MicroPython Introduction (cont.)

MCHE 201 – Spring 2018

Dr. Joshua Vaughan

Rougeou 225

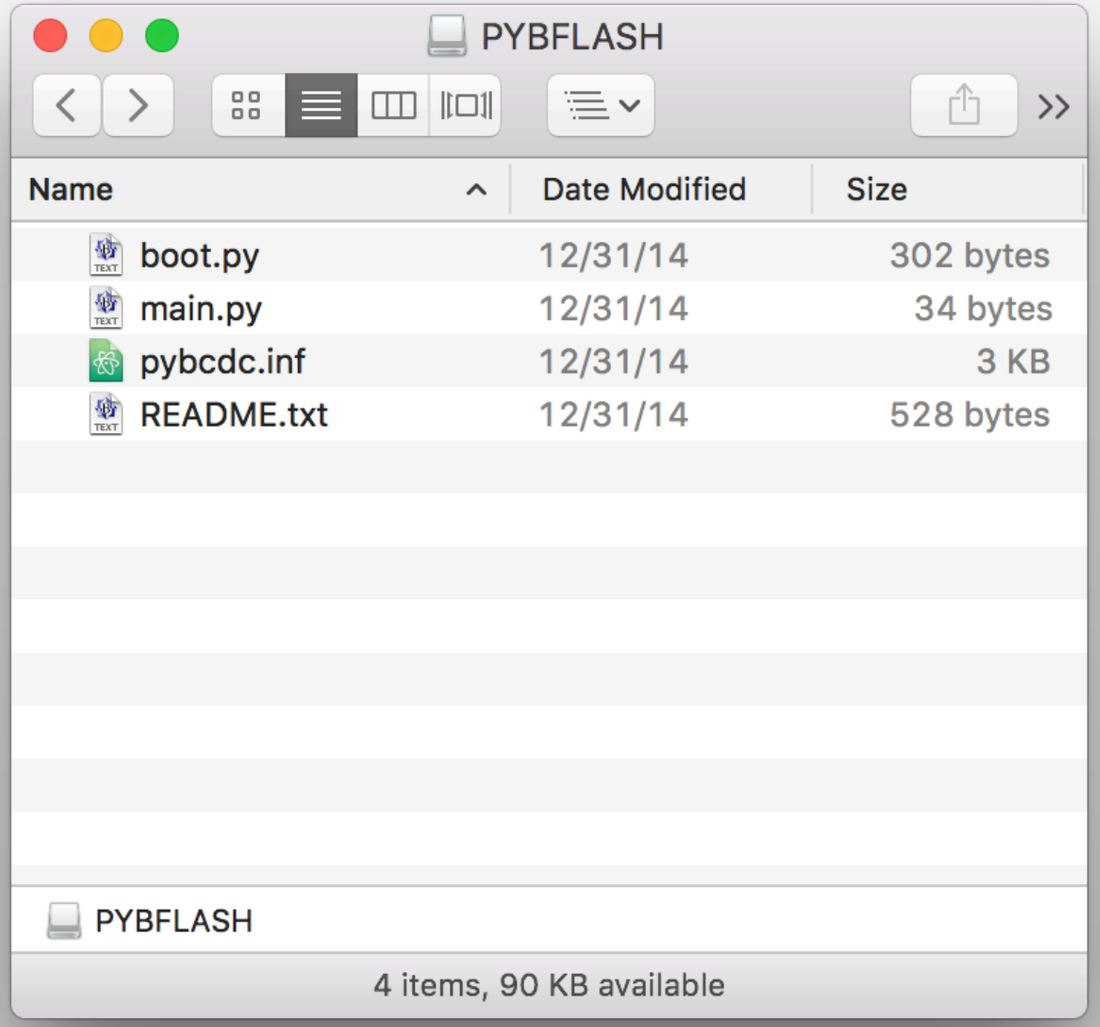
joshua.vaughan@louisiana.edu

@Doc_Vaughan

MicroPython File Review

- `boot.py`
  - Runs every time the pyboard boots
  - Use for setup and configuration

- `main.py`
  - Executed immediately after `boot.py`
  - Use for your “main” code
  - Can reference other files



Review of Using `imports`

Just prepend the variable or function you want to use with the “name” that you imported

```
# Import the pyboard functions
import pyb

# To use a function from pyb, put pyb.
# in front of the function name.
RED_LED = pyb.LED(1)
```

