# ECE 303 Lab Technical Memos

Nicholas Sica

October 2, 2020

# Contents

# 1 Lab 1

## 1.1 Discussion

The point of this lab was to introduce us to the Arduino toolkit and allow anyone who is new to the platform time to adjust and get familiar with the tools presented to them. The lab was straightforward, connect an LED and resistor to a pin on the arduino and writing simplistic code to change the pulse width modulation values. The output is as expected, with the intensity of the light changing based off the number put into the serial monitor, zero being off and 255 being full brightness.

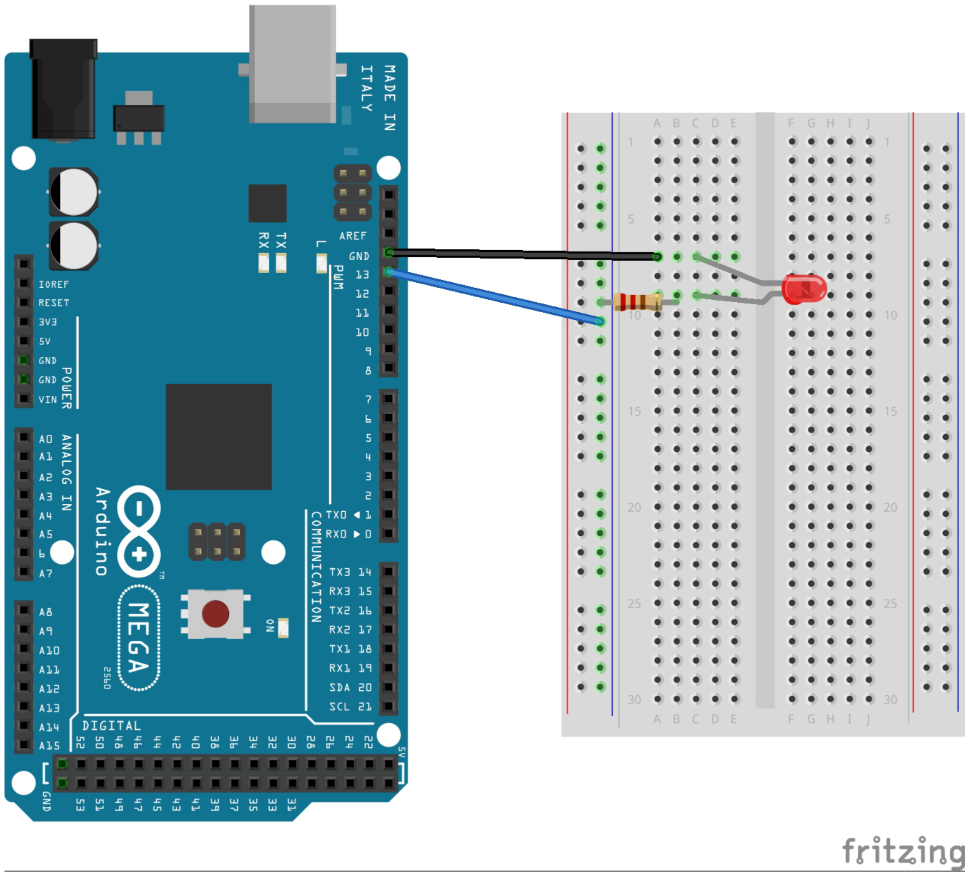

Figure 1: Basic LED Circuit Setup

## 2 Lab 2

### 2.1 Discussion

This lab was used to get us acquainted with timers and learn not only how to set up the correct bit values, but also how to use them efficiently. A lot of the time was spent debugging the bit values that the different masks are initialized with as well as learning how to turn the interrupts off efficiently. When run, every LED blinks at a slow pace and each guess for the code either causes it to blink faster, if wrong, or turn off, if correct. When the user is out of attempts, the remaining LEDs stay permanently on.

$$OCR3A = \frac{16 \times 10^6}{p \times f} - 1 \tag{1}$$

The value of the register was found using equation 1, where p is the prescalar, in this case 1024, and f is the target frequency, in this case around 0.5 hertz or a 2 second period. Every subsequent wrong guess, it was divided by two to make it go faster.

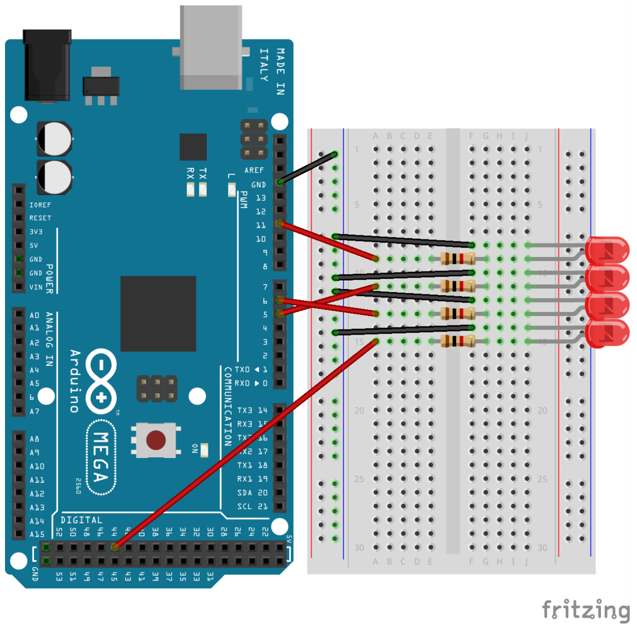

Figure 2: Codebreaker Circuit Setup

## 2.2 Hardware

Figure 2 shows the setup of the circuit. The setup of the hardware was very similar to the previous lab, but this time there are four LEDs and care was taken to connect to them to the correct pin corresponding to the timer we want to use for it.

# 3 Lab 3

## 3.1 Discussion

This lab was used to get us to get more comfortable with timers in a different way as well as introduce us to a photocell. This time around not a lot of time was spent getting the correct timer values, we just had to make sure that they represented the correct duty cycle. The results for this lab were a bit worse due to lights coming from my computer. I covered it with a box, but there was bound to be some leakage. It would have been better if I could isolate my circuit in a container and have my photocell the perfect distance from the LED. After copying all the data to excel the voltage divider equation shown in Equation 2 was used to get the resistances of the photocell and led. $V_{DD}$ is the source voltage, in this case 5V, $R_{gnd}$ is the resistor leading to ground, 10k in the photocell circuit and 1k in the led circuit, $V_{out}$ is the voltage read from pin A0 for the led and pin A1 for the photocell.

$$R = \frac{V_{DD} \times R_{gnd}}{V_{out}} - R_{gnd} \tag{2}$$

The voltage of the LED and photocell were found by using Kirchoff's Volt-

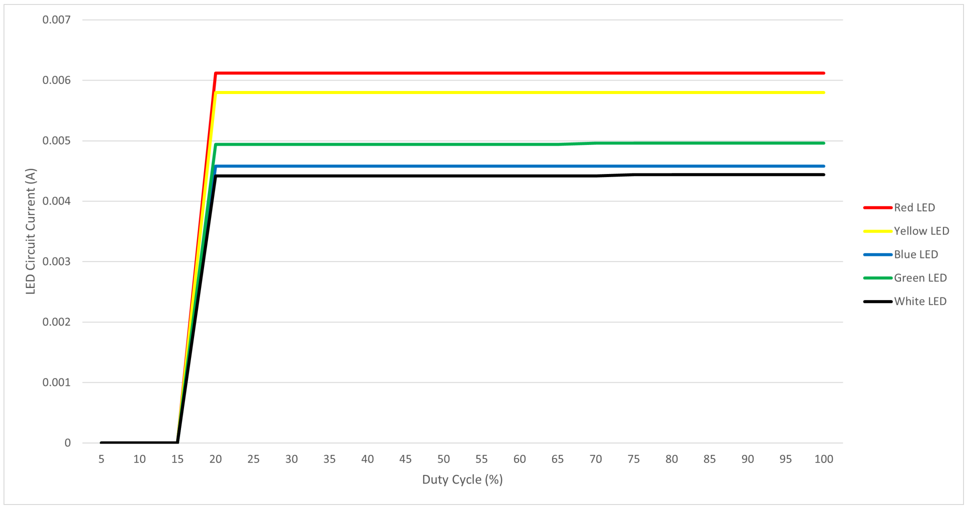

Figure 3: Duty Cycle Versus LED Circuit Current

age law and subtracting the voltage across the respective resistor from 5V.

