Drexel University

To: Dr. Peters

From: Tyler Ostinato

Date: October 16, 2021

Re: ECE 303-Lab 4-Serial Communications

Purpose

Learn how to make MATLAB and Arduino communicate via the serial terminal. Using this communication we will be able to automatically graph our data in MATLAB.

Discussion

Increment Duty Cycle | 1.1.1

In lab 3 we were tasked with creating a PWM signal to create a duty cycle that would change the effective voltage across an LED and thus change the LED's intensity. In this lab I was able to use analogWrite() to set the duty cycle of the LED. The analog pin only accepts integers 0-255, there for to get 100 increments we use 255/100 = 2.55. Increasing the analog input by 2.55 will roughly get us 1% increments in the duty cycle (Figure 1).

```
// Get 10 samples into an array
while(count_samples < 10){
  val=Serial.parseInt();
  analogWrite(5,counter); // Change duty cycle
  val1 = analogRead(photo_pin);
  integer_arr[count_samples] = val1;
  count_samples++;
  mean+= val1;
}
counter+=2.55; // Increase duty cycle by 1%
```

Figure 1: Using analogWrite() to increment duty cycle in 1% increments.

Outlier Detection | 1.1.2

Some of the voltage readings may vary when reading the ADC. To account for this issue, I have added logic to compute the standard deviation of 10 samples at each duty cycle and remove any values that are greater than 1 standard deviation from the mean value (Figure 2).

```
// Calculate standard deviation and remove values that fall outside of it
mean = mean/10;
// Calculate the variance
float variance_total=0;
int variance_count=0;
while(variance_count < 10){
  int power_val = (integer_arr[variance_count] - mean) * (integer_arr[variance_count] - mean);
  variance_total += power_val;
  variance_count++;
}

// Get the standard deviation
variance_total = variance_total/10;
int sd = sqrt(variance_total);

// Remove values over 1 SD away from the mean
// Add remaining values and take the avg; this will be the avg of the valid sample data
int avg_of_sample=0;
int num_of_valid_samples=0;
for(int i=0; i<10; i++){
  if(abs(integer_arr[i] - mean) < variance_total){
    avg_of_sample += integer_arr[i];
    num_of_valid_samples++;
  }
}
```

Figure 2: Outlier detection to get rid of unreliable voltage readings across the photosensor.

Serial Communication | 1.1.3

MATLAB has the ability to read the serial terminal output from the Arduino. This gives the user the ability to read data from the Arduino and interpret that data directly into MATLAB. The first step is to tell MATLAB which communications port and baud rate to look at using serialport() (Figure 3). This will connect MATLAB to our serial terminal output. Next we need to transmit(TX) and receive(RX) data between MTALAB and the Arduino so the can properly communicate (Figure 4). It is important to communicate both was to ensure our data lines up properly.

```
% Set up communications
arduino=serialport("COM8",9600,"Timeout",15);
```

Figure 3: Setup communication between Arduino and MATLAB.

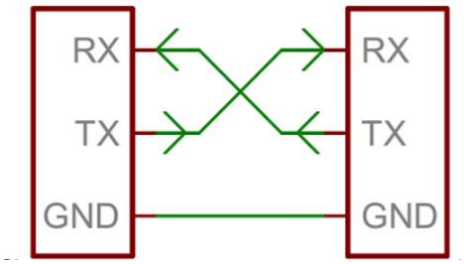


Figure 4: Schematic of MATLAB and Arduino sending and receiving information to communicate via the serial terminal.

