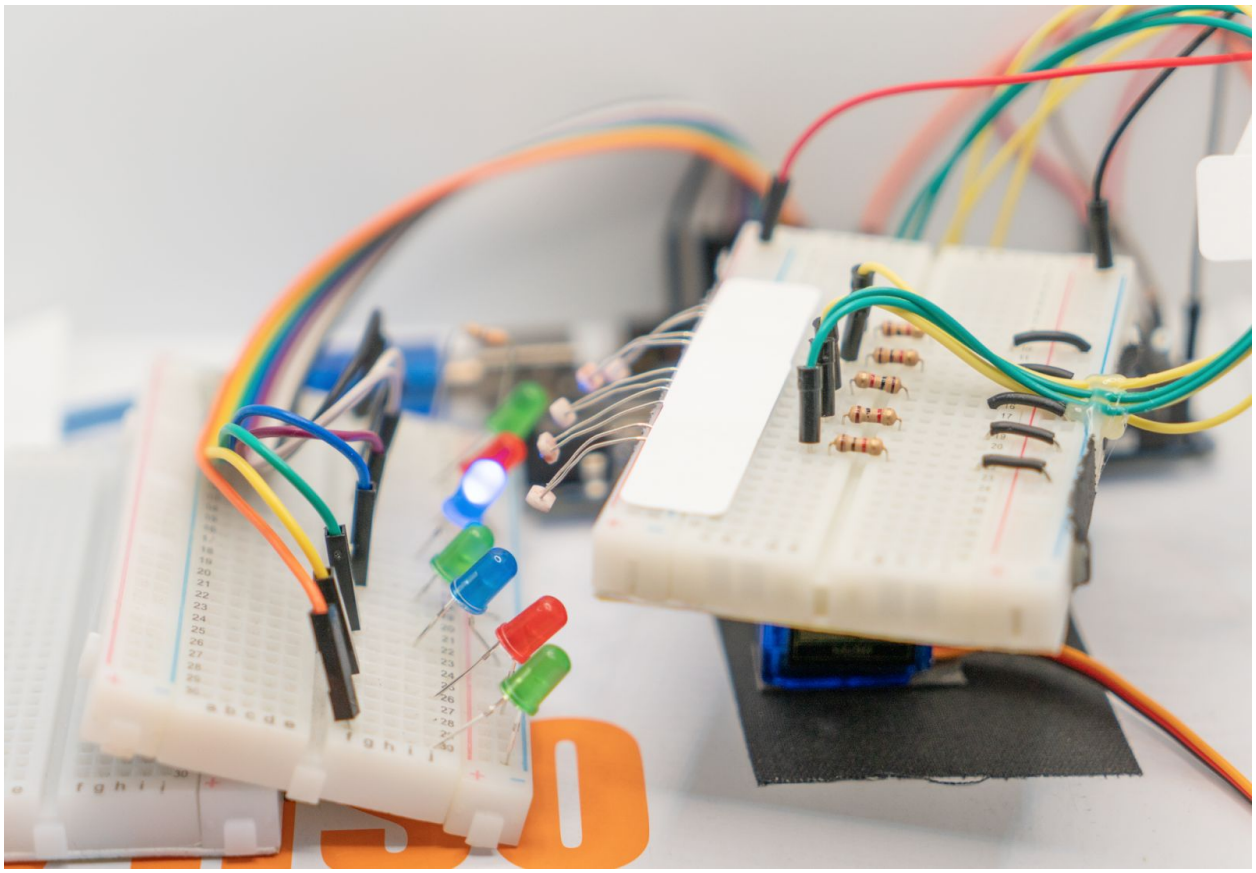
Light Following Arm | 4



Electrical and Computer Engineering 5

Light Following Arm

Developed by ECE 5 Spring 2020 Tutors/TA Team

Professor John Eldon | Professor Vikash Gilja | Professor Drew Hall | Professor Truong Nguyen

Overview

In this lab, you will be building a light following arm using much of the knowledge gained throughout the quarter with a better understanding of sensors, actuators, programming, systems and controls. You will be using photoresistors and PID controls to make the servo arm following lit up LEDs in a row of LEDs.

Important Note: Each challenge builds off the previous challenge, so don't take anything apart before you're done with all challenges!

