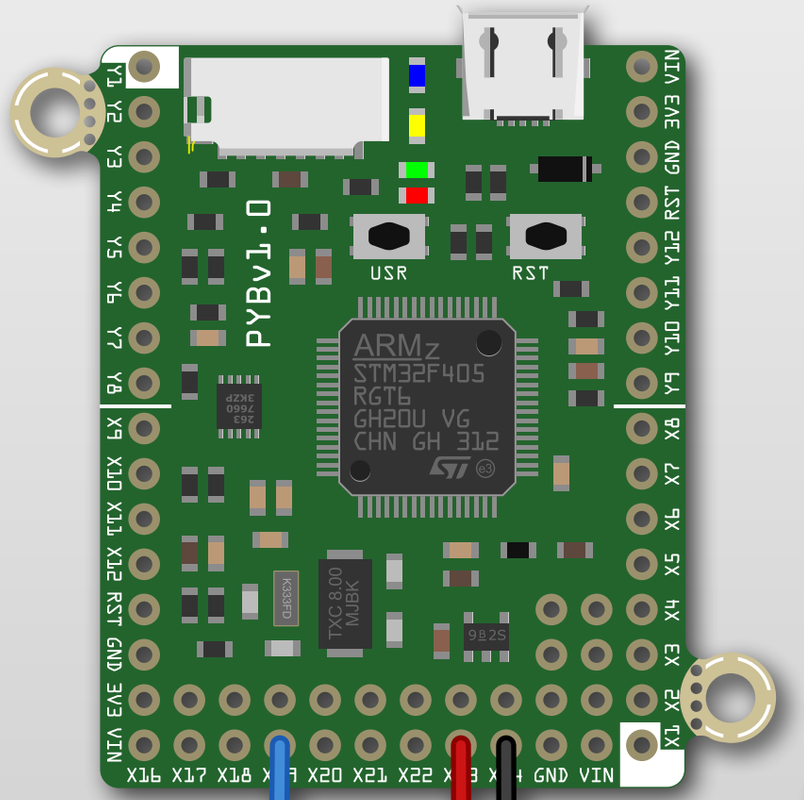
```
# Set up the analog-to-digital converter
adc = pyb.ADC(pyb.Pin("X22"))

# Now read the flex sensor every 500ms, forever
while (True):
    # Read the value of the flex sensor. Should be 0-4095
    flex_value = adc.read()

    # We can convert the value to the voltage we are reading.
    # 0=0V and 4095=3.3V
    voltage = 3.3 / 4095 * flex_value

    # print out the values, nicely formatted
    print("ADC:     {:5d}".format(flex_value))
    print("Voltage: {:5.2f}".format(voltage))

    # sleep for 500ms
    time.sleep_ms(500)
```