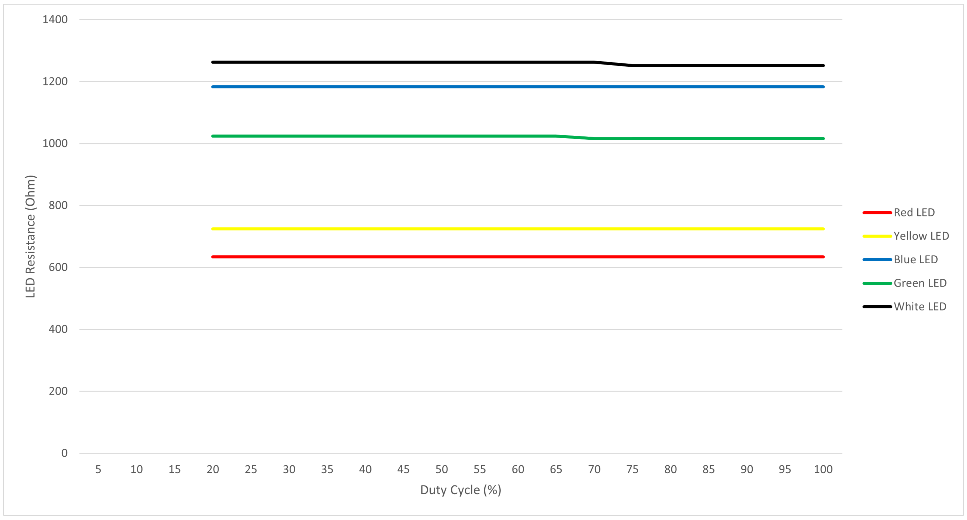

Figure 4: Duty Cycle Versus LED Resistance

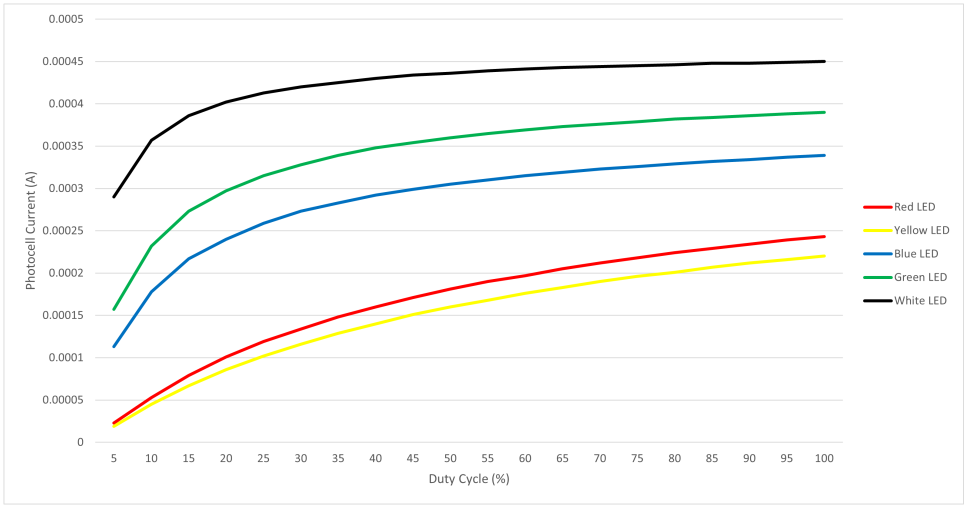

Figure 5: Duty Cycle Versus Photocell Current

Lastly, the currents were found using the previous values found for voltage and resistance and applying Ohm's Law. An error in the values read from pin A0 was introduced due to the fact that we were reading it from a PWM source. Figure 3 shows the duty cycle versus LED circuit current which is what you'd expect, a constant value after it gets up and running. Figure 4

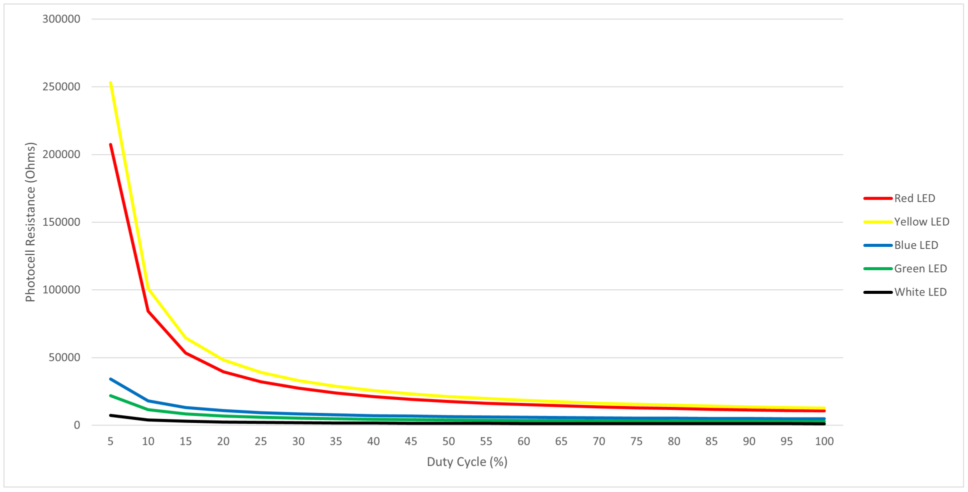

Figure 6: Duty Cycle Versus Photocell Resistance

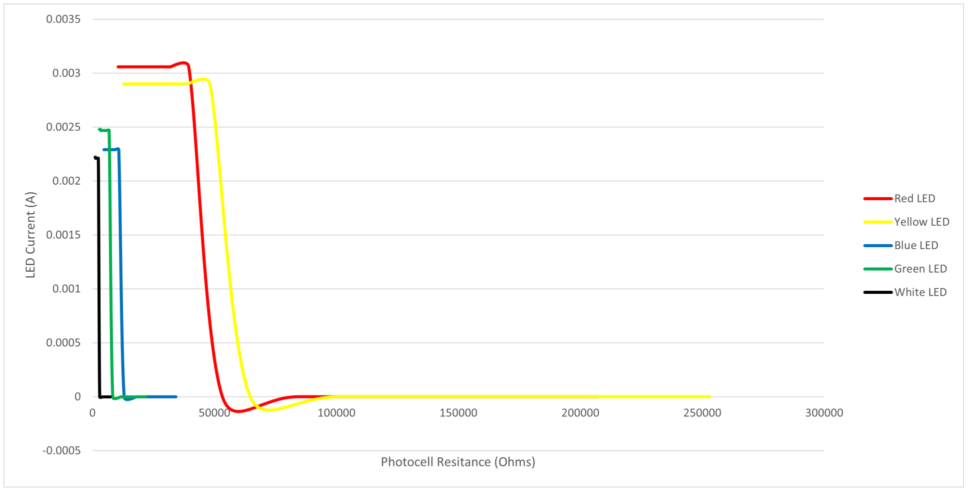

Figure 7: Photocell Resistance Versus LED Current

shows the duty cycle versus the LED resistance which is a constant value due to the voltage applied not changing. Figure 5 shows duty cycle versus photocell current which is increasing due to the resistance in the photocell decreasing as the LED strength increases. Figure 6 shows duty cycle versus photocell resistance which is decreasing due to the LED growing in strength. Finally, Figure 7 which shows photocell resistance versus LED current which is the most interesting because it is fairly constant until it suddenly dips.

## 3.2 Hardware

Figure 8 shows the setup of the circuit. The setup of the hardware was very simple, but could've been improved with a housing that blocks out light. You could also get different results depending on the distance the photocell was from the LED, which could lead to skewed results when swapping the LED out for another color.

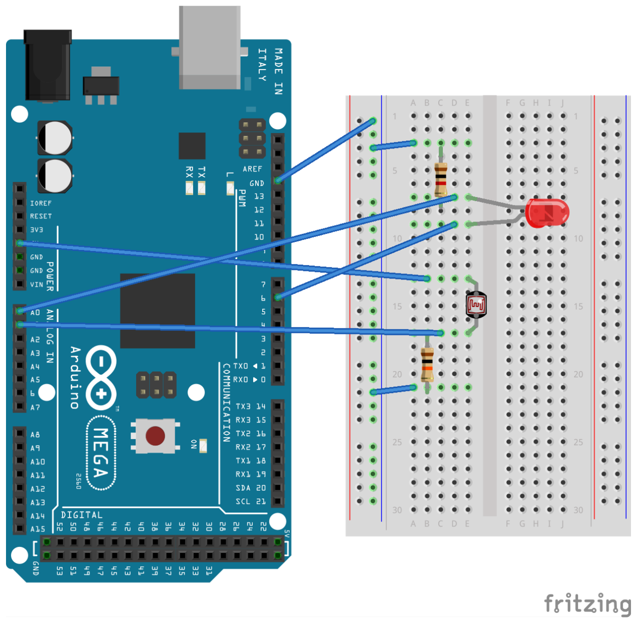

Figure 8: Photocell and LED Circuit Setup

# 4 Lab 4

## 4.1 Discussion

This lab was used to introduce us to using Python or MATLAB in sync with the arduino and teach us how to connect the two and gather data automatically. This time around not a lot of time was spent getting the correct timer values, we just had to make sure that they represented the correct duty cycle. The results for this lab, like the last one, were a bit worse due to lights coming from my computer. I covered it with a box, but there was bound to be some leakage. It would have been better if I could isolate my circuit in a container and have my photocell the perfect distance from the LED. The same basic idea was used as the last lab except this time there was serial communication through Python and all the graphs were made using matplotlib. The values of the voltage, current, and

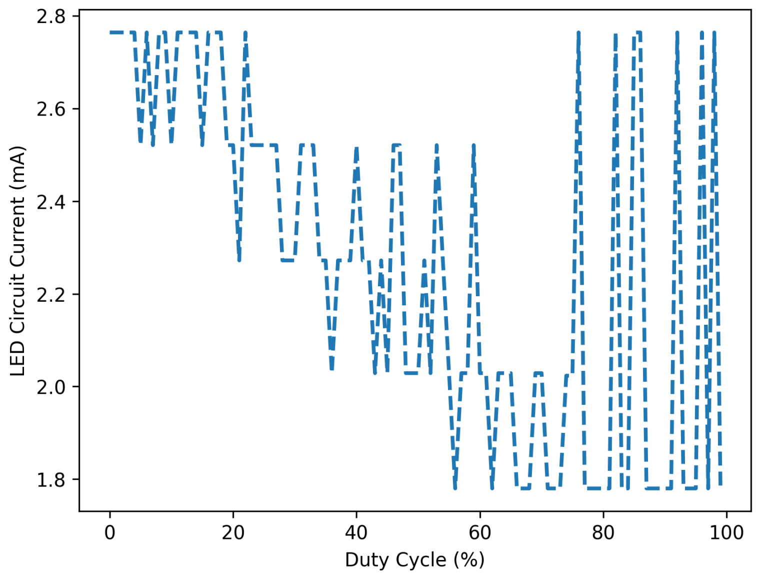

Figure 9: White Duty Cycle Versus LED Circuit Current

resistance were found using the same method as the last lab. An error in the values read from pin A0 was introduced due to the fact that we were reading it from a PWM source. Figure 9 shows the duty cycle versus LED circuit current. You are seeing a finer grain look than last time and even get to see