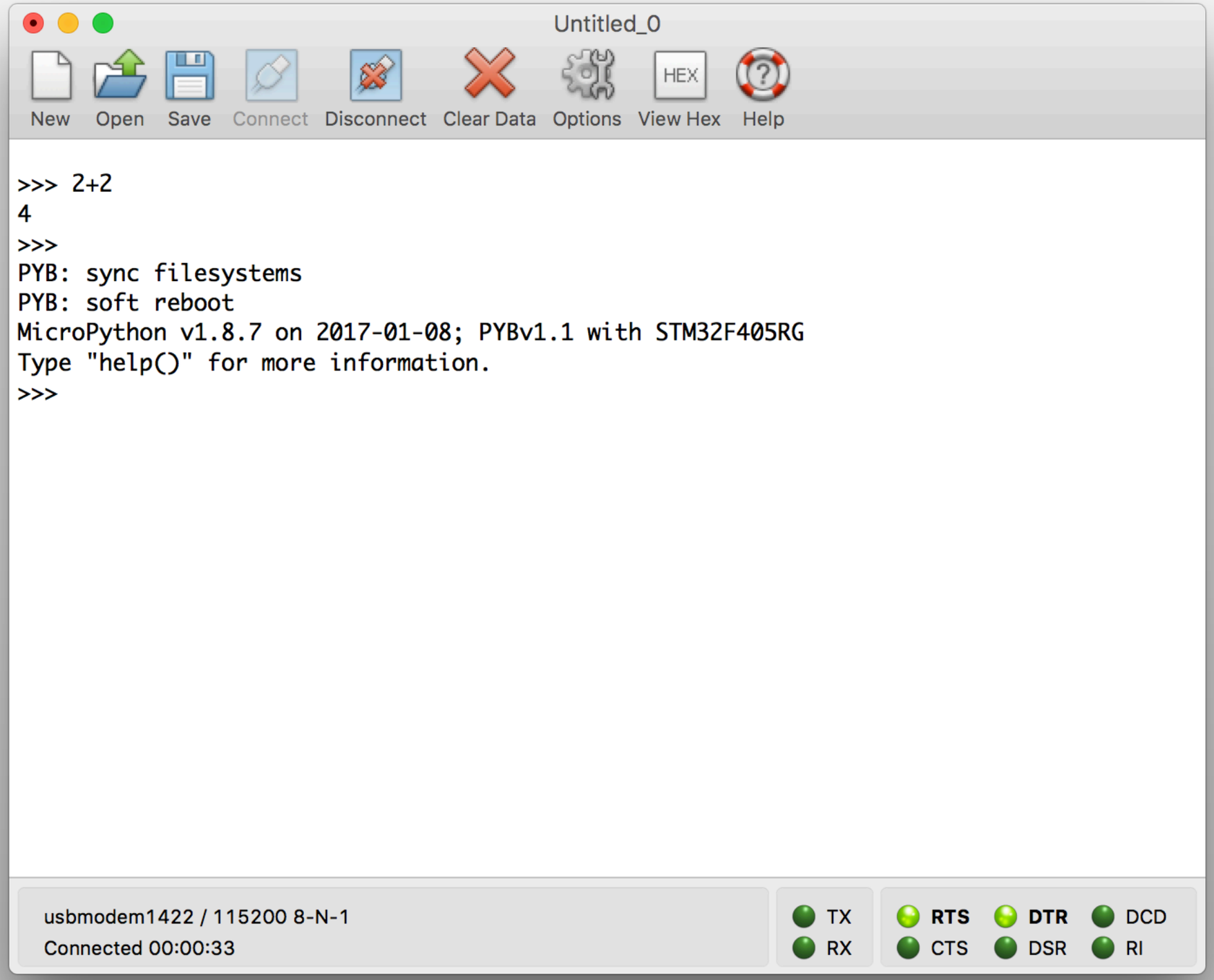
Review of Using `imports`

Just prepend the variable or function you want to use with the “name” that you imported

```
# Import time module
import time

# sleep for 1 second
time.sleep(1)

# sleep for 500 milliseconds
time.sleep_ms(500)

# sleep for 10 microseconds
time.sleep_us(10)
```