This document is subject to change at any time to correct mistakes / make clarifications. Please access this from Google Docs directly instead of downloading it.

What You Will Need

Materials:

- 1 Arduino Mega 2560
- 7 LEDs
- 1 Resistor (330Ω)
- 5 Resistors (1kΩ)
- 5 Photoresistors (5528)
- 1 Servo
- Wires
- 2 Breadboards

Machinery:

- Computer / Laptop

Software:

- Arduino Software (IDE)
- MATLAB (>= R2017a)

Other:

- Tape

Challenge #1: Row of LEDs

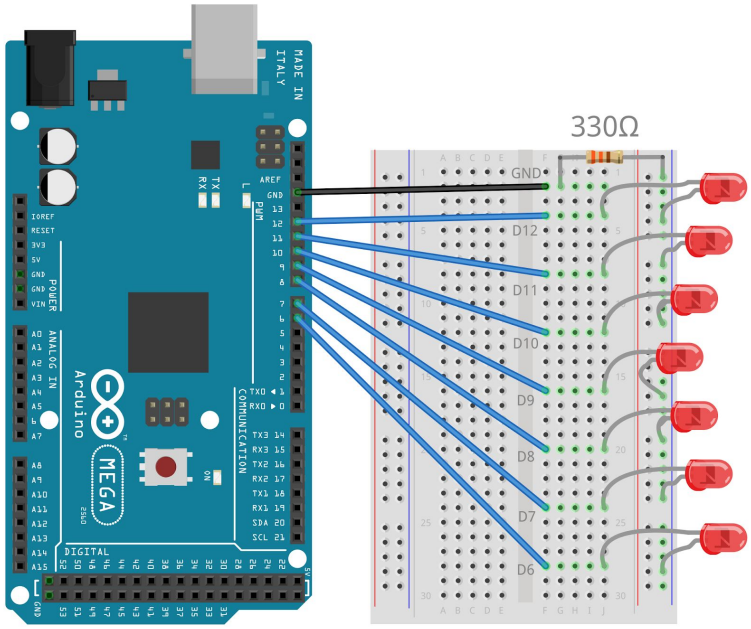
Objective

In this challenge, you will make a row of LEDs that the servo arm will point at. Follow the given circuit diagram and use the code to test to make sure all the LEDs are wired correctly and light up.

Components

- 1 Arduino Mega
- 7 LEDs
- 1 Resistor (330 ohms)
- 8 Wires
- 1 Breadboard

Circuit Diagram



Note: Try to use all the same color LEDs if possible (since different colors may have varying brightness). Arrange the LEDs in a slight arc (in the shape of the servo arm's path of motion).

Try not to use Yellow LED as it is very dim compared with other colors.

Task

- Set up the row of LEDs following the circuit diagram above.
- Test the row of LEDs by completing the following code to turn the 7 LEDs on-off one by one.

Code

```
#define NUM_OF_LEDS 7
int ledPin[] = {6, 7, 8, 9, 10, 11, 12};


void ledTest() {
  for (int i = 0; _____; i++) { // complete for loop
    digitalWrite(_____, _____);  // turn LED on
    delay(500);
    digitalWrite(_____, _____);  // turn LED off
  }
}

void setup() {
  for (int i = 0; _____; i++) { // complete for loop (same as before)
    pinMode(_____, _____); // define LED pinmode for each LED
  }
}

void loop() {
  ledTest();
}
```

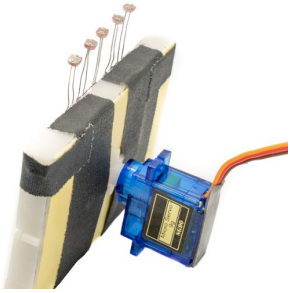
Challenge #2: Assembling the Servo Arm

Objective

In this challenge, you will attach the photoresistors onto the servo arm. You will be using a separate breadboard from the one with the row of LEDs (Challenge #1).

Components

- 5 Photoresistors
- 5 Resistors (1kΩ)
- 1 Servo
- Wires
- 1 breadboard
- Tape



Task

- Follow the instructions below to connect the servo, and assemble the servo arm.
- Test the servo arm by completing the following code.

