**2024\_05\_03\_Troubleshooting**

Plan:

Use Parts of Binary Counter Circuit and Code that works to try to get this working.

**Lesson\_4\_Troubleshooting\_Notes\_and\_Journal**

**2024.05.03**

* Look at Code
* Look at Circuit
* Use Lesson 6 Circuit
* Get Blue wire LED working
* Pare away unused LEDS, wires and resisters
* Re-Draw Circuit
* Test

**Code**

* First Revision
* # Define GPIO pins for the LED and button – 5/3/2024 Based on Lesson 4 Circuit
* led\_pin = machine.Pin(3, machine.Pin.OUT) # WAS 14
* button\_pin = machine.Pin(8, machine.Pin.IN, machine.Pin.PULL\_DOWN) # WAS 15
* Second Revision

A white background with black text

Description automatically generated

* Original Lesson 4 Glance at Code
* import machine
* import time
* # Define GPIO pins for the LED and button
* led\_pin = machine.Pin(14, machine.Pin.OUT)
* button\_pin = machine.Pin(15, machine.Pin.IN, machine.Pin.PULL\_DOWN)
* # Function to measure reaction time
* def reaction\_time\_tester():
* time.sleep(5)
* led\_pin.on() # Turn on the LED
* start\_time = time.ticks\_ms() # Get start time in milliseconds
* # Wait for button press
* while True:
* if button\_pin.value() == 1:
* end\_time = time.ticks\_ms() # Get end time in milliseconds
* reaction\_time = end\_time - start\_time # Calculate reaction time
* print('Reaction Time:', reaction\_time, 'ms')
* break
* led\_pin.off() # Turn off the LED
* # Call the function to start the reaction time tester
* reaction\_time\_tester()

* Lesson 6 REFERENCE 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

**Blue Wire is Fifth Pin: 5 | GP3, etc.**

**Yellow Wire is Eleventh Pin: 11 | GP8, etc.**

A circuit board with wires and a green circuit board

Description automatically generated

A computer chip with many colored labels

Description automatically generated with medium confidence