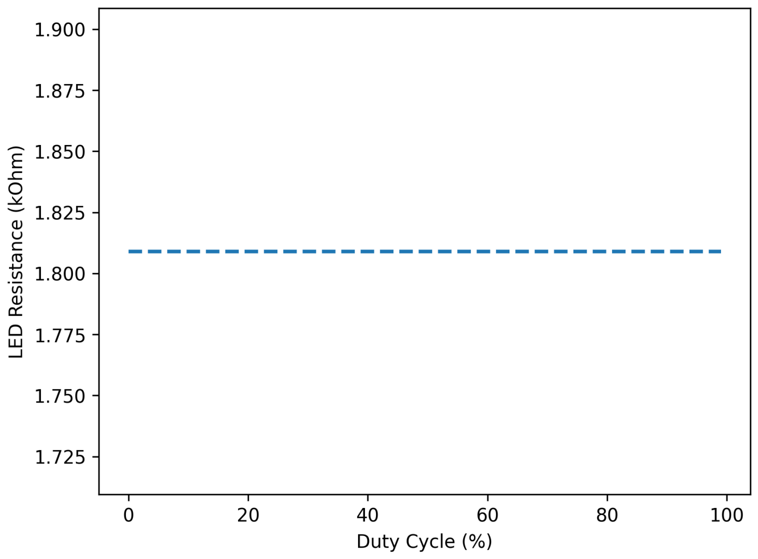

Figure 10: White Duty Cycle Versus LED Resistance

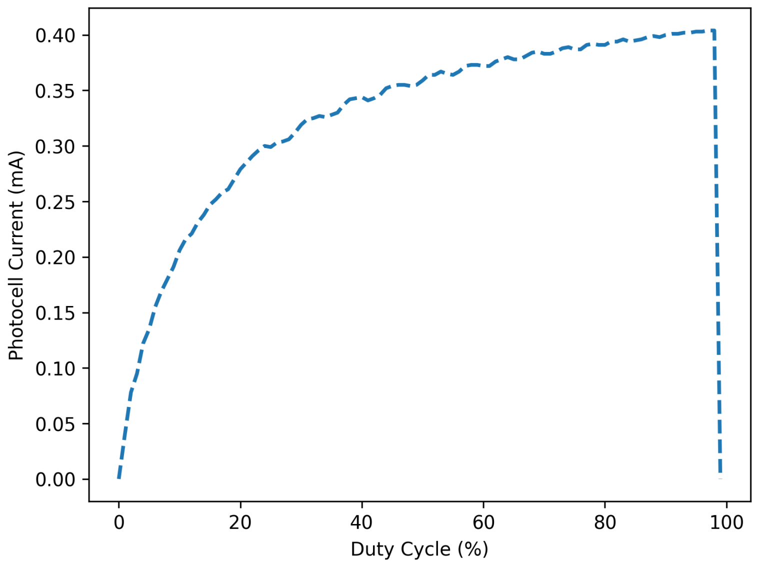

Figure 11: White Duty Cycle Versus Photocell Current

a bit of the PWM in action. Figure 10 shows the duty cycle versus the LED resistance which is a constant value due to the voltage applied not changing.

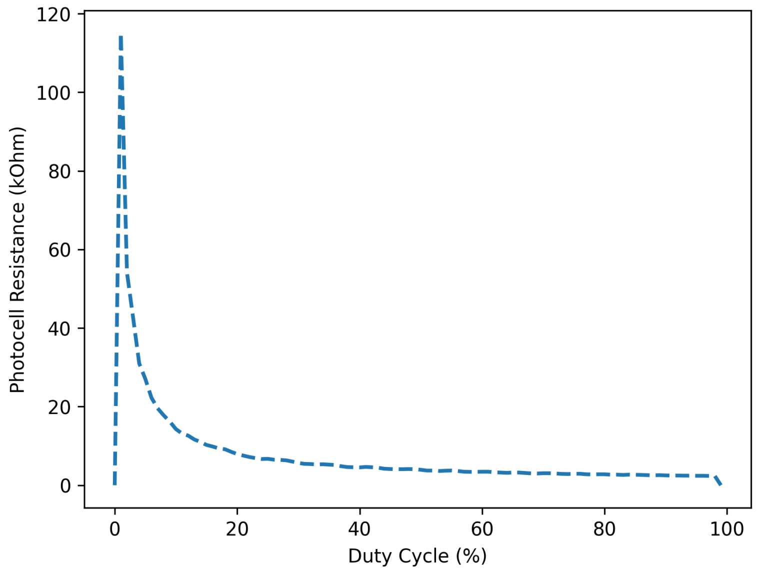

Figure 12: White Duty Cycle Versus Photocell Resistance

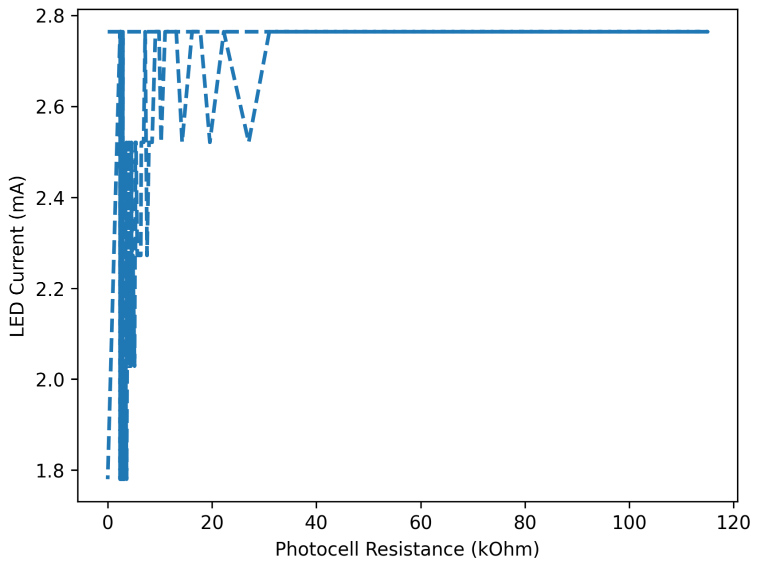

Figure 13: White Photocell Resistance Versus LED Current

Figure 11 shows duty cycle versus photocell current which is increasing due

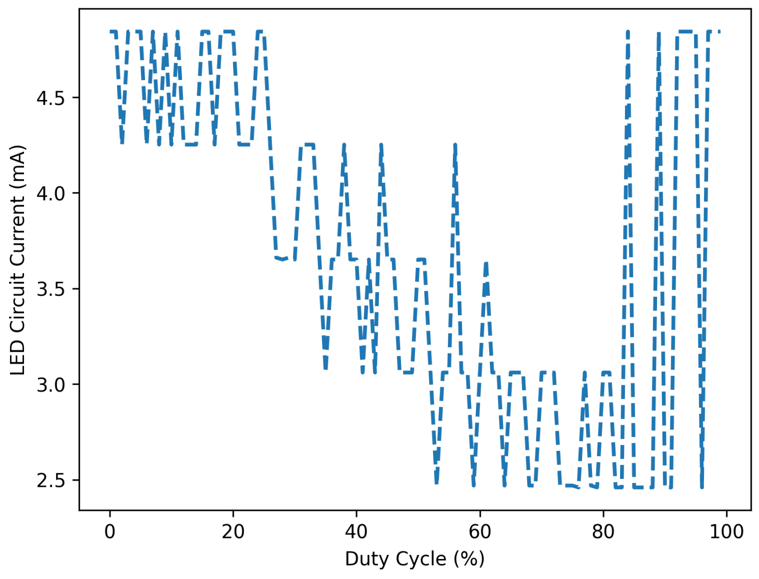

Figure 14: Red Duty Cycle Versus LED Circuit Current

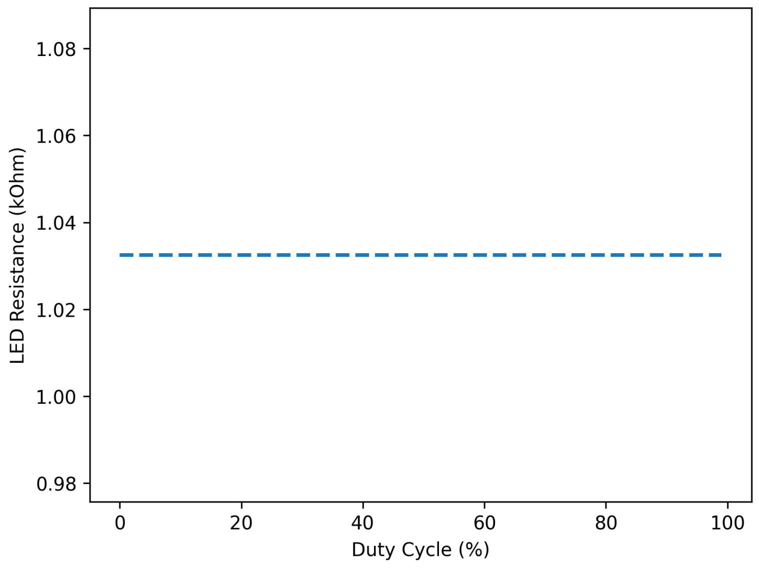

Figure 15: Red Duty Cycle Versus LED Resistance

to the resistance in the photocell decreasing as the LED strength increases.