Instructions

1. Hook up servo to the Arduino shown in **Fig 2.1**. Use Digital Pin 3 for the servo control.
2. Hook up the resistors and 5 ground wires on a breadboard shown in **Fig 2.2 below**. Space them evenly and use 1kΩ (brown-black-red). Don't worry about its connection to Arduino yet. (Will add those in the next challenge)

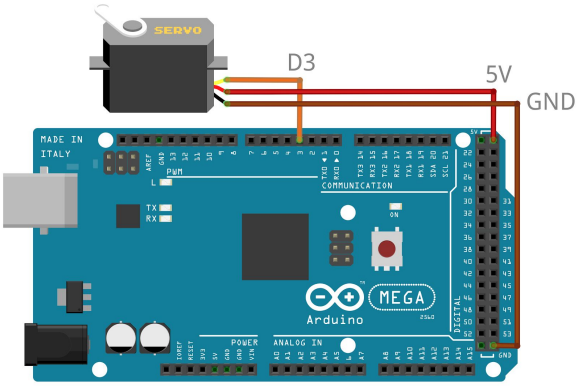


Figure 2.1

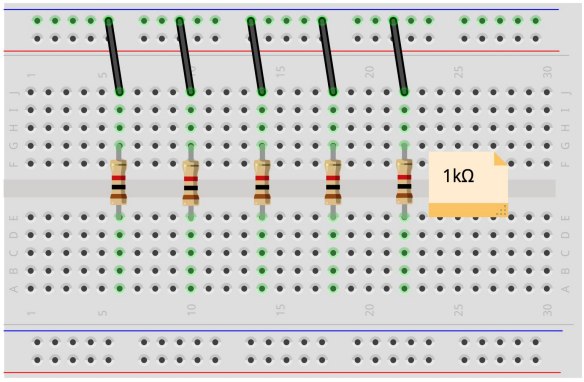


Figure 2.2

IMPORTANT: The next steps are crucial to the performance of our final system -- make sure to read carefully, and assemble/align the photoresistors as good as you can

3. In order for the photoresistors to come out straight, we need to cut short one of the legs for each photoresistor. Follow **Fig 2.3** to align both legs to "a" column of the breadboard (blue line), and cut one of the legs at around the + power rail (red line). Your result should resemble **Fig 2.4** on the next page.

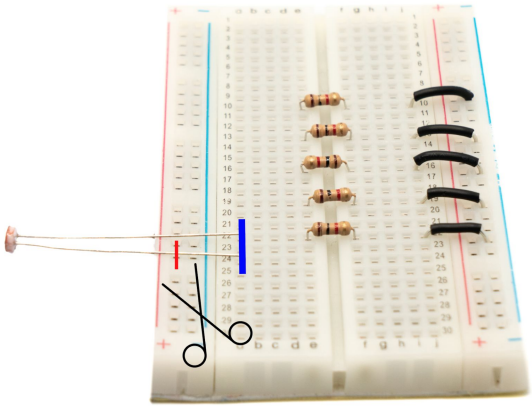


Figure 2.3

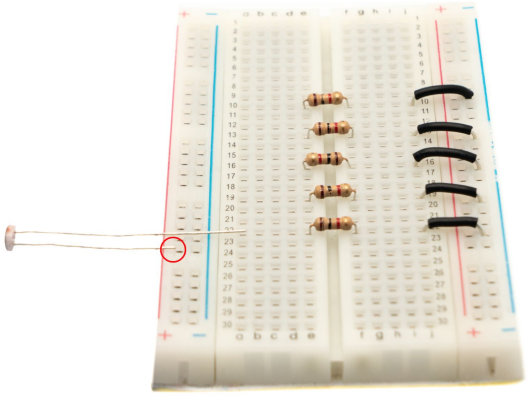


Figure 2.4

4. Bend both legs at equal amounts. You should be able to plug them into the breadboard and fit perfectly. (**Fig 2.5**)

5. Repeat above two steps for the other 4 photoresistors, and follow **Fig 2.6** to plug them in. Space all photoresistors as evenly as possible. Your outcome should resemble **Fig 2.7** on the bottom.

