**Lesson\_X\_Notes\_and\_Journal**

**2024.04.XX**

* Look at Code
* Look at Circuit
* Draw Circuit
* Build Circuit
* Start Lesson

**Code**

* Glance at Code

import machine

import time

# Define GPIO pins for the 4 LEDs

led\_pins = [

machine.Pin(0, machine.Pin.OUT),

machine.Pin(1, machine.Pin.OUT),

machine.Pin(2, machine.Pin.OUT),

machine.Pin(3, machine.Pin.OUT)

]

button\_pin = machine.Pin(8, machine.Pin.IN, machine.Pin.PULL\_DOWN)

# Initialize LED states for 4 LEDs

led\_states = [0] \* len(led\_pins)

# Function to update LED states

def update\_led\_states():

for i, led in enumerate(led\_pins):

if led\_states[i] == 1:

led.on()

else:

led.off()

# Function to create a padded binary string

def padded\_binary\_string(value, length):

binary\_str = bin(value).replace("0b", "")

while len(binary\_str) < length:

binary\_str = '0' + binary\_str

return binary\_str

# Function to increment binary counter

def increment\_counter():

# Convert LED states to integer

binary\_value = int(''.join(map(str, led\_states)), 2)

binary\_value += 1

# If value is 16 (10000 in binary), reset to 0

if binary\_value >= 16:

binary\_value = 0

binary\_string = padded\_binary\_string(binary\_value, len(led\_pins))

# Update LED states

for i, bit in enumerate(binary\_string):

led\_states[i] = int(bit)

# Main loop

while True:

if button\_pin.value() == 1:

increment\_counter()

update\_led\_states()

time.sleep(0.2) # Button debounce delay

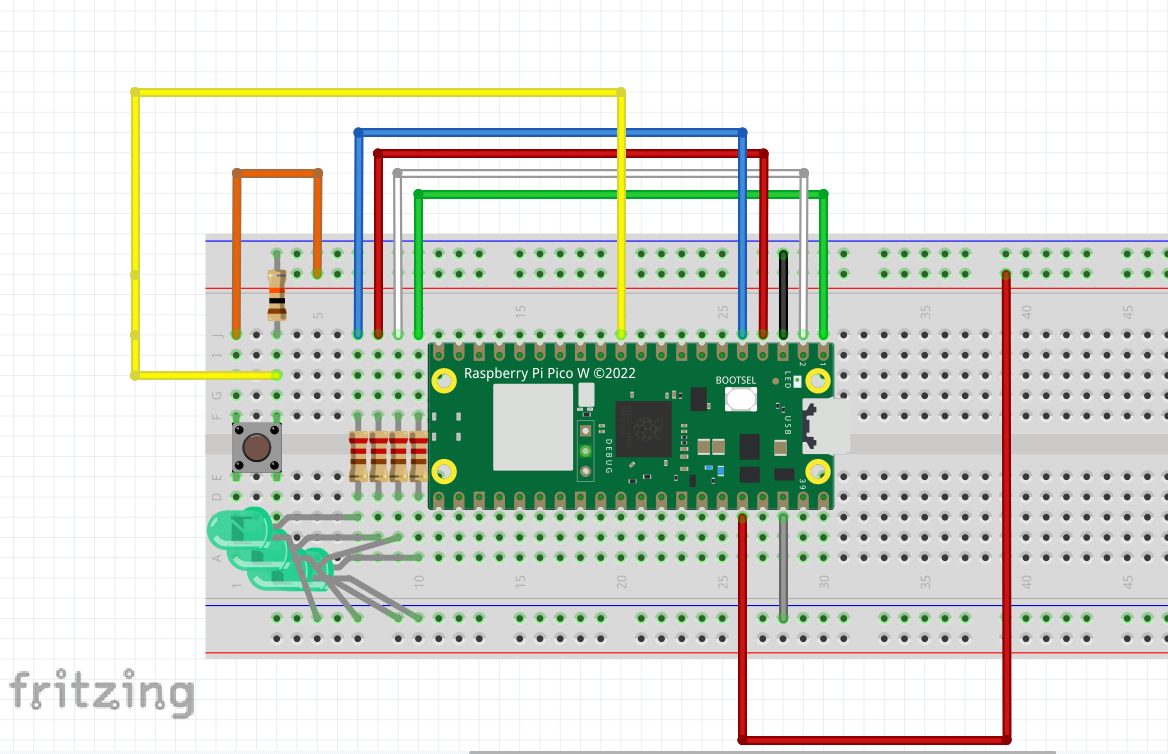
**Circuit**

* Theirs

A circuit board with wires and wires

Description automatically generated

* Mine



A diagram of a circuit

Description automatically generated

* As built