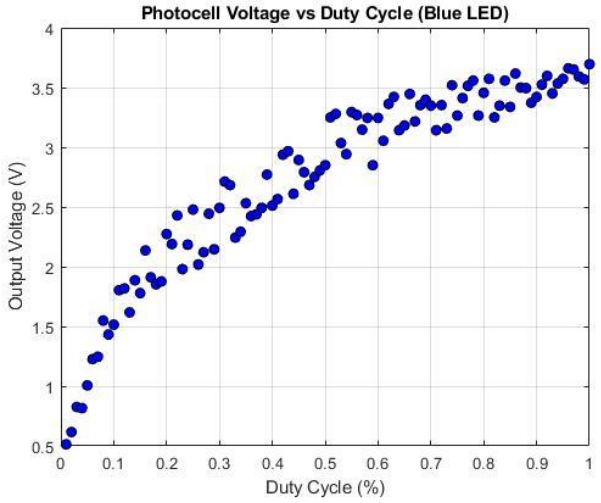
Plotting the Data | 1.1.4

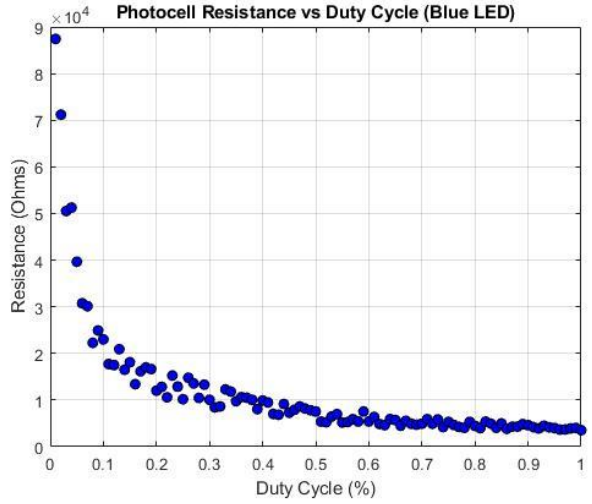
MATLAB can now read the output of the Arduino and begin to plot the information. Using the same formulas from lab 3 I was able to take the inputs in MATLAB and create 3 plots automatically (See Appendix).

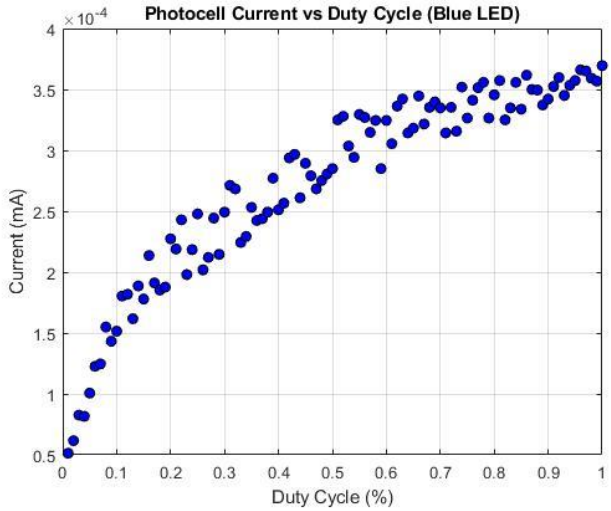
Conclusion

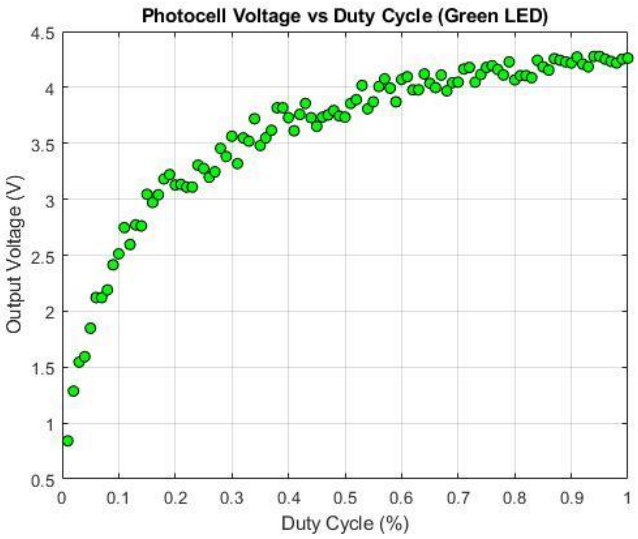
This lab demonstrated using the serial terminal of the Arduino to communicate with an external application. This proves useful as the Arduino can't do everything we need, and other applications may be required to do more complex calculations much easier. This saved me lots of time graphing the data as I didn't have to make many separate graphs. Once I created the graphs for one LED I was able to change the LED and run the code again, saving me lots of time.