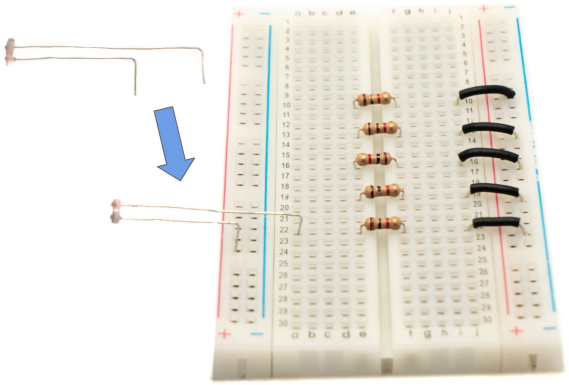


Figure 2.5

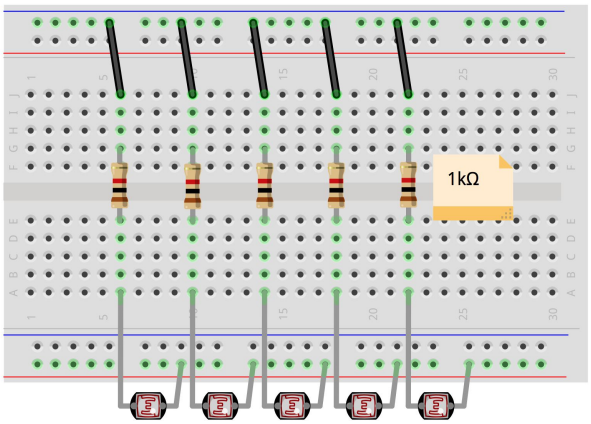


Figure 2.6

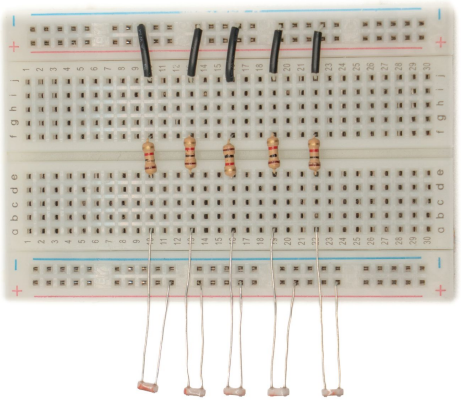


Figure 2.7

6. Glue or tape across the 5 photoresistors to fix their position and alignments. See **Figure 2.8** on the right

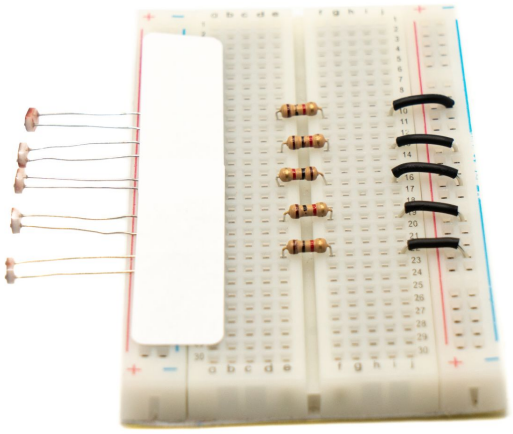


Figure 2.8

7. Find the long plastic servo arm attachment. (**Fig 2.9**) Detach the servo arm attachment if it is connected to the servo. Stick the servo arm attachment to the center of the breadboard (on the back) by carefully tape AND/OR peeling off the yellow protective film to use the adhesive on the back of the breadboard. Make sure to not tape over the center hole of the arm that connects onto the servo motor. (See **Fig 2.10, 2.11**)