REPL Special Command Review

- `Control-d` will perform a soft reboot

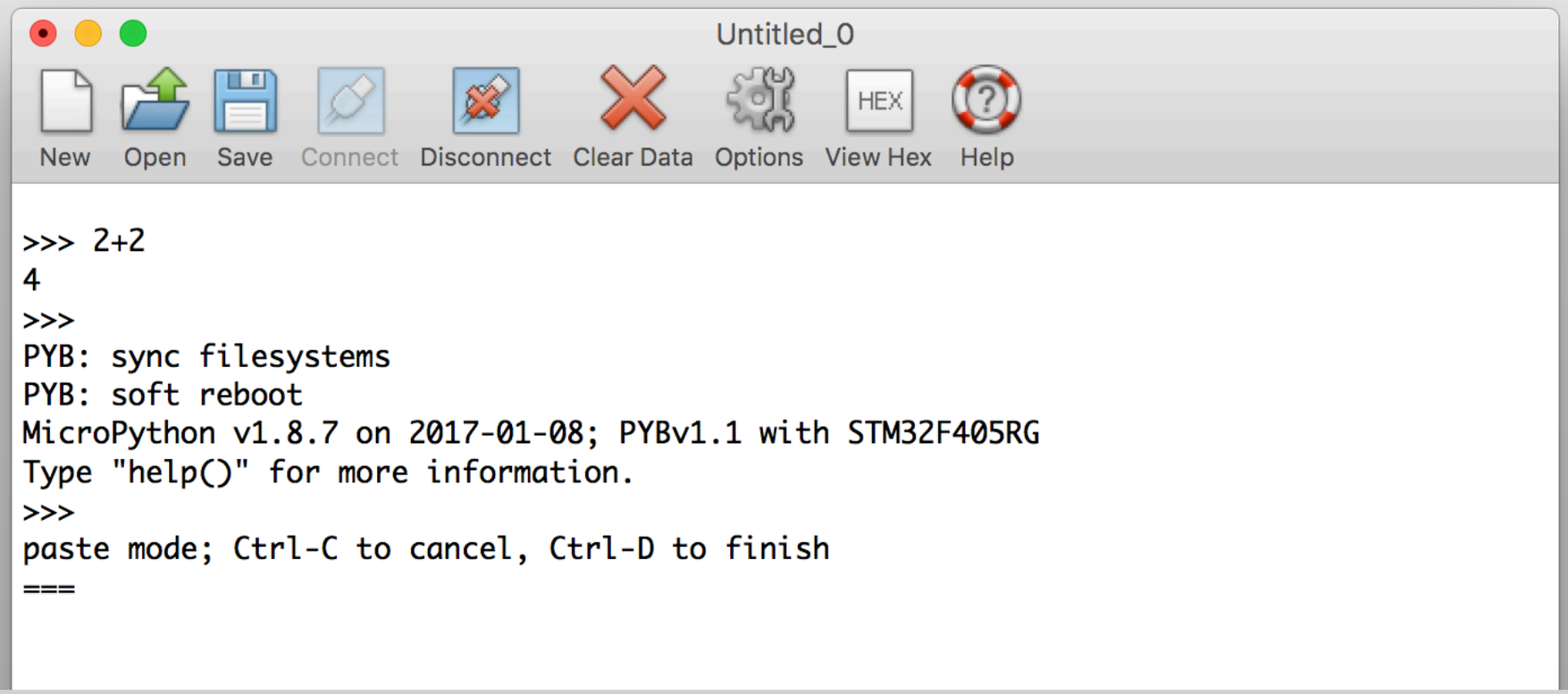


REPL Special Command Review

- `Control-d` will perform a soft reboot
- `Control-c` will kill any running script

REPL Special Command Review

- `Control-d` will perform a soft reboot
- `Control-c` will kill any running script
- `Control-e` will enter paste mode
  - Paste as usual
  - Use `Control-d` to exit paste mode



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`
- More links coming to class webpage
- If you don’t remember the syntax, look it up

Recommended Workflow

- Connect the board to your computer and start the REPL in CoolTerm
- Work on scripts (mostly `main.py` in MCHE201) in a local folder with Atom
- Drag edited versions to `PYFLASH`
- `Control-d` in the REPL to perform a soft reboot and run edited `main.py`

In-Class Exercise 1

- Print the odd numbers between 1 and 27
- *Hint:* A for loop would be a good way to do this.

Exercise 1 – Solution 1

```
# ----- Method 1 -----
# In this first method, we create a range
# of 14 numbers, then simply do the math
# to convert the list to odd numbers

for counter in range(14):
    oddNumber = 2 * counter + 1

    print(oddNumber)
```

Exercise 1 – Solution 2

```
# ----- Method 2 -----
# Here, we'll use a for loop with a properly
# defined range. Here, we use the
# extra terms available in the range function.
# The order is
#    range(start, stop, increment)
# We have to extend the range past 27 because
# the last number listed in not included in the
# range.

for counter in range(1, 29, 2):
    print(counter)
```

Exercise 1 – Solution 3

```
# ----- Method 4 -----
# Here, we'll use a while loop and increment the
# counter ourselves. We'll increment it by 2
# each time to only get the odd numbers. We
# could also increment by 1 and either do math
# on counter to create an odd number, as we did
# in Method 1, or use one an if statement, like
# we did in Method 3

counter = 1

while counter <= 27:
    print(counter)
    counter = counter + 2
```

In-Class Exercise 2

- Print the odd numbers between 1 and 27
- When the number is 13, print “Counter = 13... Bad Luck!!!” and turn on the red LED

Exercise 2 – Solution 1

```
import pyb  # import the pyboard module

# Assign the 1st LED to variable RED_LED
RED_LED = pyb.LED(1)

# ----- Method 1 -----
for counter in range(14):
    # Same math as Exercise 1
    oddNumber = 2 * counter + 1

    if oddNumber == 13:
        print("Counter = 13... Bad Luck!!!")
        RED_LED.on()      # Turn the RED_LED on
    else:
        print(oddNumber)
        RED_LED.off()     # Turn the RED_LED off
```

Exercise 2 – Solution 2

```
import pyb  # import the pyboard module

# Assign the 1st LED to variable RED_LED
RED_LED = pyb.LED(1)

# ----- Method 2 -----
for counter in range(1, 29, 2):

    if counter == 13:
        print("Counter = 13... Bad Luck!!!")
        RED_LED.on()      # Turn the RED_LED on
    else:
        print(counter)
        RED_LED.off()     # Turn the RED_LED off
```