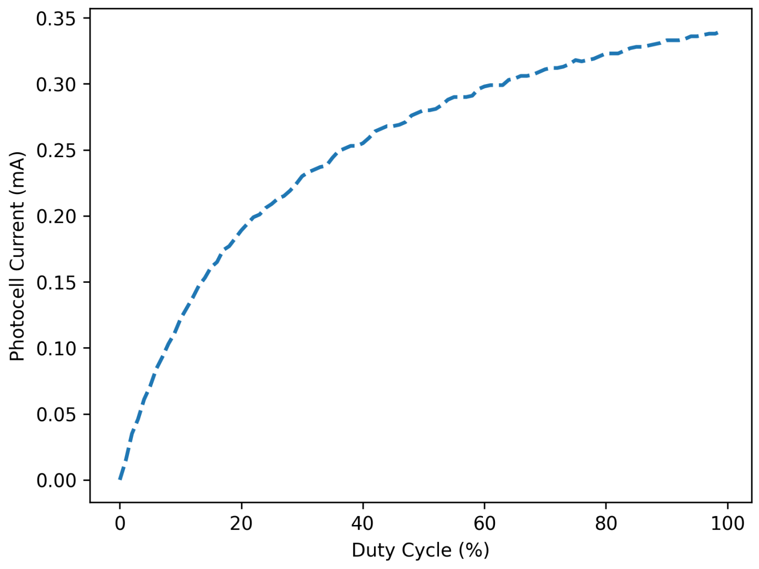

Figure 16: Red Duty Cycle Versus Photocell Current

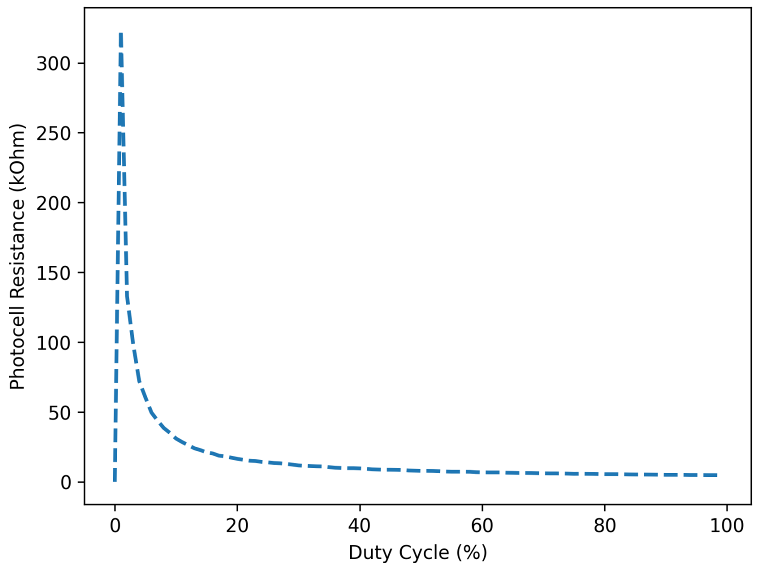

Figure 17: Red Duty Cycle Versus Photocell Resistance

The part at the end where it dips is probably due to the imperfect setup we have and finer granularity than the excel version. Figure 12 shows duty cycle

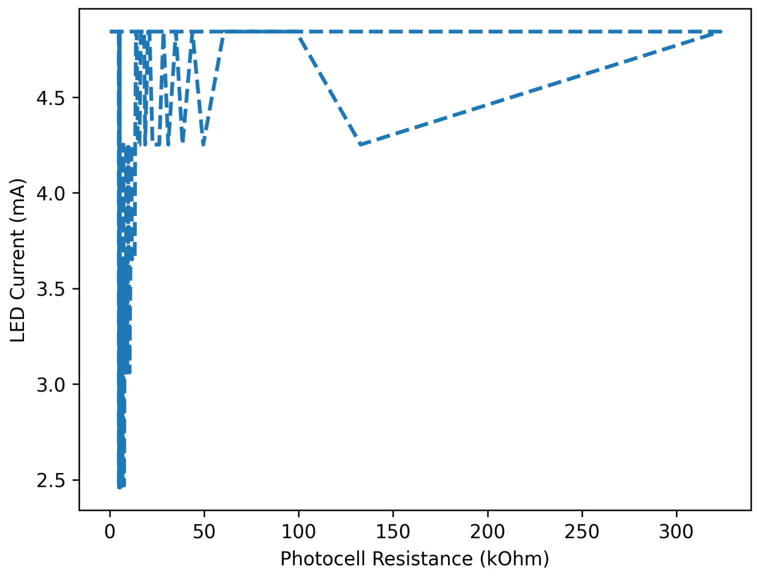

Figure 18: Red Photocell Resistance Versus LED Current

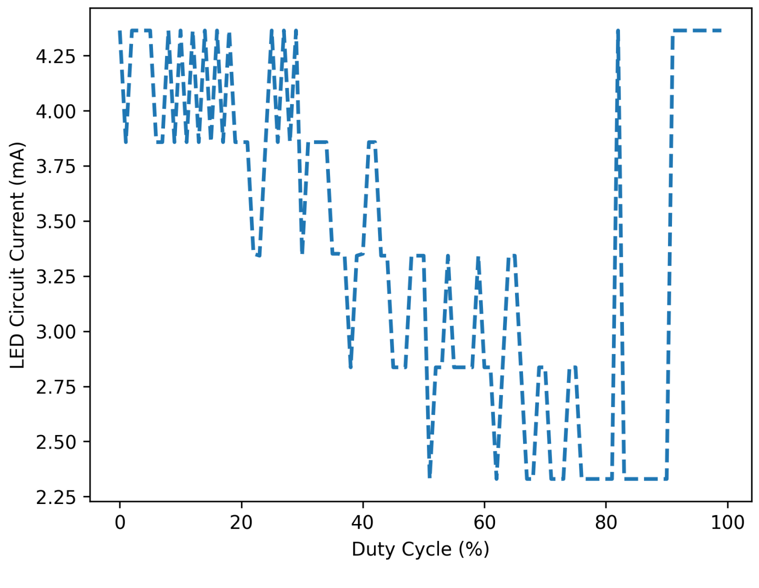

Figure 19: Yellow Duty Cycle Versus LED Circuit Current

versus photocell resistance which is decreasing due to the LED growing in

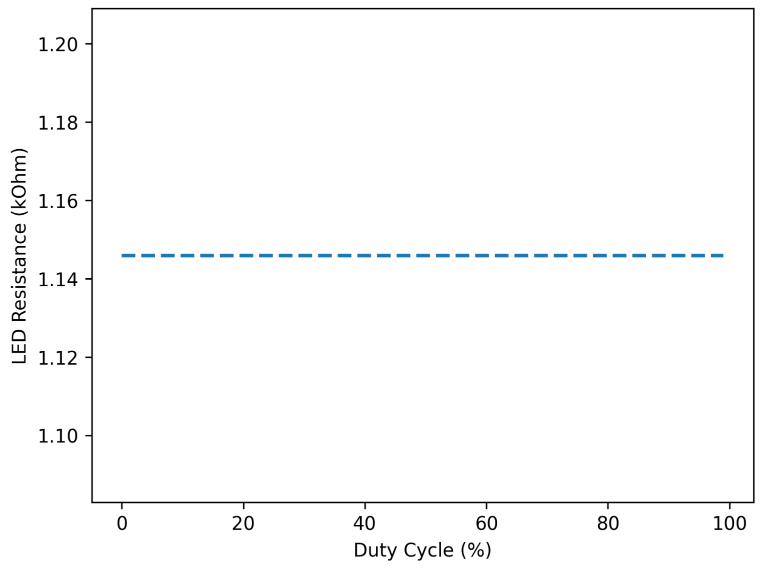

Figure 20: Yellow Duty Cycle Versus LED Resistance

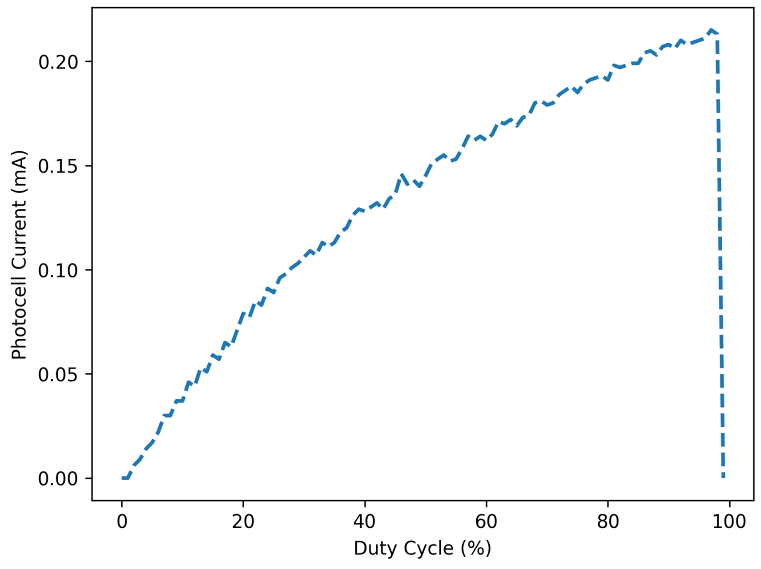

Figure 21: Yellow Duty Cycle Versus Photocell Current

strength. It starting at zero could also be due to the imperfect setup or the photoreistor not picking up the light at a low strength. Finally, Figure 13

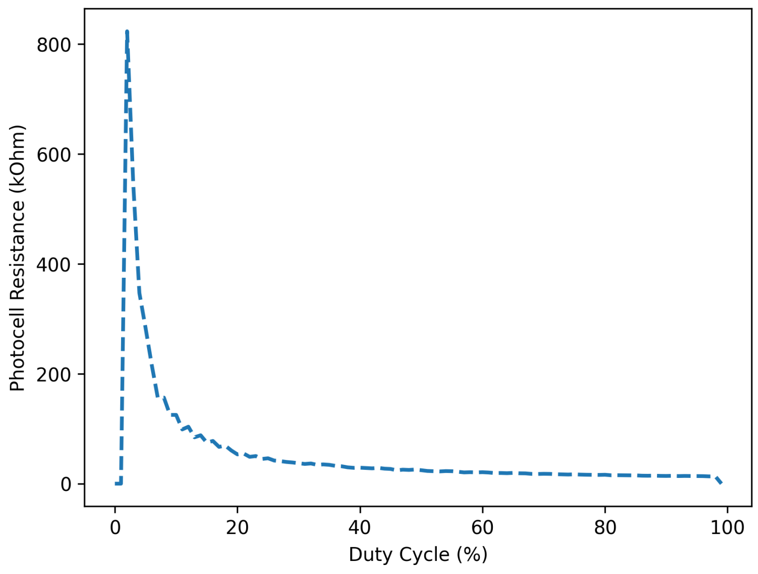

Figure 22: Yellow Duty Cycle Versus Photocell Resistance

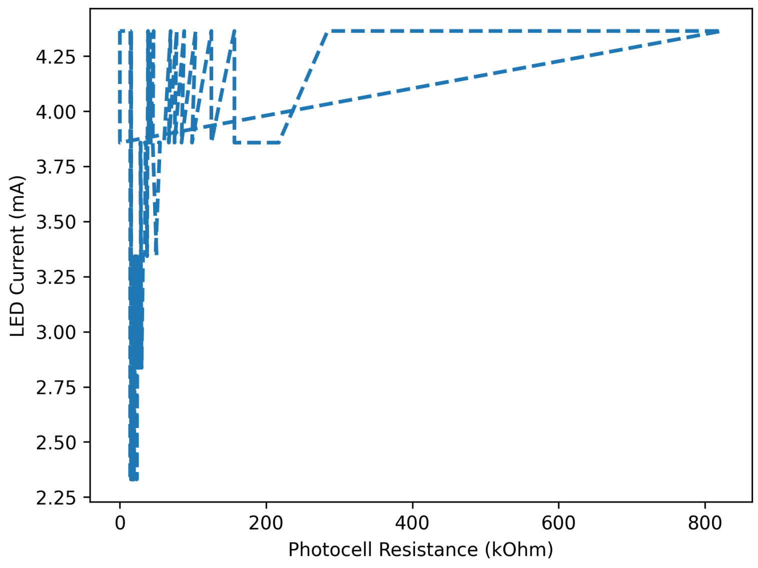

Figure 23: Yellow Photocell Resistance Versus LED Current

which shows photocell resistance versus LED current which is different than last time due to the fact that the LED current isn't constant because of the