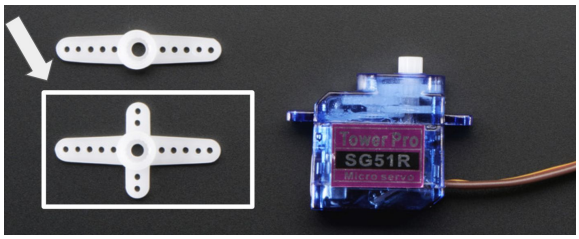


Figure 2.9

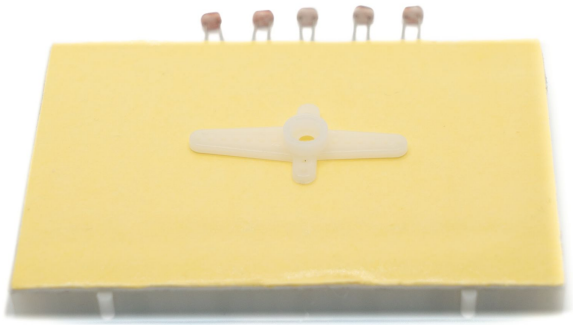


Figure 2.10

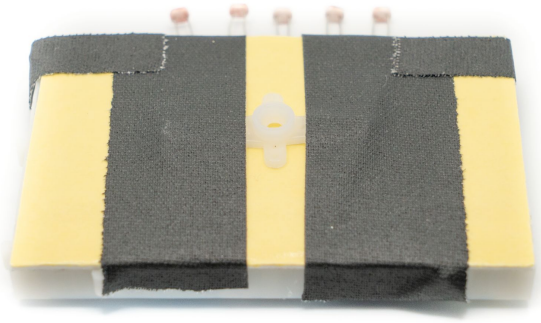


Figure 2.11

8. Upload the code below, and comment out the `servoSweep()` function in the `loop()`.
9. Once you run the code, the servo should be at position 90 degrees (Recall servo has a range of motion between 0 and 180 degrees -- 90 degrees is the center). Attach the servo arm attachment with the breadboard from step 7 onto the servo, with the photoresistors facing directly to front (parallel to servo's wires). (See **Figure 2.12** on the right)

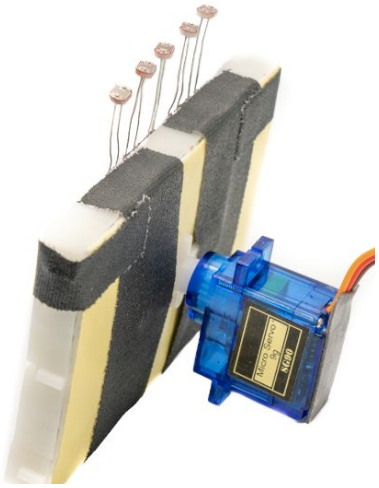


Figure 2.12

10. Complete the following code blanks. Uncomment the `servoSweep()` function in the `loop()` and run the code again to test the servo arm's range of motion. (**Note:** when you test the servo, you may want to tape it to the table to secure it in place while the arm moves).

Code

```
#include <Servo.h>

Servo myServo;  // servo object to control servo
int pos = 0;    // store servo position


void servoSweep() {
  for (pos = 0; pos <= 180; ___ ) { // increase pos by 10 degree intervals
    myServo.write(___); // write current servo position
    delay(100);
  }
  for (pos = 180; pos >= 0; ___ ) { // decrease pos by 10 degree intervals
    myServo.write(___); // write current servo position
    delay(100);
  }
}

void setup() {
  myServo.attach(___); // servo pin number (orange wire)
  myServo.write(___); // servo center position (refer to step 9)
}

void loop() {
  servoSweep(); // comment this out when positioning servo arm
}
```