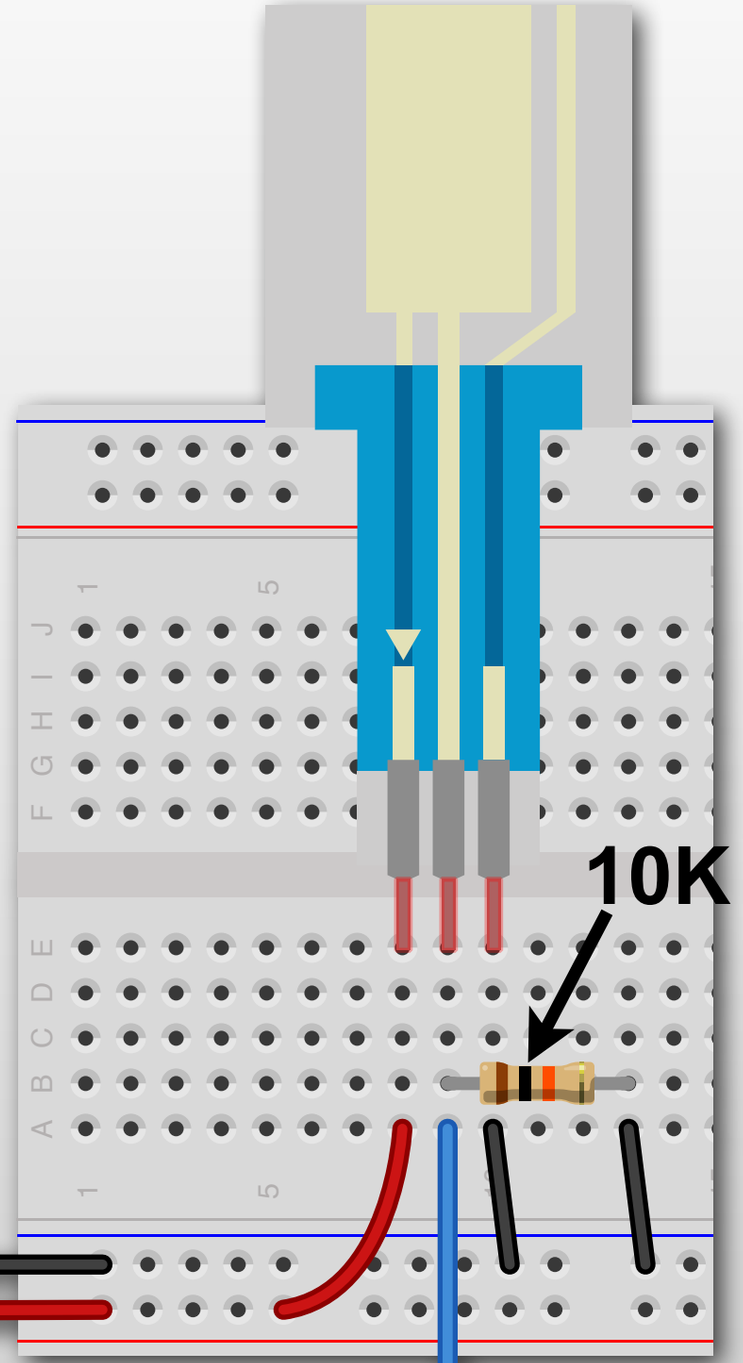
In-class Exercise 5

- Divide the flex sensor range into four
- Turn on the same number of LEDs as the “range number” of the current state of the flex sensor.
- In other words, when the sensor is not bent, no LEDs should be on. When it’s bent a little, one LED should turn on. When it’s bent to its maximum, all 4 LEDs should be on.