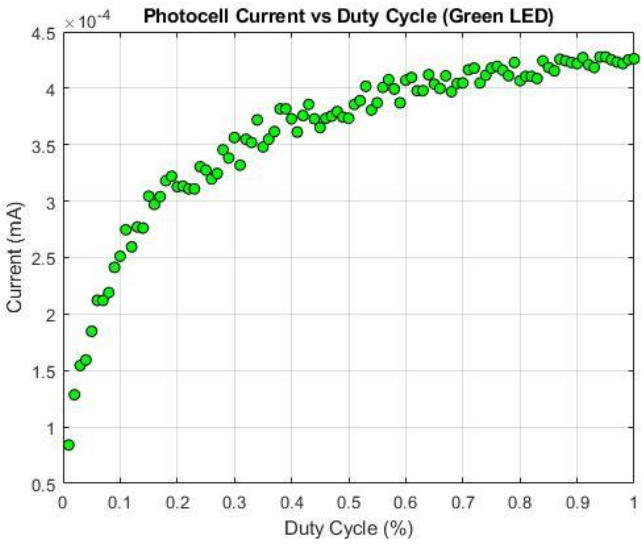
Appendix

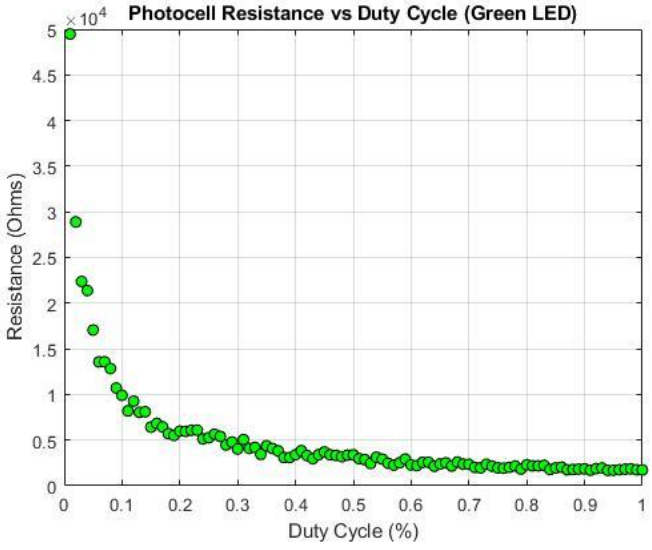

Photocell Voltage vs Duty Cycle (Blue LED)
Output Voltage (V)
Duty Cycle (%)

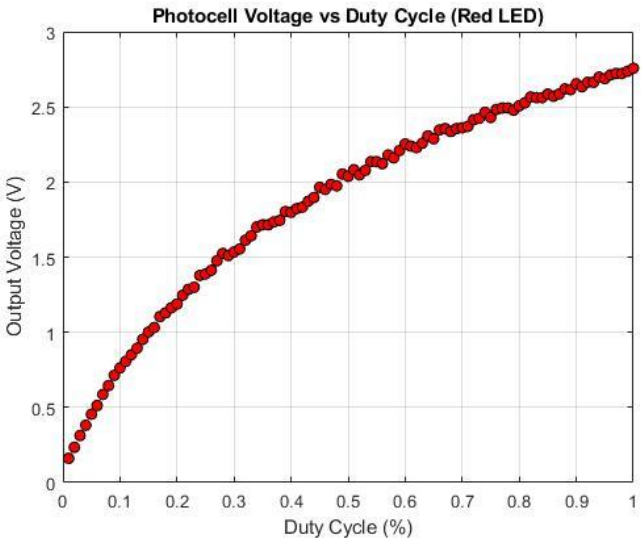

Photocell Resistance vs Duty Cycle (Blue LED)
×10^4
Resistance (Ohms)
Duty Cycle (%)

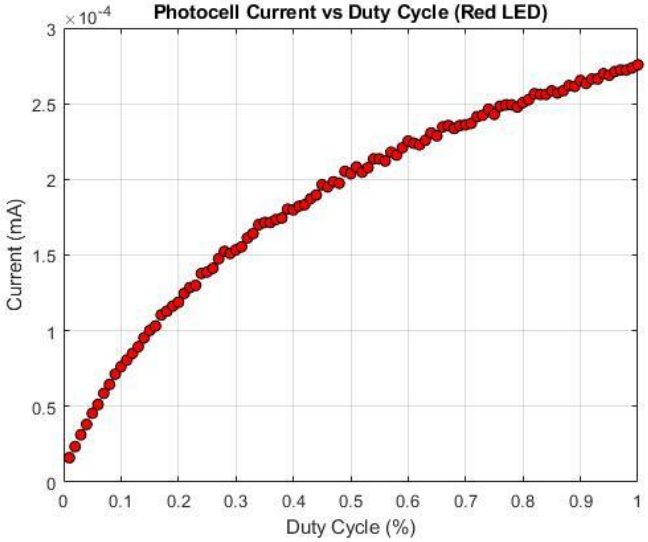

Photocell Current vs Duty Cycle (Blue LED)
×10^-4
Current (mA)
Duty Cycle (%)