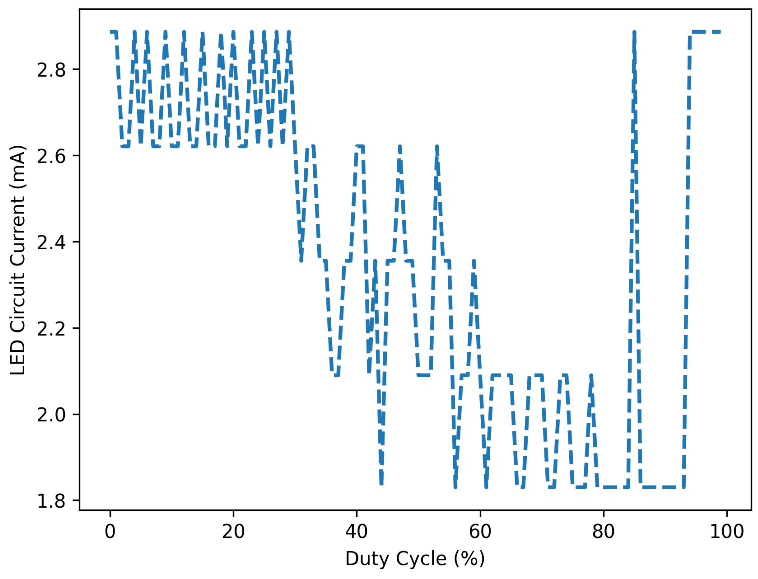

Figure 24: Blue Duty Cycle Versus LED Circuit Current

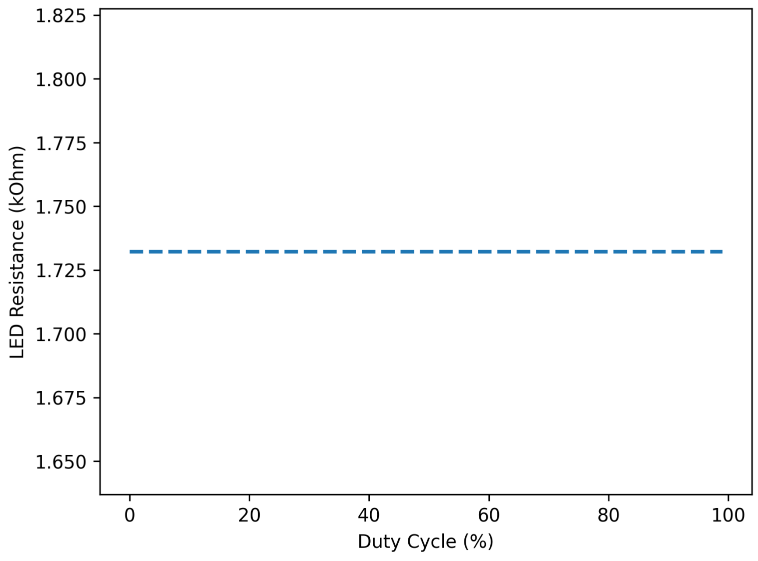

Figure 25: Blue Duty Cycle Versus LED Resistance

decreased step size causing it to be jumpier.

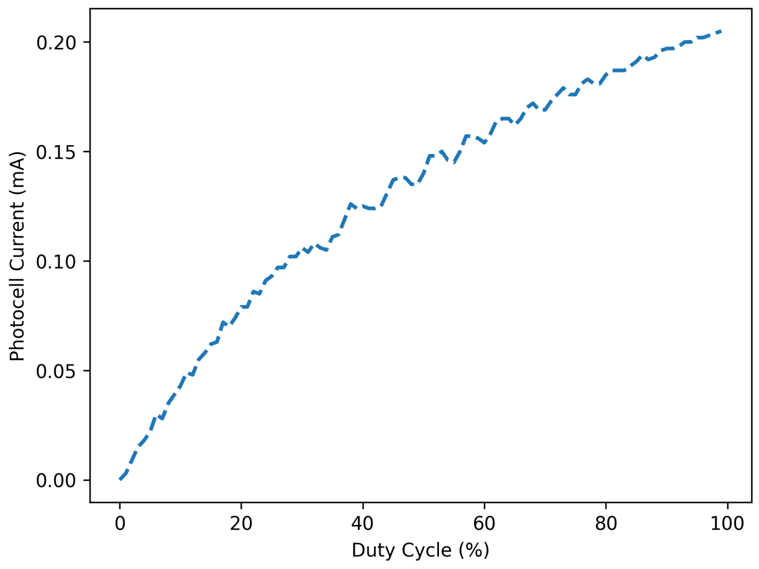

Figure 26: Blue Duty Cycle Versus Photocell Current

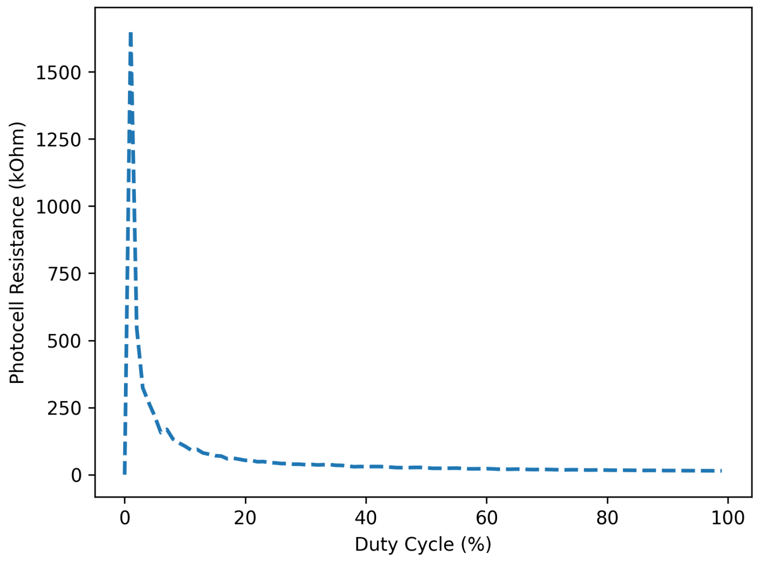

Figure 27: Blue Duty Cycle Versus Photocell Resistance

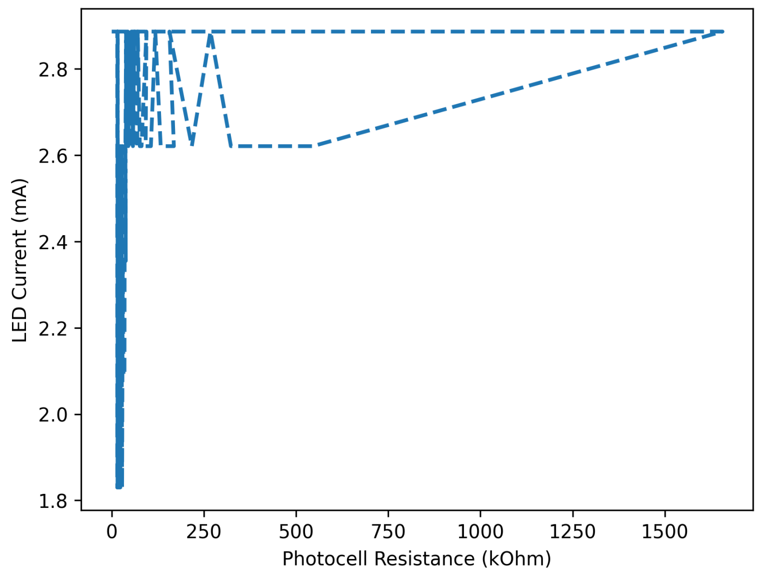

Figure 28: Blue Photocell Resistance Versus LED Current

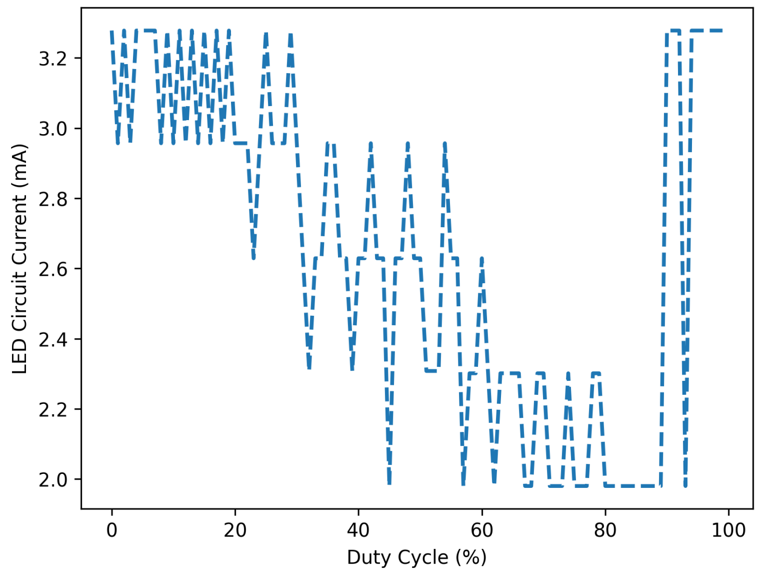

Figure 29: Green Duty Cycle Versus LED Circuit Current

## 4.2 Hardware

Figure 8 shows the setup of the circuit. The setup of the hardware was very simple, but could've been improved with a housing that blocks out light. You