The Soft Potentiometer

Soft Potentiometer Hardware Setup

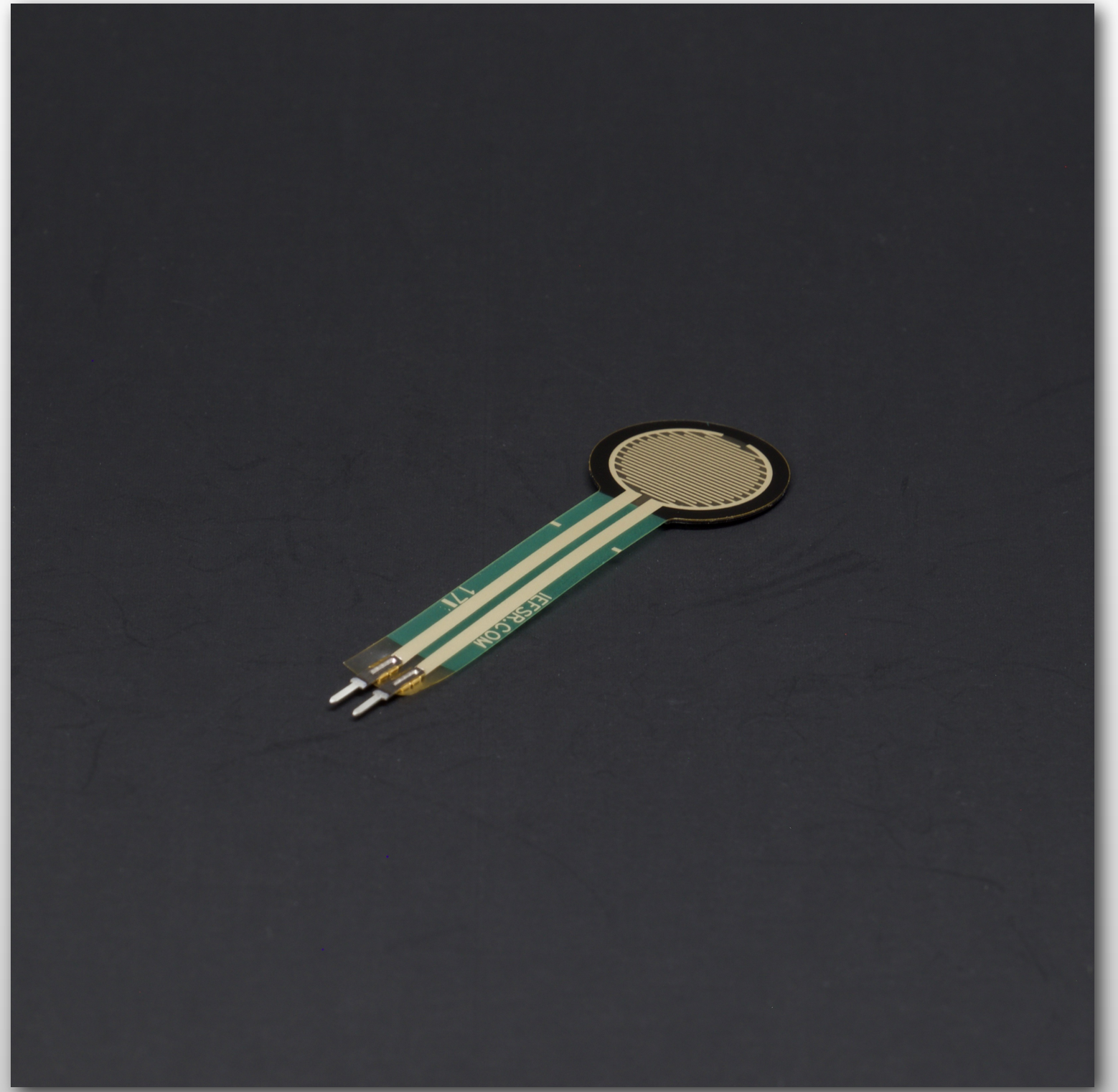




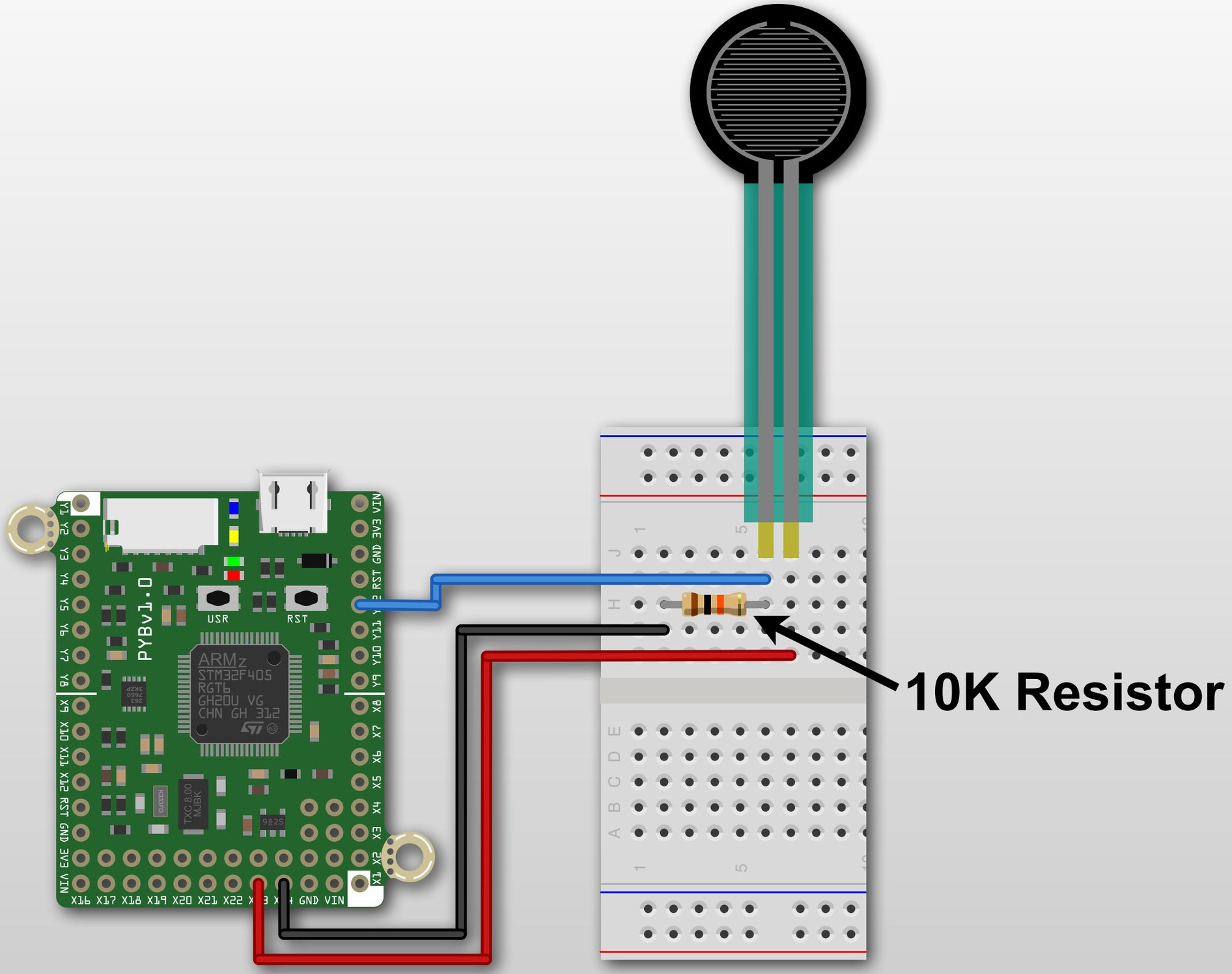
Soft Pot MicroPython Code

```
import pyb       # import the pyboard module
import time      # import the time module

# Set up the analog-to-digital converter
adc = pyb.ADC(pyb.Pin("X19"))

# Now read the soft pot every 500ms, forever
while (True):
    # Read the value of the soft pot. It should be in the range 0-4095
    pot_value = adc.read()

    if pot_value == 0:
        print("It looks like you're not pressing the pot.\n)

    else:
        if pot_value > 3500:
            touch_location = "top"
        elif pot_value < 500:
            touch_location = "bottom"
        else:
            touch_location = "middle"

        print("You're pressing the {} of the pot.".format(touch_location))

    # sleep for 500ms before looping to read the sensor again
    time.sleep_ms(500)
```

The Force Sensitive Resistor (FSR)



Force Sensitive Resistor (FSR)



FSR MicroPython Code

```
# Set up the analog-to-digital converter
adc = pyb.ADC(pyb.Pin("Y12"))

# Now read the FSR every 500ms, forever
while (True):
    # Read the value of the FSR. It should be in the range 0-4095
    fsr_value = adc.read()

    # We can convert the value to the voltage we are reading.
    # 0=0V and 4095=3.3V
    voltage = 3.3 / 4095 * fsr_value

    if fsr_value > 3900:
        print("Wow!... You’re strong!”)
    elif fsr_value < 200:
        print("Press the round part to test your strength.")
    else:
        print(Vary how hard you’re pressing to watch the values change.")

    # sleep for 500ms
    time.sleep_ms(500)
```

In-class Exercise 6

- Vary the intensity of the onboard blue LED based on how hard you are pressing on the FSR

- Pressing harder should make the light brighter

GitHub Reminder

All of the code contained in this lecture is available at the MCHE201 Class Repository on GitHub:

`https://github.com/DocVaughan/MCHE201---Intro-to-Eng-Design`