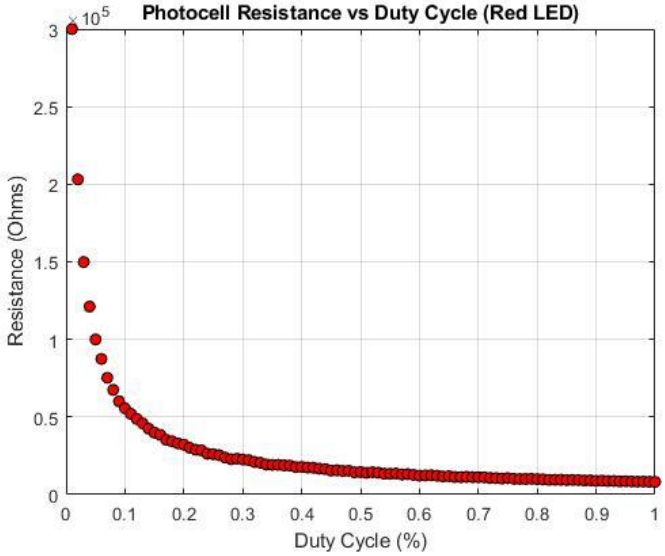

Photocell Voltage vs Duty Cycle (Green LED)
Output Voltage (V)
Duty Cycle (%)

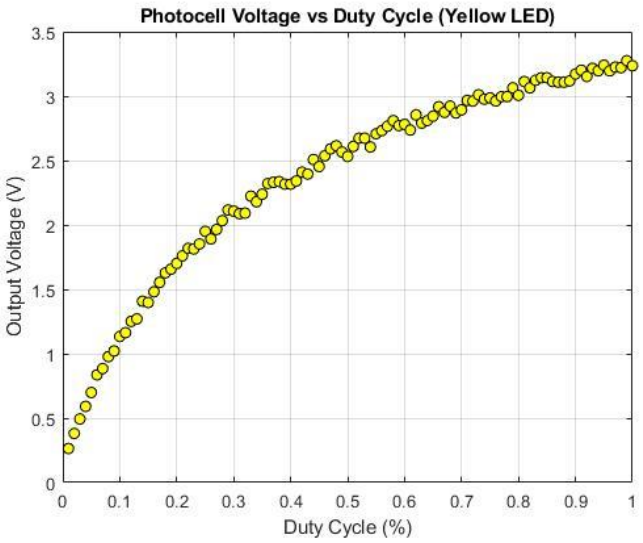

Photocell Current vs Duty Cycle (Green LED)
×10^-4
Current (mA)
Duty Cycle (%)

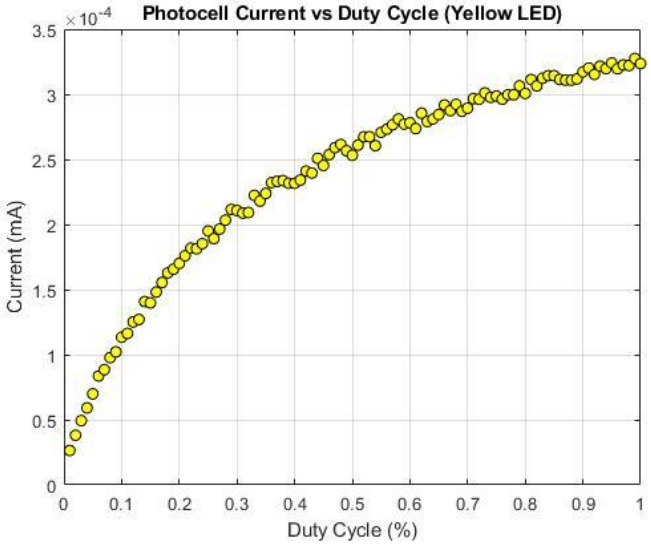

Photocell Resistance vs Duty Cycle (Green LED)
×10^4
Resistance (Ohms)
Duty Cycle (%)