The pyboard

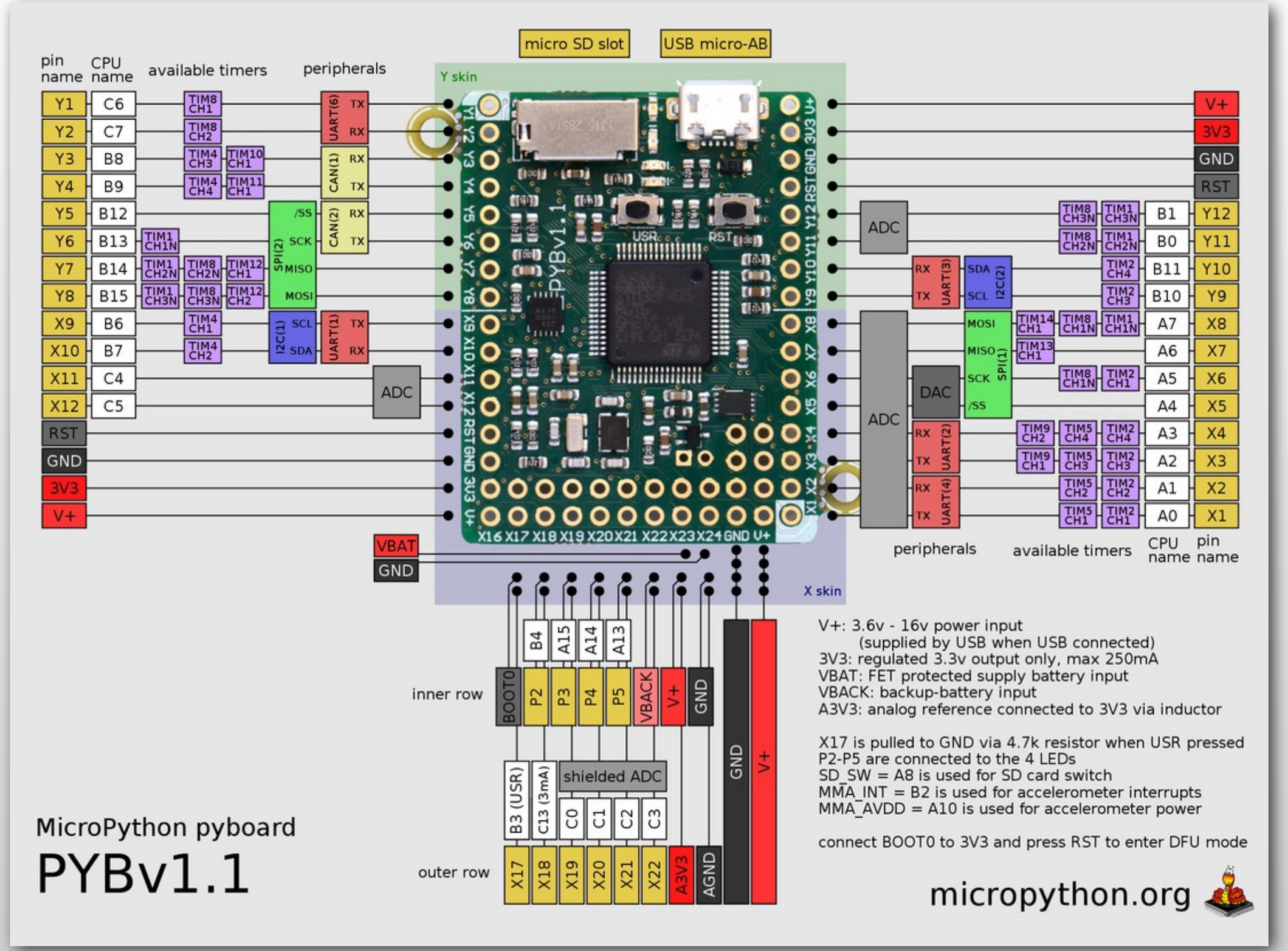
