Programming 4 Homework - Analog Signals

For programming homework, we will begin the activities in class and use them to go over and reinforce the pre-lab content.

WARNING:

When making adjustments to your circuit, it's best practice to disconnect it from power and double check your wiring before re-connecting it. The Redboard is fairly robust but this will minimize the risk of causing it any damage.

If you are struggling with coding already, when using the SparkFun example codes. I recommend retyping at least the part that confuses you, line by line. This will force you to examine it incrementally and take the time to improve your understanding of what each line of the code does. You can skip retyping the commented portions although your final submission will need to be fully commented.

You should currently be able to

- Access the Arduino IDE
- Flash a Program to your RedBoard
- Save your work and recover your work
- Properly set up and submit documents for SparkFun assignments
- Use the Arduino IDE
- Break down a basic Arduino Program
- Utilize Digital Signals
- Use Basic Hardware
    - LEDs
    - Buttons

This assignment will focus on:

- Utilize Analog Signals
- Use Analog signals with Basic Hardware
  - LEDs
- Use new hardware:
  - Potentiometer
  - Photocell

The required deliverable of this assignment, parts 1, are broken up in a few steps. Keep these in mind while working on them. They involve:

- **Build an example**
  - Build a simple example
  - DO NOT SUBMIT THIS AS THE ASSIGNMENT DELIVERABLE
- **Understand the example**
  - Break down the example to understand how to use the concepts yourself
- **Modify the example**
  - Use that new understanding to meet the assignment requirements

YOU MAY WORK IN PAIRS. You are still required to submit your own copies of the code and video. You may both use the same video but it must be submitted by both. You must both submit your own TopHat answers.

Review

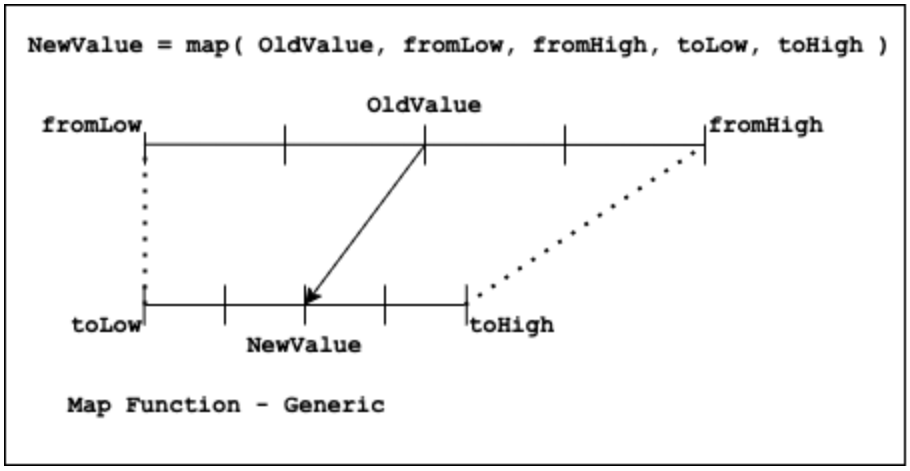
Map(...)

In Lesson 1, we introduced the `map(...)` command. This caused some confusion so hopefully the following helps clarify:

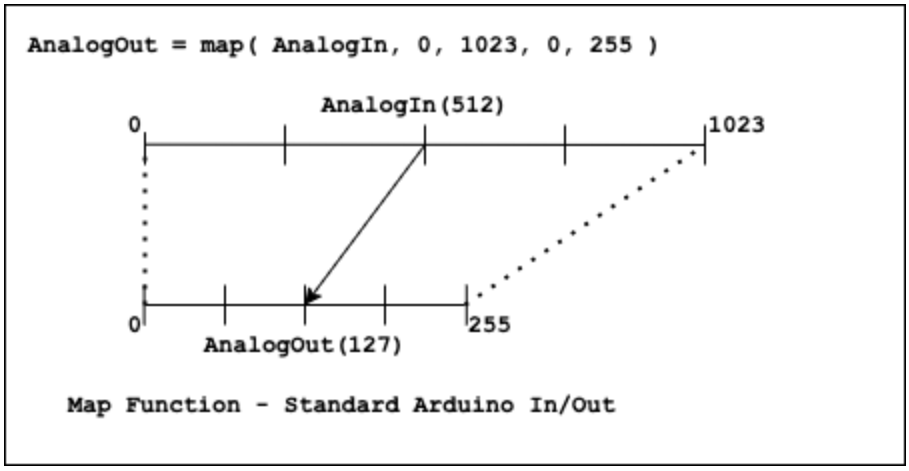
- `NewValue = map(OldValue, fromLow, fromHigh, toLow, toHigh);`
  - Re-maps a number from one range to the corresponding value on another range.
  - `OldValue`
    - The number on the current range to be mapped to the target range
  - `fromLow`
    - The lower bound of the value's current range
  - `fromHigh`
    - The upper bound of the value's current range