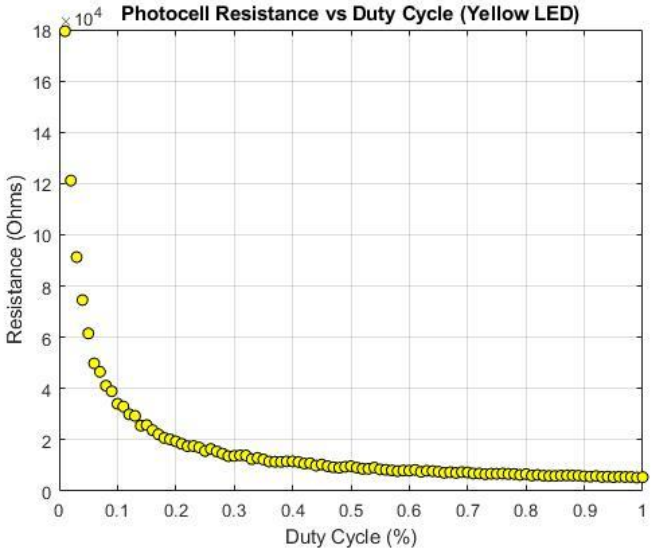

Photocell Voltage vs Duty Cycle (Red LED)
Output Voltage (V)
Duty Cycle (%)


Photocell Current vs Duty Cycle (Red LED)
×10^-4
Current (mA)
Duty Cycle (%)


Photocell Resistance vs Duty Cycle (Red LED)
×10^5
Resistance (Ohms)
Duty Cycle (%)


Photocell Voltage vs Duty Cycle (Yellow LED)
Output Voltage (V)
Duty Cycle (%)


Photocell Current vs Duty Cycle (Yellow LED)
×10^-4
Current (mA)
Duty Cycle (%)


Photocell Resistance vs Duty Cycle (Yellow LED)
×10^4
Resistance (Ohms)
Duty Cycle (%)

MATLAB Script

```
close all;clear all;clc
% Set up communications
arduino=serialport("COM8",9600,"Timeout",15);
pause(1)
x = 0:100;
y = zeros(1,101);
for K=0:100
flush(arduino)
write(arduino,2,'string')
pause(0.5)
a=read(arduino,4,'string');
flush(arduino)
y(K+1)=str2double(a);
disp([K,y(K+1)])
end
delete(arduino);
clear arduino;

figure
plot(x/100,y*5/1023,'ko','MarkerFaceColor','blue')
grid on
title('Photocell Voltage vs Duty Cycle (Blue LED)')
xlabel('Duty Cycle (%)')
ylabel('Output Voltage (V)')

figure
voltage = y*5/1023;
resistance = ((10000*5)./voltage) - 10000;
plot(x/100,resistance,'ko','MarkerFaceColor','blue')
grid on
title('Photocell Resistance vs Duty Cycle (Blue LED)')
xlabel('Duty Cycle (%)')
ylabel('Resistance (Ohms)')

figure
voltage = y*5/1023;
resistance = ((10000*5)./voltage) - 10000;
current = 5./(10000 + resistance);
plot(x/100,current,'ko','MarkerFaceColor','blue')
grid on
title('Photocell Current vs Duty Cycle (Blue LED)')
xlabel('Duty Cycle (%)')
ylabel('Current (mA)')
```

Arduino Sketch

```
int photo_pin(A0);
int val1=0;
unsigned int val=0;
unsigned int counter=0;
int integer_arr[10];

void setup() {
 // put your setup code here, to run once:
 Serial.begin(9600);
 pinMode(5,OUTPUT);
 Serial.setTimeout(100);
}

void loop() {

 int count_samples=0;
 int mean=0;
 // Get 10 samples into an array
 while(count_samples < 10){
  val=Serial.parseInt();
  analogWrite(5,counter); // Change duty cycle
  val1 = analogRead(photo_pin);
  integer_arr[count_samples] = val1;
  count_samples++;
  mean+= val1;
 }

 // Calculate standard deviation and remove values that fall outside of it
 mean = mean/10;
```

```
 // Calculate the variance
 float variance_total=0;
 int variance_count=0;
 while(variance_count < 10){
  int power_val = (integer_arr[variance_count] - mean) * (integer_arr[variance_count] - mean);
  variance_total += power_val;
  variance_count++;
 }

 // Get the standard deviation
 variance_total = variance_total/10;
 int sd = sqrt(variance_total);

 // Remove values over 1 SD away from the mean
 // Add remaining values and take the avg; this will be the avg of the valid sample data
 int avg_of_sample=0;
 int num_of_valid_samples=0;
 for(int i=0; i<10; i++){
  if(abs(integer_arr[i] - mean) < variance_total){
   avg_of_sample += integer_arr[i];
   num_of_valid_samples++;
  }
 }

 Serial.println(abs(avg_of_sample/num_of_valid_samples));

 counter+=2.55; // increment the LED duty cycle after getting samples
}
```