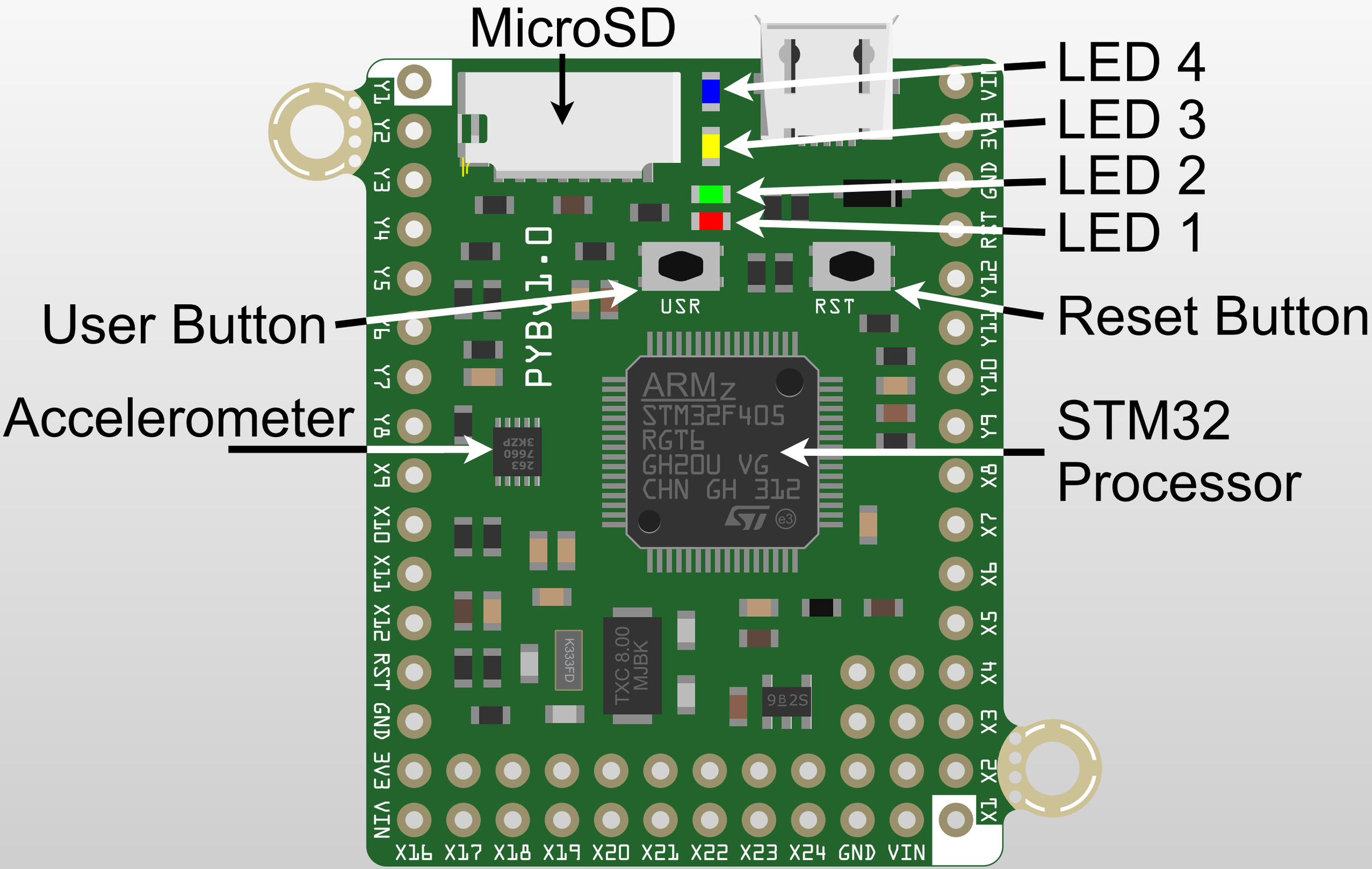