Challenge #3: Connecting the Photoresistors

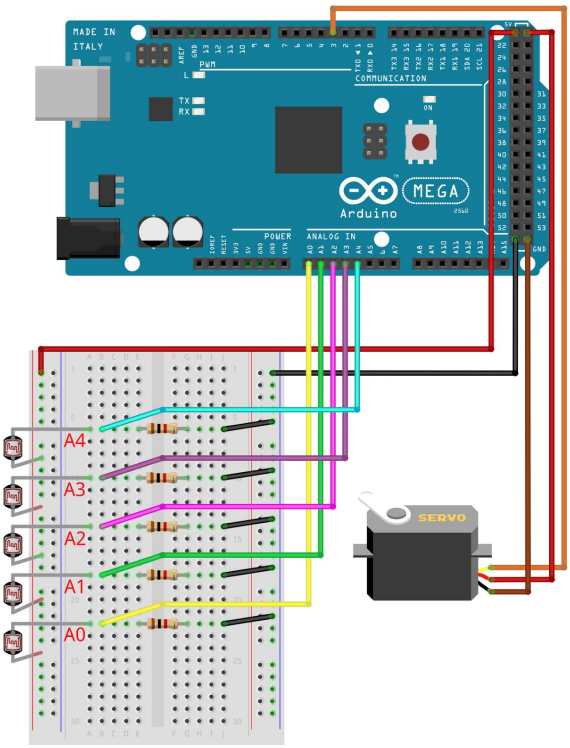
Objective

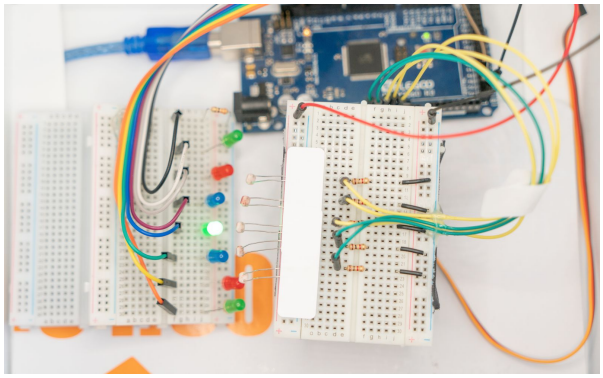
In this challenge, you will connect the photoresistors to the Arduino. Follow the given circuit diagram and use the code to test that all the photoresistors are wired correctly, and are positioned optimally to read different light intensities.

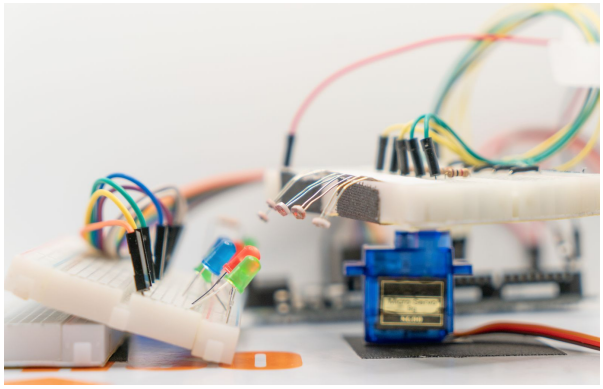
Components

- Setup from Challenge #2
- Wires
- Tape

Circuit Diagram







Task

- Connect the 5 photoresistor outputs and 5V / GND rails to Arduino following the circuit diagrams above.
- Bend your row of photoresistors downwards so they point directly to the row of LEDs. (We will fine turn this in the next challenge)
- Complete and use the code below to test that each photoresistor reads correctly.
    - Use your finger to block them one by one from left to right (with photoresistors facing forward), and see light intensities printing on Serial changes, following the order of left to right.

Code

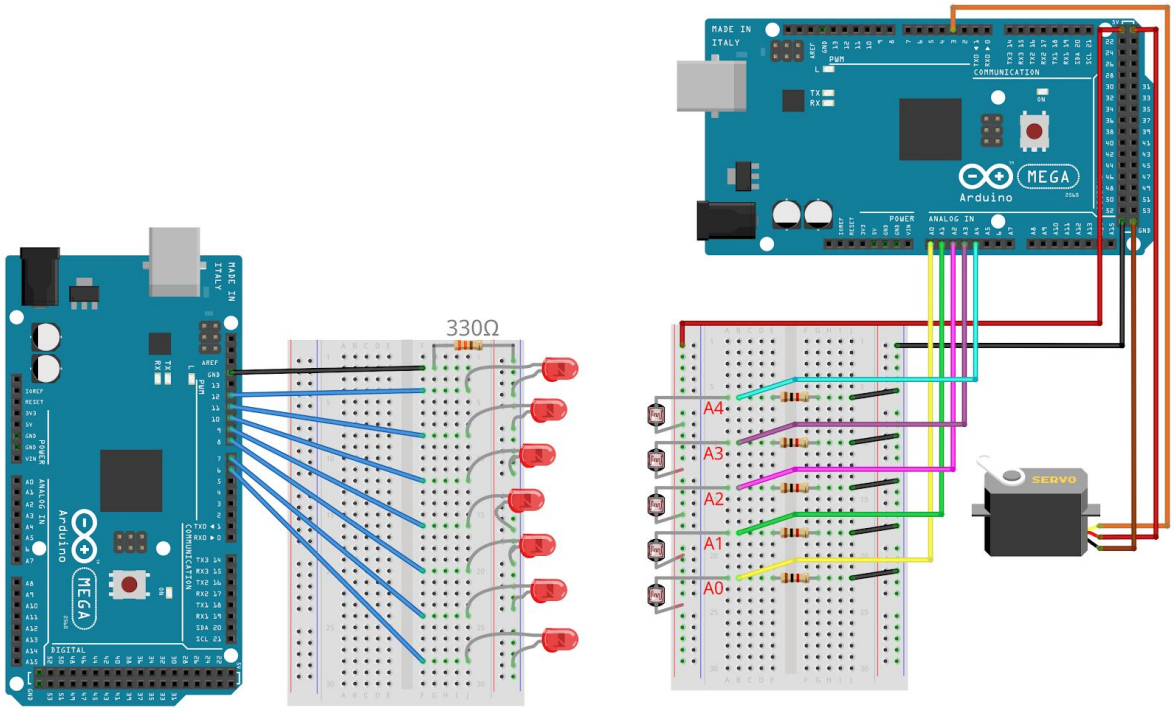
```
#define NUM_OF_PHOTORESISTORS 5
int photoresistorPin[] = {A0, A1, A2, A3, A4};


void readLight() {
  for(int i = 0; _____ ; i++) { // complete For loop
    Serial.print(analogRead(_____)); // print each photoresistor value
    Serial.print(" ");
  }
}

void setup() {
  Serial.begin(9600);
}

void loop() {
  readLight();
  Serial.println();
  delay(200); // slow down output
}
```