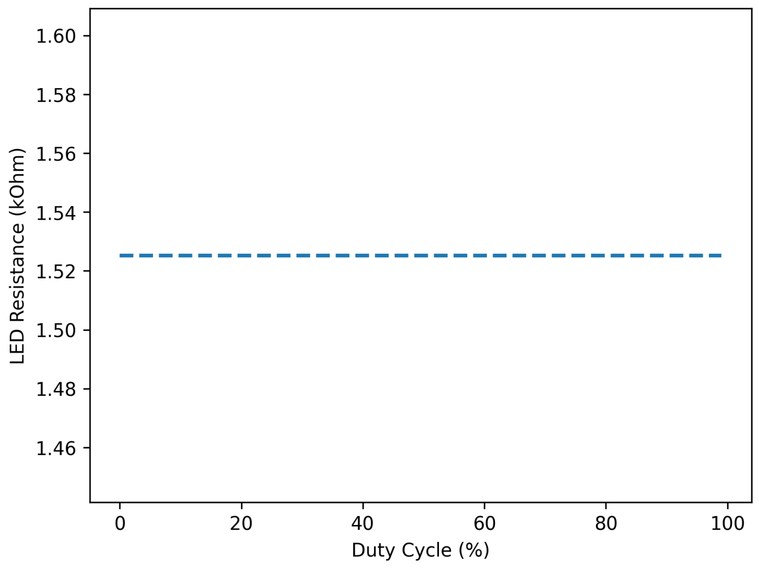

Figure 30: Green Duty Cycle Versus LED Resistance

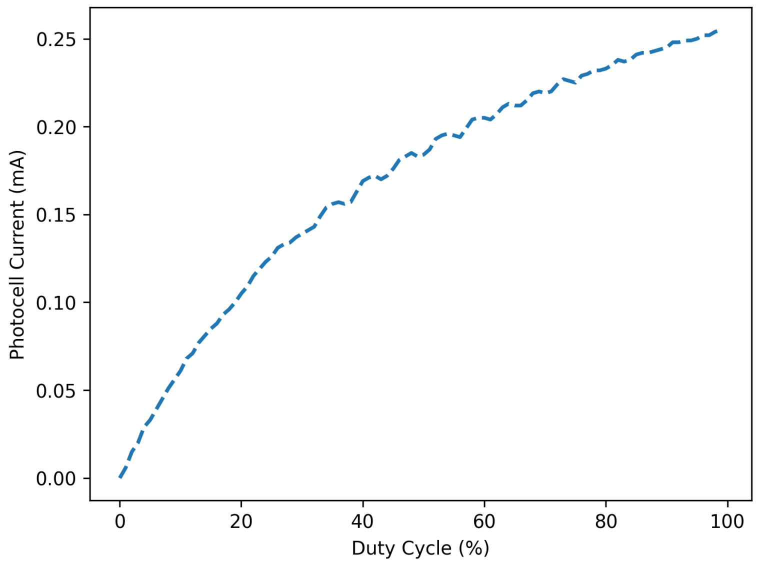

Figure 31: Green Duty Cycle Versus Photocell Current

could also get different results depending on the distance the photocell was from the LED, which could lead to skewed results when swapping the LED

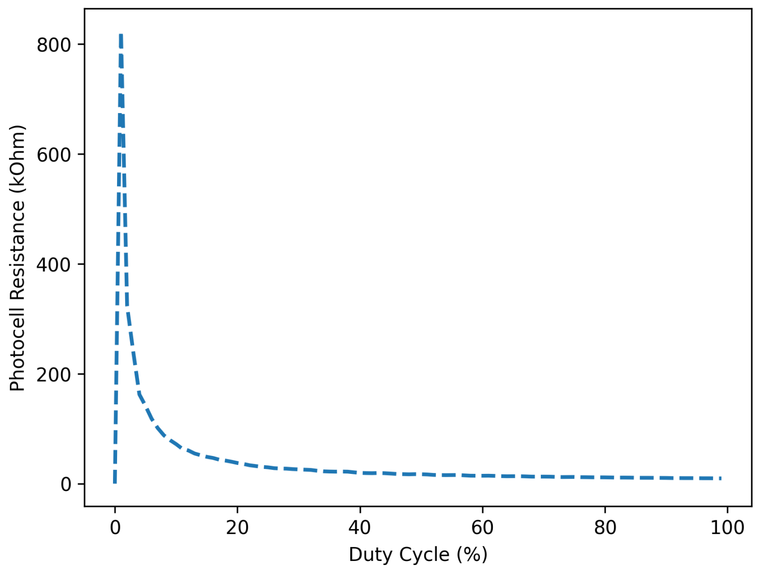

Figure 32: Green Duty Cycle Versus Photocell Resistance

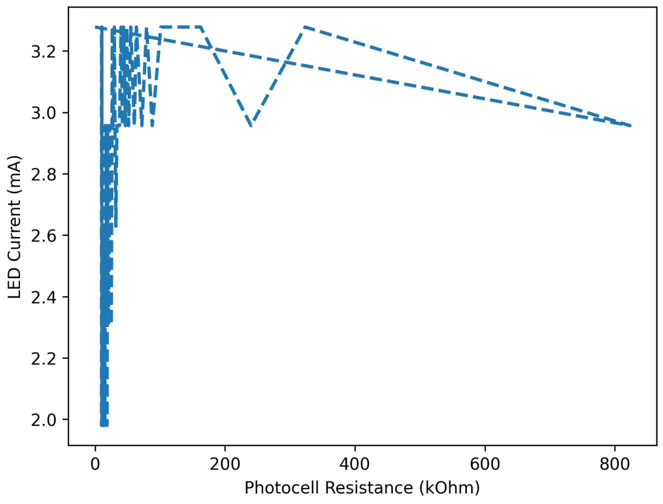

Figure 33: Green Photocell Resistance Versus LED Current

out for another color.

# 5 Lab 5

## 5.1 Discussion

This lab was used to introduce us to building a graphical user interface, or GUI, to assist with testing to not only make it easier, but also more user friendly. Using the base python setup from lab 4, the GUI was built by moving the previous contents of main into its own function and putting all the GUI setup in main. Figure 34 shows the GUI built with Tkinter with the graphs only appearing after the test is finished. There was a concern that the button could be hit multiple times and cause some unknown behavior, so it is disable after the test starts and enables when it finishes. The duty cycle label counts up to show the current duty cycle and the annoyance of the GUI freezing due to the blocking nature of the button function is no more as it is multithreaded. The results for this lab, like the last one, were a bit worse due to lights coming from my computer. I covered it with a box, but there was bound to be some leakage. It would have been better if I could isolate my circuit in a container and have my photocell the perfect distance from the LED. The values of the voltage, current, and resistance

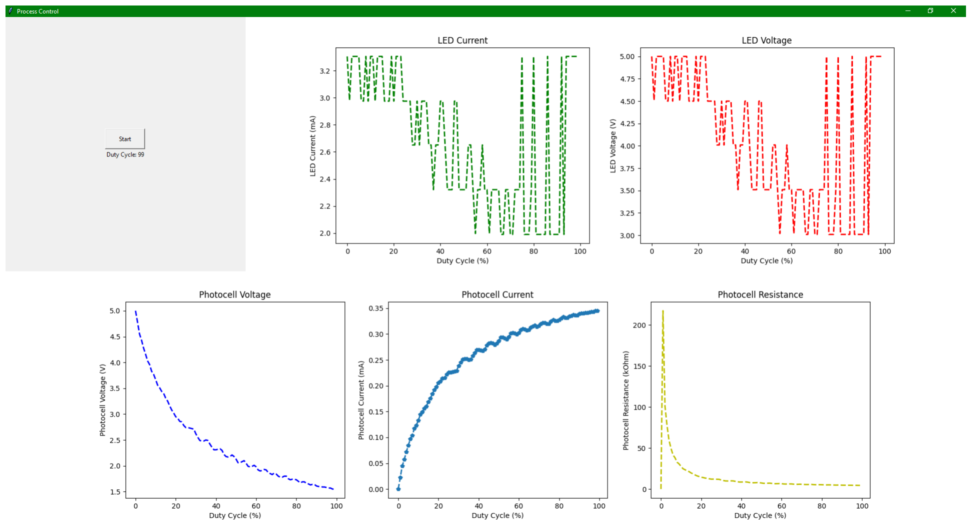

Figure 34: GUI After Green LED Test

were found using the same method as lab 2. An error in the values read from pin A0 was introduced due to the fact that we were reading it from a PWM source. In the duty cycle versus LED voltage and duty cycle versus LED current graphs, there are more samples being taken which is why you can see the general PWM wave. In the duty cycle versus LED voltage graph, there is the expected behavior of starting high and decreasing as the photocell resistance decreases. The duty cycle versus current graph shows an increase in current as time goes on which is also to be expected. You can see tiny oscillations in the different graphs which is theorized to be caused by the timer behind the analog write not being fast enough to smooth the curves well. The duty cycle versus photocell resistance graph mimics the photocell voltage graph because it is what you would get if you divide the current graph from the voltage graph as that's how it was generated. Another thing to keep note of is that the graphs have very different scales from one another so while the numbers may seem similar in size, they aren't.

## 5.2 Hardware

Figure 8 shows the setup of the circuit. The setup of the hardware was the same simple desing as the previous labs and again could've been improved with a housing that blocks out light. You could also get different results depending on the distance the photocell was from the LED, which could lead to skewed results.