Image via `micropython.org`

The Onboard Hardware



Onboard LEDs Review

- Numbered 1-4
- Follow same pattern as earlier RED_LED example

```
import pyb # import the pyboard module

# Assign the names to the onboard LEDs
RED_LED = pyb.LED(1)
GREEN_LED = pyb.LED(2)
YELLOW_LED = pyb.LED(3)
BLUE_LED = pyb.LED(4)
```

Onboard LED methods

- For all 4 onboard LEDs
  - `on()` – turn the LED on
  - `off()` – turn the LED off
  - `toggle()` – toggle the state of the LED

- For the third (yellow) and fourth (blue) LEDs
  - `intensity()` – set or get the brightness of the LED
    - ✦ If a number is inside, set to that value (between 0-255)
    - ✦ If no argument, get the current intensity

Reading the Onboard Button

- It's a "switch" in MicroPython

- We can:
  - Get its current state manually and/or
  - Set up code to run automatically any time it's pressed

- For both, we need to set up a "switch" object

```
import pyb # import the pyboard module

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

Manually Reading the Button

```
import pyb # import the pyboard module

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

# call the variable assigned like it’s a
# function. True, if pressed. It will
return False.
button()
```

Reading the Button Indefinitely

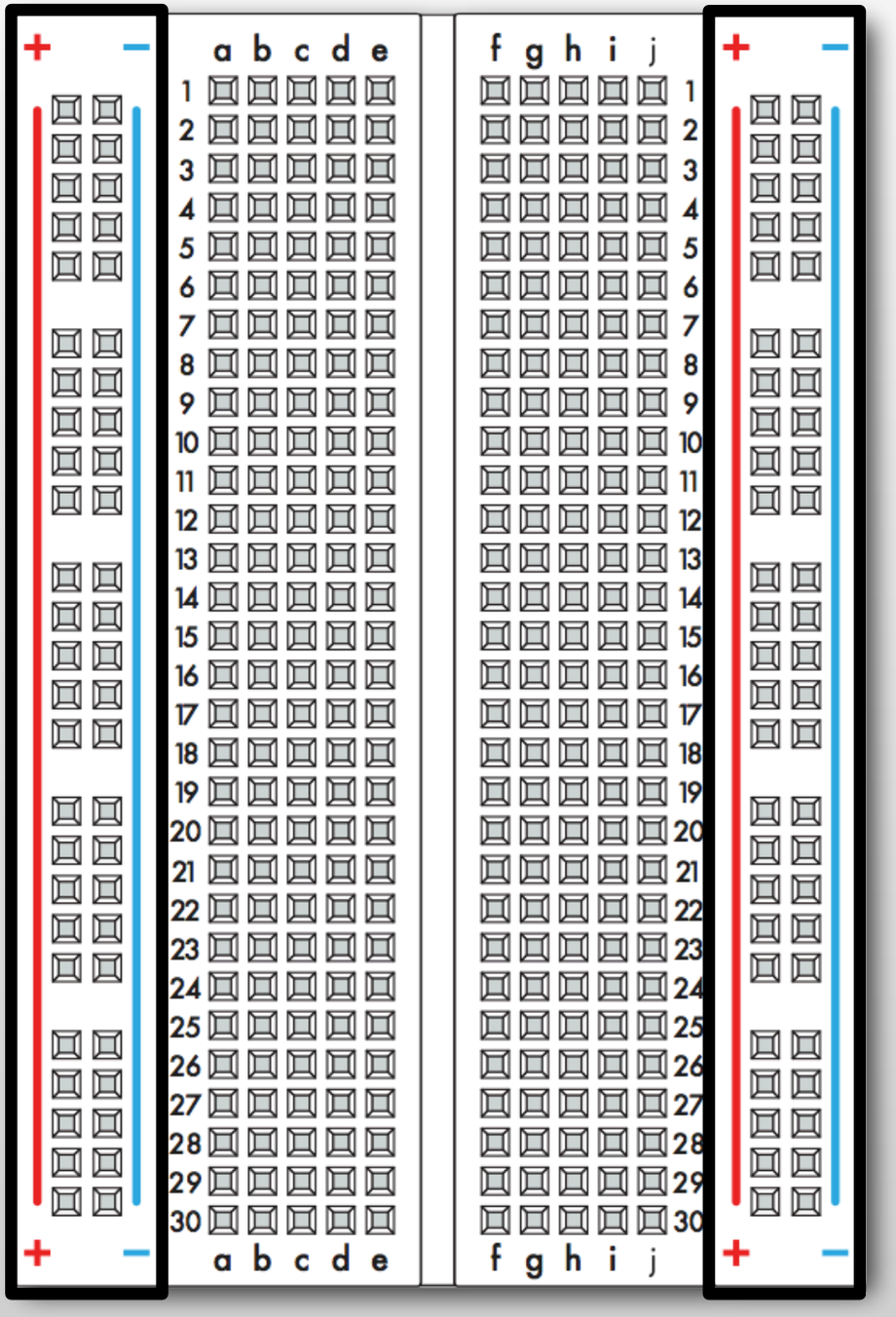
```
import pyb  # import the pyboard module
import time # import the time module

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

# The condition for this while is always true, so
# it runs forever
while (True):
  # button() is True if the button is pressed
  if (button()):
    print("Button Pressed!")