Challenge #4: Following the Light

Objective

In this challenge, you will be combining all the previous challenges to create the light following arm! We will optimize our positioning of sensors and LEDs to achieve the best result.

Circuit Diagram

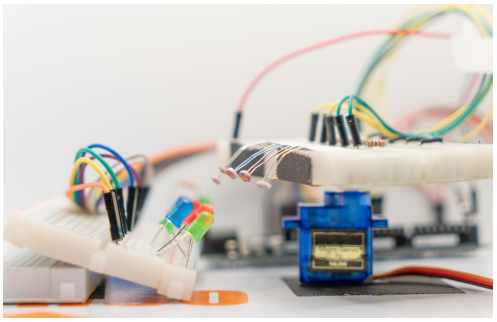


Task

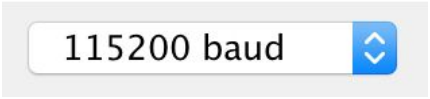
- Upload ***Lab4Ch4.ino*** (available on Canvas). Follow the instructions below to optimize your row of LEDs / photoresistors positioning.
- Unzip ***Lab4Final.zip*** and upload ***Lab4Final.ino*** (available on Canvas). Use ***Controller.mlapp*** to run the program. Perform final debugging if something is not working.

Instructions

1. Try your best to bend the photoresistors so that at all servo angles, the LEDs could point to them directly. This will ensure a maximum difference of light we read across the photoresistors.



2. Open Serial monitor and change the baud rate on the bottom right to 115200 to match with the code.
(This allows a faster data transmission)



3. Open Serial Monitor after uploading the Arduino code *Lab4Ch4.ino (download from Canvas)*. After calibration, observe the max and min readings by the photoresistor. Your positioning in the last challenge should allow a significant difference between those two numbers.

```
Calibration started!
Calibration Finished!
----------Max Readings----
520 489 374 491 485
----------Min Readings----
95 115 71 88 67
-------------------------
```

4. If you performed any adjustments to your positioning, hit the red reset button on Arduino so it recalibrates. Once you have achieved a good difference on those calibration readings, look at the mapped photoresistor values below. You should see the photoresistor facing the LED currently turning close to 100. On the right most column, the error calculated should be going gradually up and down indicating the angle difference between the LED and the middle of our robot arm.

```
96 57 24 17 7 1.63
95 58 24 18 7 1.62
93 63 26 18 6 1.60
87 70 30 20 8 1.55
88 68 28 20 8 1.56
82 73 32 21 8 1.53
80 75 32 21 9 1.52
76 78 34 22 8 1.22
70 84 37 25 9 1.17
66 89 41 26 11 1.13
63 91 42 27 10 1.11
60 93 45 28 11 1.08
```

5. Once you have reached a good point for the hardware set up, upload *Lab4Final.ino*, and use the MATLAB GUI to connect to Arduino and run the final light following code.