# Appendices

## A Program Code

### A.1 Lab 1 Code

```
int led = 13;

void setup()
{
        pinMode(led, OUTPUT);
}

void loop()
{
        digitalWrite(led, HIGH);
        delay(1000);
        digitalWrite(led, LOW);
        delay(1000);
}
```

Listing 1: Lab 1 Arduino Code- Blinky

```
#include <Arduino.h>

int led = 13;
int intensity = 0;

void setup()
{
    pinMode(led, OUTPUT);
    analogWrite(led, LOW);
    Serial.begin(9600);
    Serial.print("Please enter a number from 0 to 255: ");
}

void loop()
{
    if(Serial.available() > 0)
    {
        intensity = Serial.parseInt();
        Serial.print("\nGot number: ");
        Serial.println(intensity, DEC);
        analogWrite(led, intensity);
        Serial.print("Please enter a number from 0 to 255: ")
            ;
    }
}
```

Listing 2: Lab 1 Arduino Code- Variable Brightness

## A.2 Lab 2 Code

```
#include <Arduino.h>

const int leds[] = {11, 5, 6, 44};
long password;
uint8_t correct_nums = 0b0000;
uint8_t num_tries = 0;
// = (16 * 10^6) / (1024 * 0.5) - 1
uint16_t starting_freq = 31248;
bool locked_out = false;

void setup()
{
    noInterrupts();
    for(int i = 0; i < 4; i++)
    {
        pinMode(leds[i], OUTPUT);
        digitalWrite(leds[i], LOW);
    }

    // Setup timer 1 pin 11 channel A
    TCCR1A = 0;
    TCCR1B = 0;
    TIMSK1 = 0;
    TCNT1  = 0;
    OCR1A  = starting_freq;
    TCCR1B |= (1 << WGM12);
    // 1024 prescalar
    TCCR1B |= (1 << CS12) | (0 << CS11) | (1 << CS10);
    TIMSK1 |= (1 << OCIE1A);

    // Setup timer 3 pin 5 channel A
    TCCR3A = 0;
    TCCR3B = 0;
    TIMSK3 = 0;
    TCNT3  = 0;
    OCR3A  = starting_freq;
    TCCR3B |= (1 << WGM32);
    TCCR3B |= (1 << CS32) | (0 << CS31) | (1 << CS30);
    TIMSK3 |= (1 << OCIE3A);

    // Setup timer 4 pin 6 channel A
    TCCR4A = 0;
    TCCR4B = 0;
    TIMSK4 = 0;
    TCNT4  = 0;
    OCR4A  = starting_freq;
    TCCR4B |= (1 << WGM42);
    TCCR4B |= (1 << CS42) | (0 << CS41) | (1 << CS40);
    TIMSK4 |= (1 << OCIE4A);

    // Setup timer 5 pin 44 channel A
    TCCR5A  = 0;
    TCCR5B  = 0;
    TIMSK5  = 0;
    TCNT5   = 0;
    OCR5A   = starting_freq;
    TCCR5B |= (1 << WGM52);
    TCCR5B |= (1 << CS52) | (0 << CS51) | (1 << CS50);
    TIMSK5 |= (1 << OCIE5A);

    Serial.begin(9600);
    randomSeed(analogRead(0));
    password = random(10000);
    Serial.print("Password is ");
    Serial.println(password, DEC);
    Serial.println("Please enter guess:");
    interrupts();
}

void loop()
{
    if(num_tries < 5)
    {
        if(Serial.available() > 0)
        {
            int guess = Serial.parseInt();
            Serial.print("Guess is ");
            Serial.println(guess, DEC);
            int temp_password = password;
            for(int i = 0; i <= 3; ++i)
            {
                if(guess % 10 == temp_password % 10)
                {
                    correct_nums |= 1 << i;
                    switch(i)
                    {
                        case 0: TIMSK1 = 0;
                        case 1: TIMSK3 = 0;
                        case 2: TIMSK4 = 0;
                        case 3: TIMSK5 = 0;
                    }
                    digitalWrite(leds[i], LOW);
                }
                guess = guess / 10;
                temp_password = temp_password / 10;
            }

            TCNT1 = 0;
            TCNT3 = 0;
            TCNT4 = 0;
            TCNT5 = 0;
            OCR1A = OCR1A / 2;
            OCR3A = OCR3A / 2;
            OCR4A = OCR4A / 2;
            OCR5A = OCR5A / 2;
            if(num_tries < 4)
                Serial.println("Please enter guess: ");
            num_tries++;
        }
    }
    else if(!locked_out)
    {
        TIMSK1 = 0;
        TIMSK3 = 0;
        TIMSK4 = 0;
        TIMSK5 = 0;
        Serial.println("Out of tries!");
        for(int i = 0; i < 4; i++)
        {
            if((correct_nums & (1 << i)) == 0b0000)
                digitalWrite(leds[i], HIGH);
        }
            locked_out = true;
    }

}

ISR(TIMER1_COMPA_vect)
{
    digitalWrite(leds[0], !digitalRead(leds[0]));
}

ISR(TIMER3_COMPA_vect)
```

```
{
    digitalWrite(leds[1], !digitalRead(leds[1]));
}

ISR(TIMER4_COMPA_vect)
{
    digitalWrite(leds[2], !digitalRead(leds[2]));
}

ISR(TIMER5_COMPA_vect)
{
    digitalWrite(leds[3], !digitalRead(leds[3]));
}
```

Listing 3: Lab 2 Arduino Code

## A.3 Lab 3 Code

```
#include <Arduino.h>

int led = 6;

void setup()
{
    pinMode(led, OUTPUT);
    pinMode(A0, INPUT);
    pinMode(A1, INPUT);

    noInterrupts();
    // Setup fast PWM timer 4 channel A pin 6
    TCCR4A = 0;
    TCCR4B = 0;
    TIMSK4 = 0;
    TCNT4  = 0;
    ICR4   = 12500;
    OCR4A  = 625;
    TCCR4A |= (1 << WGM41);
    TCCR4A |= (1 << COM4A1);
    TCCR4B |= (1 << WGM43);
    TCCR4B |= (0 << CS41) | (1 << CS40);

    Serial.begin(9600);
    interrupts();
}

```

```
void loop()
{
    if(OCR4A <= 12500)
    {
        delay(2000);
        Serial.println("Duty Cycle, LED Resistor Voltage,
           Photocell Resistor Voltage");

        int duty_cycle = (int)((float)OCR4A / (float)ICR4 *
           100);
        Serial.println(duty_cycle);

        int led_resistor_value = analogRead(A0);
        float led_resistor_voltage = (float)
           led_resistor_value * (5.0 / 1023.0);
        Serial.println(led_resistor_voltage);

        int photo_resistor_value = analogRead(A1);
        float photo_resistor_voltage = (float)
           photo_resistor_value * (5.0 / 1023.0);
        Serial.println(photo_resistor_voltage);
        Serial.println();
        OCR4A += 625;
    }
}
```

Listing 4: Lab 3 Arduino Code

### A.4 Lab 4 Code

```
#include <Arduino.h>
#include <string.h>

int led = 13;
int start_run = 0;
int duty_cycle = 0;

float calcValue(const uint8_t port);

void setup()
{
    pinMode(led, OUTPUT);
    pinMode(A0, INPUT);
    pinMode(A1, INPUT);
    analogWrite(led, LOW);
    Serial.begin(9600);
}

void loop()
{
    if(Serial.available() > 0)
    {
        String ser = Serial.readString();
        start_run = 1;
        if(Serial.readString() == "start")
        {
            start_run = 1;
            analogWrite(led, duty_cycle);
            duty_cycle += 1;
        }
    }

    if(start_run && duty_cycle < 100)
    {
        float led_resistor_voltage = calcValue(A0) * (5.0 / 1023.0);
        float photo_resistor_voltage = calcValue(A1) * (5.0 / 1023.0);

        Serial.println("Duty Cycle, LED Resistor Voltage, Photocell Resistor Voltage");
        Serial.println(duty_cycle);
        Serial.println(led_resistor_voltage);
        Serial.println(photo_resistor_voltage);
        Serial.println();

        duty_cycle += 1;
        analogWrite(led, 255 * ((float)duty_cycle / 100.0));
    }
    else if(start_run)
    {
        Serial.println("done");
        duty_cycle = 0;
        start_run = 0;
    }
}

float calcValue(const uint8_t port)
{
    int good_vals[5];
    int values[5];
    double avg_val = 0;
    // Get values and find avg
    for(int i = 0; i < 5; i++)
    {
        delay(1000);
        int val = analogRead(port);
        values[i] = val;
        avg_val += (double)val / 5.0;
    }

    // Calculate standard deviation
    double std_dev = 0;
    for(int i = 0; i < 5; i++)
        std_dev += pow(values[i] - avg_val, 2);
    std_dev = sqrt(std_dev / 5.0);

    int good_vals_idx = 0;
    for(int i = 0; i < 5; i++)
    {
        if((values[i] < avg_val + 2*std_dev) || (values[i] >
            avg_val - 2*std_dev))
        {
            good_vals[good_vals_idx] = values[i];
            good_vals_idx++;
        }
    }

    float final_value = 0;
    for(int i = 0; i < good_vals_idx; i++)
        final_value += (float)good_vals[i];

    if(good_vals_idx == 0)
        final_value = 0;
    else
        final_value /= (float)good_vals_idx;

    return final_value;
}
```

Listing 5: Lab 4 Arduino Code

```
import serial
from tkinter import *
import matplotlib.pyplot as plt
```

```python
import numpy as np
import time
from itertools import zip_longest

duty_cycles = []
led_voltages = []
led_currents = []
photocell_voltages = []
photocell_currents = []
photocell_resistances = []

def main():
    global duty_cycles, led_voltages, photocell_voltages
    duty_cycles.clear()
    led_voltages.clear()
    photocell_voltages.clear()
    with serial.Serial("COM6", 9600, timeout=0.1) as ser:
        time.sleep(2)
        print("Connection to Arduino established!")
        ser.write(b"start\r\n")
        title = ""
        duty_cycle = 0
        led_voltage = 0.0
        photocell_voltage = 0.0
        while title != "done":
            title = ser.readline().decode("utf-8").strip()
            if title == "Duty Cycle, LED Resistor Voltage, Photocell Resistor Voltage":
                duty_cycle = int(ser.readline().decode("utf-8").strip())
                led_voltage = float(ser.readline().decode("utf-8").strip())
                photocell_voltage = float(ser.readline().decode("utf-8").strip())
                duty_cycles.append(duty_cycle)
                led_voltages.append(led_voltage)
                photocell_voltages.append(photocell_voltage)

            if title:
                print("Title", title)
                print(duty_cycle)
                print(led_voltage)
                print(photocell_voltage)
    plot_graphs()