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

In-Class Exercise 3

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

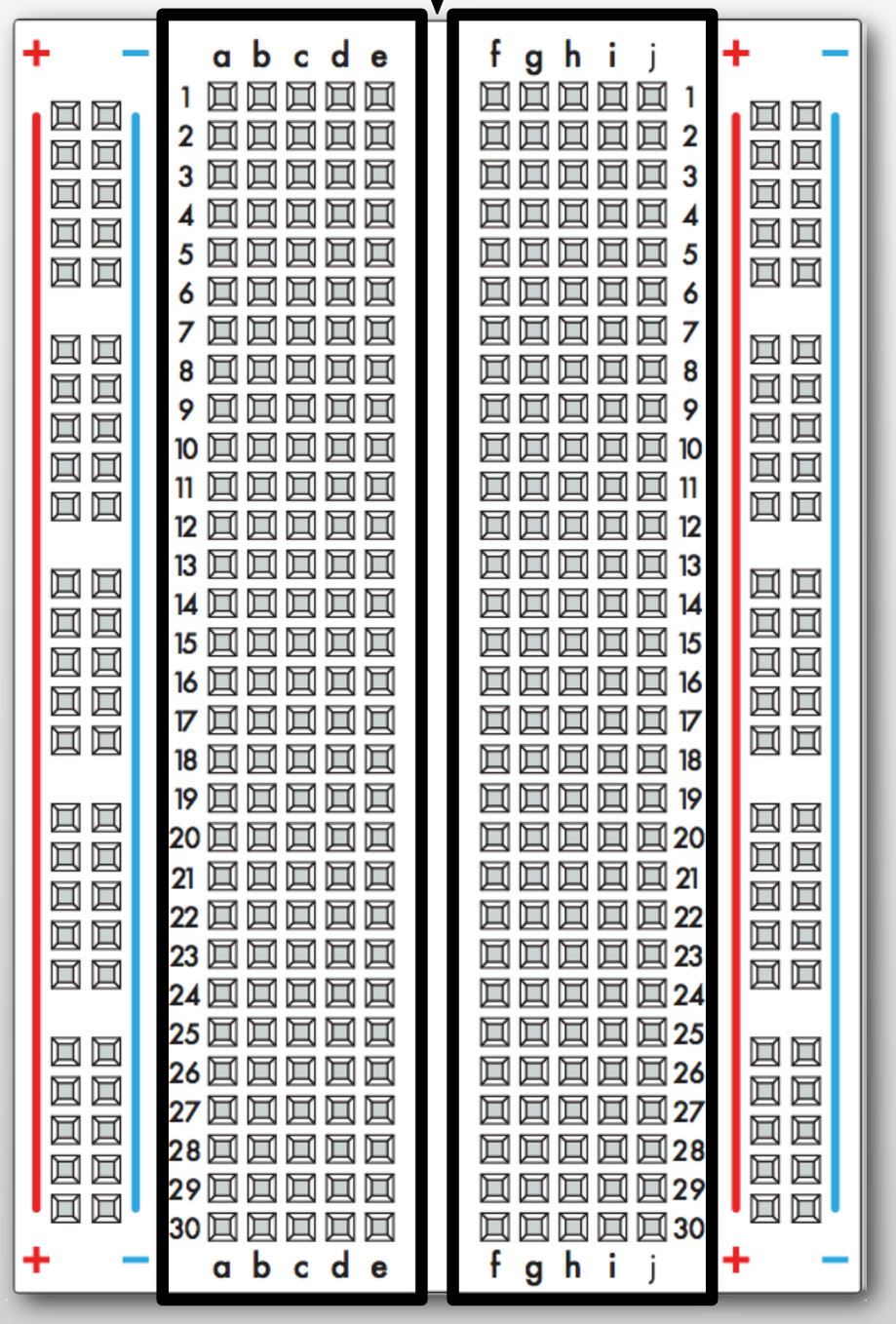
Basic Breadboard Setup

Connected by column



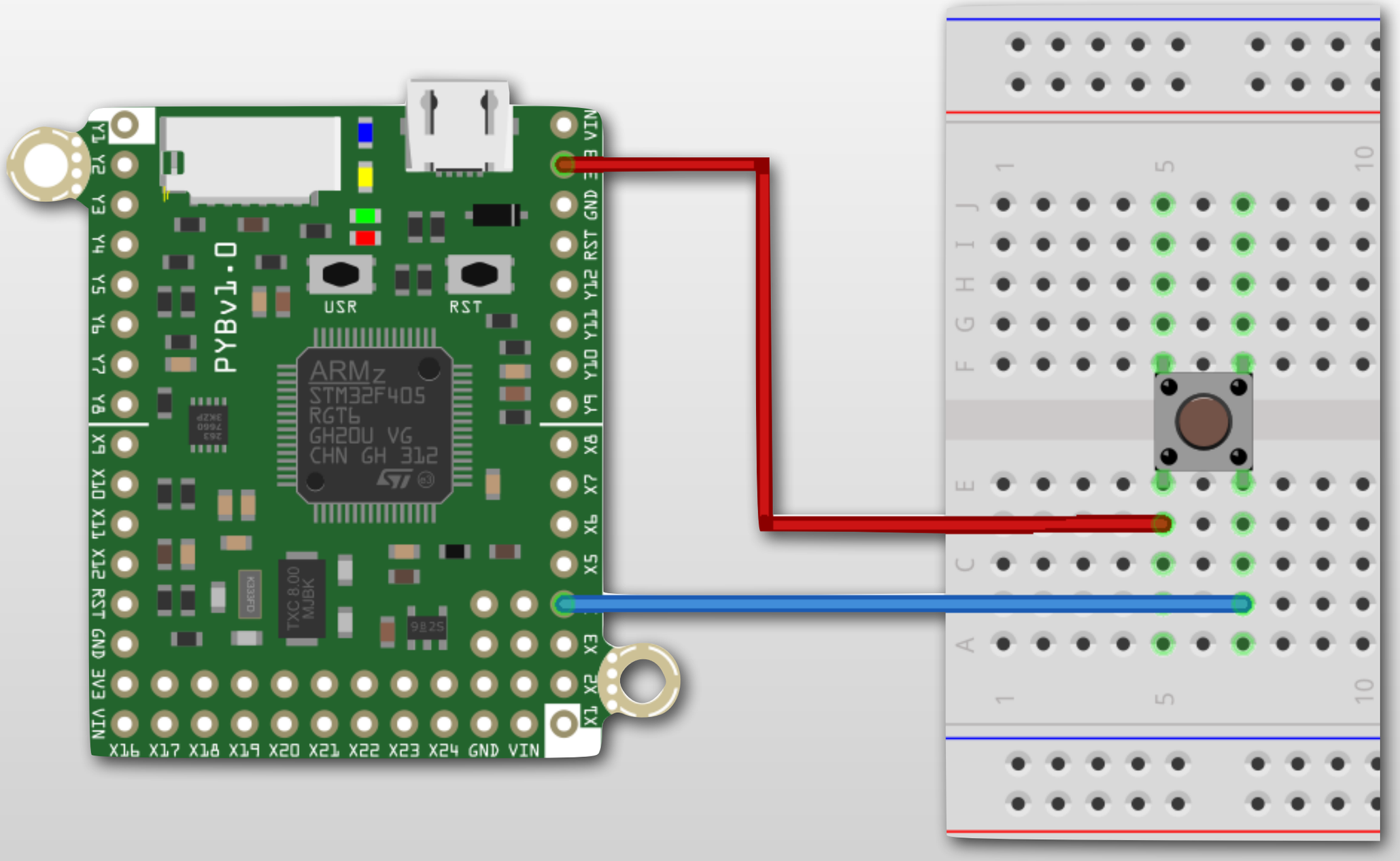
Basic Breadboard Setup

Break in connection



Connected by row

Pushbutton Hardware Setup