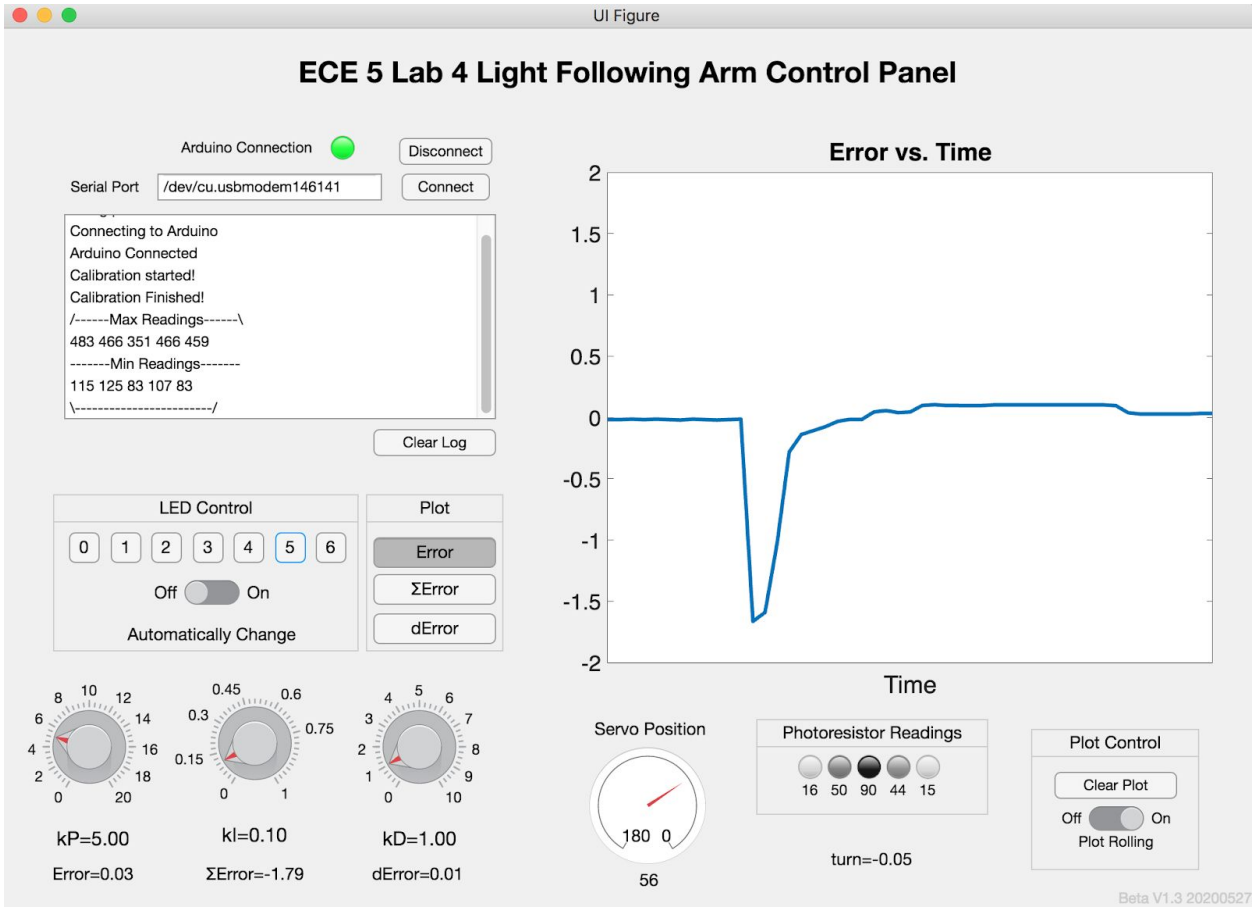
Code

Access from Canvas. Nothing needed to modify.

Challenge #5: Graphing Error with MATLAB GUI

Objective

In this challenge, you will be connecting your Arduino to MATLAB. You will send data from the Arduino to MATLAB and graph the error with MATLAB GUI.



Task

- With the help of MATLAB GUI, connect to Arduino, and fine-turn the PID parameters by adjusting the three knobs on bottom left