```

```
def plot_graphs():
    global duty_cycles, led_voltages, led_currents,
        photocell_voltages, photocell_currents,
        photocell_resistances
    led_resistance = 5.0 / max(led_voltages) - 1.0
    led_voltages = [ 5.0 - voltage for voltage in
        led_voltages ]
    led_currents = [ voltage / led_resistance for voltage in
        led_voltages ]
    photocell_resistances = [ 50.0 / voltage - 10.0 if
        voltage != 0 else 0.0 for voltage in
        photocell_voltages ]
    photocell_voltages = [ 5.0 - voltage for voltage in
        photocell_voltages]
    photocell_currents = [ voltage / resistance if resistance
         != 0 else 0.0 for voltage, resistance in zip_longest(
        photocell_voltages, photocell_resistances) ]

    color = "red"
    plt.plot(duty_cycles, led_currents, '--', linewidth=2)
    plt.xlabel("Duty Cycle (%)")
    plt.ylabel("LED Circuit Current (mA)")
    plt.savefig(color + "_duty_cycle_led_circuit_curr.png",
        dpi=300, bbox_inches="tight")

    plt.figure()
    plt.plot(duty_cycles, [led_resistance] * len(duty_cycles)
        , '--', linewidth=2)
    plt.xlabel("Duty Cycle (%)")
    plt.ylabel("LED Resistance (kOhm)")
    plt.savefig(color + "_duty_cycle_led_res.png", dpi=300,
        bbox_inches="tight")

    plt.figure()
    plt.plot(duty_cycles, photocell_resistances, '--',
        linewidth=2)
    plt.xlabel("Duty Cycle (%)")
    plt.ylabel("Photocell Resistance (kOhm)")
    plt.savefig(color + "_duty_cycle_photo_res.png", dpi=300,
         bbox_inches="tight")

    plt.figure()
    plt.plot(duty_cycles, photocell_currents, '--', linewidth
        =2)
    plt.xlabel("Duty Cycle (%)")
```

```python
    plt.ylabel("Photocell Current (mA)")
    plt.savefig(color + "_duty_cycle_photo_curr.png", dpi
        =300, bbox_inches="tight")

    plt.figure()
    plt.plot(photocell_resistances, led_currents, '--',
        linewidth=2)
    plt.ylabel("LED Current (mA)")
    plt.xlabel("Photocell Resistance (kOhm)")
    plt.savefig(color + "_photo_res_led_curr.png", dpi=300,
        bbox_inches="tight")

    plt.show()

if __name__ == "__main__":
    main()
```

Listing 6: Lab 4 Python Code

## A.5 Lab 5 Code

```c
#include <Arduino.h>
#include <string.h>

int led = 13;
int start_run = 0;
int duty_cycle = 0;

float calcValue(const uint8_t port);

void setup()
{
    pinMode(led, OUTPUT);
    pinMode(A0, INPUT);
    pinMode(A1, INPUT);
    analogWrite(led, LOW);
    Serial.begin(9600);
}

void loop()
{
    if(Serial.available() > 0)
    {
        String ser = Serial.readString();
        start_run = 1;
        if(Serial.readString() == "start")
        {
            start_run = 1;
            analogWrite(led, duty_cycle);
            duty_cycle += 1;
        }
    }

    if(start_run && duty_cycle < 100)
    {
        float led_resistor_voltage = calcValue(A0) * (5.0 / 1023.0);
        float photo_resistor_voltage = calcValue(A1) * (5.0 / 1023.0);

        Serial.println("Duty Cycle, LED Resistor Voltage, Photocell Resistor Voltage");
        Serial.println(duty_cycle);
        Serial.println(led_resistor_voltage);
        Serial.println(photo_resistor_voltage);
        Serial.println();

        duty_cycle += 1;
        analogWrite(led, 255 * ((float)duty_cycle / 100.0));
    }
    else if(start_run)
    {
        Serial.println("done");
        duty_cycle = 0;
        start_run = 0;
    }
}

float calcValue(const uint8_t port)
{
    int good_vals[5];
    int values[5];
    double avg_val = 0;
    // Get values and find avg
    for(int i = 0; i < 5; i++)
    {
        delay(1000);
        int val = analogRead(port);
        values[i] = val;
        avg_val += (double)val / 5.0;
    }

    // Calculate standard deviation
    double std_dev = 0;
    for(int i = 0; i < 5; i++)
        std_dev += pow(values[i] - avg_val, 2);
    std_dev = sqrt(std_dev / 5.0);

    int good_vals_idx = 0;
    for(int i = 0; i < 5; i++)
    {
        if((values[i] < avg_val + 2*std_dev) || (values[i] >
            avg_val - 2*std_dev))
        {
            good_vals[good_vals_idx] = values[i];
            good_vals_idx++;
        }
    }

    float final_value = 0;
    for(int i = 0; i < good_vals_idx; i++)
        final_value += (float)good_vals[i];

    if(good_vals_idx == 0)
        final_value = 0;
    else
        final_value /= (float)good_vals_idx;

    return final_value;
}
```

Listing 7: Lab 5 Arduino Code

```
import serial
from tkinter import *
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import (
    FigureCanvasTkAgg, NavigationToolbar2Tk)
import numpy as np
import time
from itertools import zip_longest
import threading

duty_cycles = []
led_voltages = []
```

```python
led_currents = []
photocell_voltages = []
photocell_currents = []
photocell_resistances = []
gui = Tk()
duty_cycle_string = StringVar()

def main():
    gui.wm_title("Process Control")
    gui.geometry("1000x600")
    gui.grid_rowconfigure(0, weight=1)
    gui.grid_rowconfigure(1, weight=1)
    gui.grid_columnconfigure(0, weight=1)
    gui.grid_columnconfigure(1, weight=2)
    frame = Frame(gui)
    frame.grid(column = 0, row=0)
    global start_btn
    start_btn = Button(frame, text="Start", command=
        start_process_thread, height=2, width=10)
    start_btn.pack(side=TOP)
    label = Label(frame, textvariable=duty_cycle_string)
    label.pack(side=TOP)
    gui.mainloop()

def start_process_thread():
    global start_btn, process_thread, duty_cycle_string
    process_thread = threading.Thread(target=run_process,
        name="child")
    process_thread.daemon = True
    process_thread.start()
    gui.after(20, check_process)
    start_btn["state"] = DISABLED
    duty_cycle_string.set("Process Started")


def check_process():
    if process_thread.is_alive():
        gui.after(20, check_process)
    else:
        start_btn["state"] = NORMAL
        plot_graphs()

def run_process():
    global duty_cycles, led_voltages, photocell_voltages
    duty_cycles.clear()
    led_voltages.clear()
    photocell_voltages.clear()
    with serial.Serial("COM6", 9600, timeout=0.1) as ser:
        time.sleep(2)
        print("Connection to Arduino established!")
        ser.write(b"start\r\n")
        title = ""
        duty_cycle = 0
        led_voltage = 0.0
        photocell_voltage = 0.0
        while title != "done":
            title = ser.readline().decode("utf-8").strip()
            if title == "Duty Cycle, LED Resistor Voltage, Photocell Resistor Voltage":
                global duty_cycle_string
                duty_cycle = int(ser.readline().decode("utf-8").strip())
                duty_cycle_string.set("Duty Cycle: " + str(duty_cycle))
                led_voltage = float(ser.readline().decode("utf-8").strip())
                photocell_voltage = float(ser.readline().decode("utf-8").strip())
                duty_cycles.append(duty_cycle)
                led_voltages.append(led_voltage)
                photocell_voltages.append(photocell_voltage)

            if title:
                print("Title", title)
                print(duty_cycle)
                print(led_voltage)
                print(photocell_voltage)

def plot_graphs():
    global led_voltages, led_currents, photocell_voltages, photocell_currents, photocell_resistances
    led_resistance = 5.0 / max(led_voltages) - 1.0
    led_voltages = [ 5.0 - voltage for voltage in led_voltages ]
    led_currents = [ voltage / led_resistance for voltage in led_voltages ]
    photocell_resistances = [ 50.0 / voltage - 10.0 if voltage != 0 else 0.0 for voltage in photocell_voltages ]
    photocell_voltages = [ 5.0 - voltage for voltage in
```

```
        photocell_voltages]
    photocell_currents = [ voltage / resistance if resistance
         != 0 else 0.0 for voltage, resistance in zip_longest(
        photocell_voltages, photocell_resistances) ]

    fig, axs = plt.subplots(1, 2)
    axs[0].plot(duty_cycles, led_currents, '--g', linewidth
        =2)
    axs[0].set(title="LED Current", xlabel="Duty Cycle (%)",
        ylabel="LED Current (mA)")
    axs[1].plot(duty_cycles, led_voltages, '--r', linewidth
        =2)
    axs[1].set(title="LED Voltage", xlabel="Duty Cycle (%)",
        ylabel="LED Voltage (V)")
    canvas = FigureCanvasTkAgg(fig, master=gui)
    canvas.draw()
    canvas.get_tk_widget().grid(row=0, column=1, sticky=W+E+N
        +S)

    fig, axs = plt.subplots(1, 3)
    axs[0].set(title="Photocell Voltage", xlabel="Duty Cycle
        (%)", ylabel="Photocell Voltage (V)")
    axs[0].plot(duty_cycles, photocell_voltages, '--b',
        linewidth=2)
    axs[1].set(title="Photocell Current", xlabel="Duty Cycle
        (%)", ylabel="Photocell Current (mA)")
    axs[1].plot(duty_cycles, photocell_currents, '--p',
        linewidth=2)
    axs[2].set(title="Photocell Resistance", xlabel="Duty
        Cycle (%)", ylabel="Photocell Resistance (kOhm)")
    axs[2].plot(duty_cycles, photocell_resistances, '--y',
        linewidth=2)
    canvas = FigureCanvasTkAgg(fig, master=gui)
    canvas.draw()
    canvas.get_tk_widget().grid(row=1, column=0, columnspan
        =2, sticky=W+E+N+S)

if __name__ == "__main__":
    main()
```

Listing 8: Lab 5 Python Code