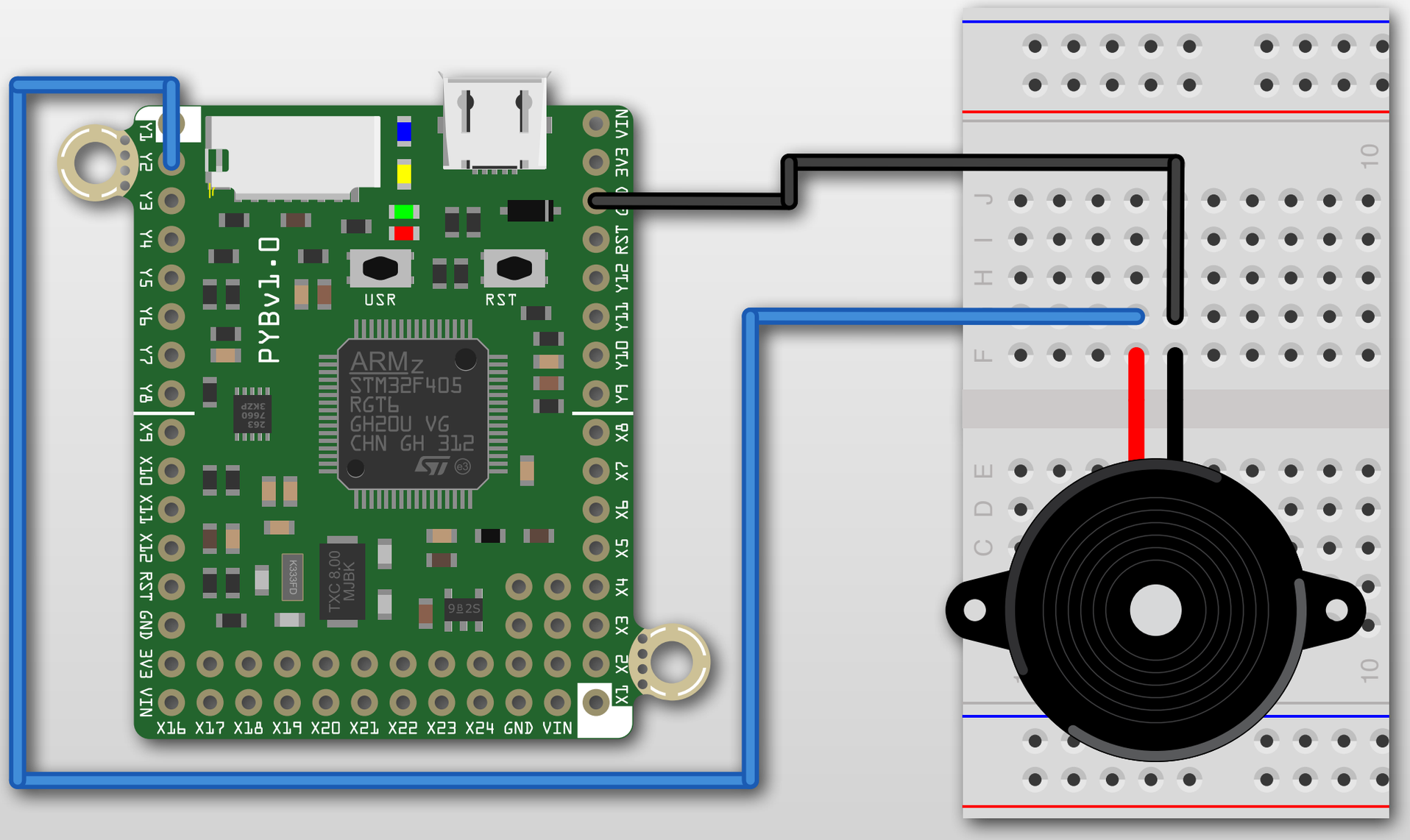
MicroPython Code for Pushbutton

```
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("X4", 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

Buzzer Hardware Setup



Buzzer MicroPython Code

https://github.com/DocVaughan/MCHE201---Intro-to-Eng-Design/blob/Spring-2018/MicroPython/pyboard%20buzzer/main.py