- `toLow`
  - The lower bound of the value's target range
- `toHigh`
  - The upper bound of the value's target range
- `NewValue`
  - The number on the target range mapper from the current range

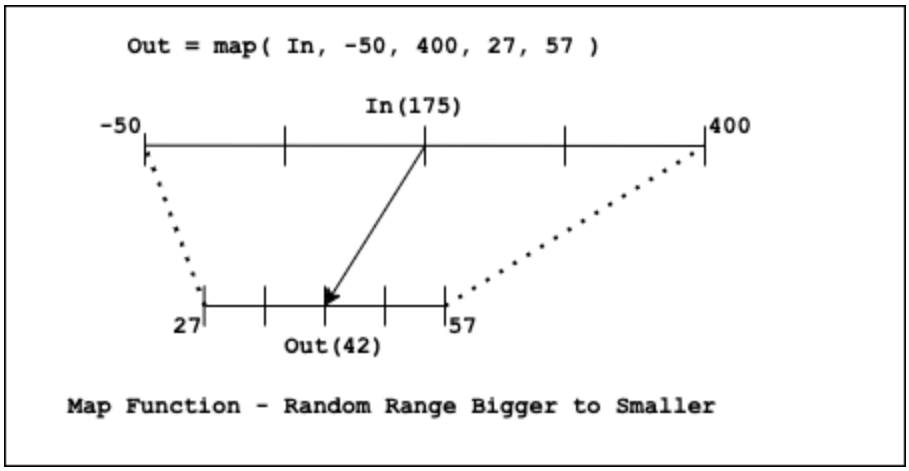
The following graphics may help to visualize this:



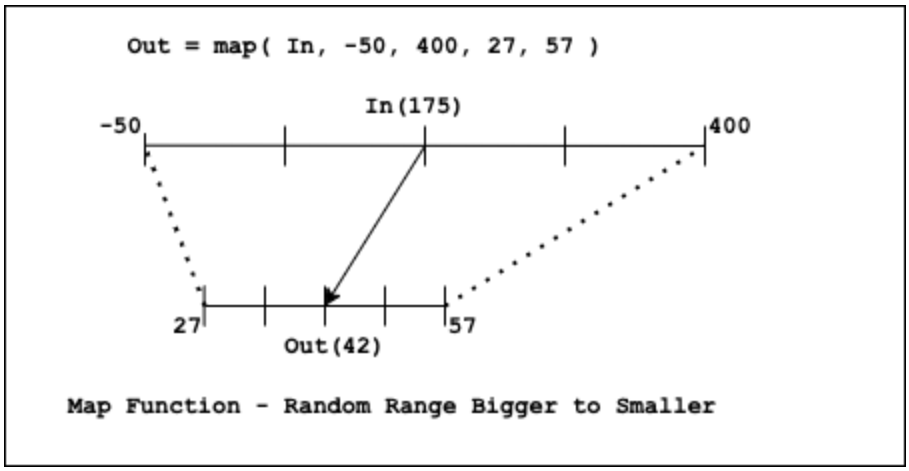
Generic representation of the Map Function



Standard Arduino Analog In to Analog Out Mapping

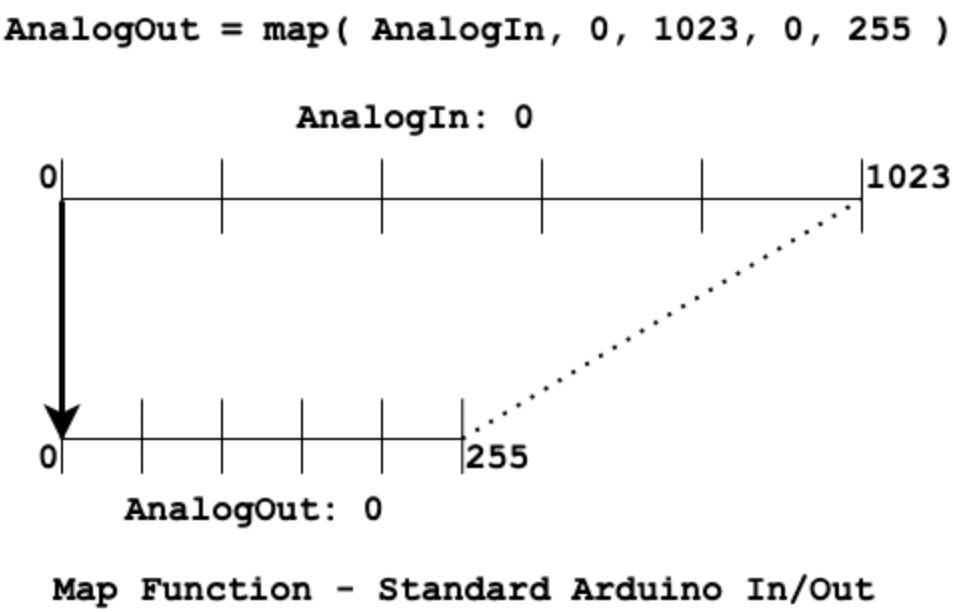


Mapping from arbitrary values - Larger to Smaller range



Mapping from arbitrary values - Smaller to Larger range

This GIF (pronounced with a hard G as it should be) shows this mapping across the standard ranges:



IFTT

IFTT in Arduino works conceptually the same as it does in every other programming language. The difference is in the syntax. For Arduino, it is:

```
if (conditional 1) {
 // Action A
}
else if (conditional 2) {
 // Action B
}
else {
 // Action C
}
```

Things to note are the `(...)` around the conditionals and the `{...}` around the actions. Those are necessary in C++ and it's variants, like Arduino. Missing them or including extraneous semi-colons instead will cause an error.

Boolean Operators

If you want to combine conditionals, you'll need to use the following boolean operators:

- `!` (logical not)
- `&&` (logical and)
- `||` (logical or)

Single operators are allowed in other languages but Arduino requires the doubles. That's typically the error involved with these, you only put `&` and not `&&`.

New

For today, we will be using an analog sensor with a limited range.

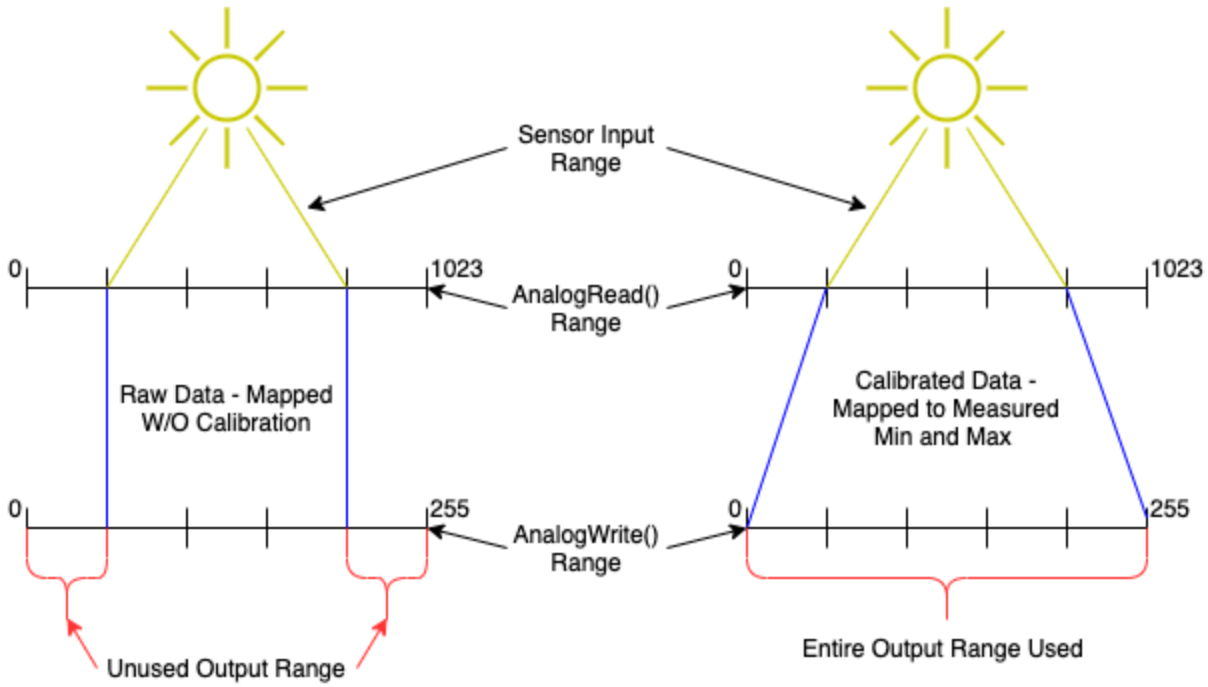
Photoresistor

Photoresistors, or photocells, are light-sensitive, variable resistors. The resistance

across the terminals decreases as the amount of light across the sensor's face increases. They're a common component typically used for measuring or reacting to changes in ambient-light.
The signal from a photoresistor requires absolute darkness or a direct extremely bright light to provide either a 0 or 1023 so it's range is typically limited by not being used in those extremes circumstances.

Calibration

This is where calibration comes in. Since the photocell will never produce a 0 or 1023 signal, it's important to know what it's actual min and maximum signal will be. The activity will go over how we do that. Knowing that information though will allow us to utilize the full range of the sensors capabilities and adjust our outputs accordingly. Otherwise, if you're doing a direct mapping of the input range to the output, then you reduce the overall capability of your design by not fully utilizing them.



This is simply how calibration works for a photocell and other similar variable input sensors. Other more complex sensors would have more specific calibration methods for them.

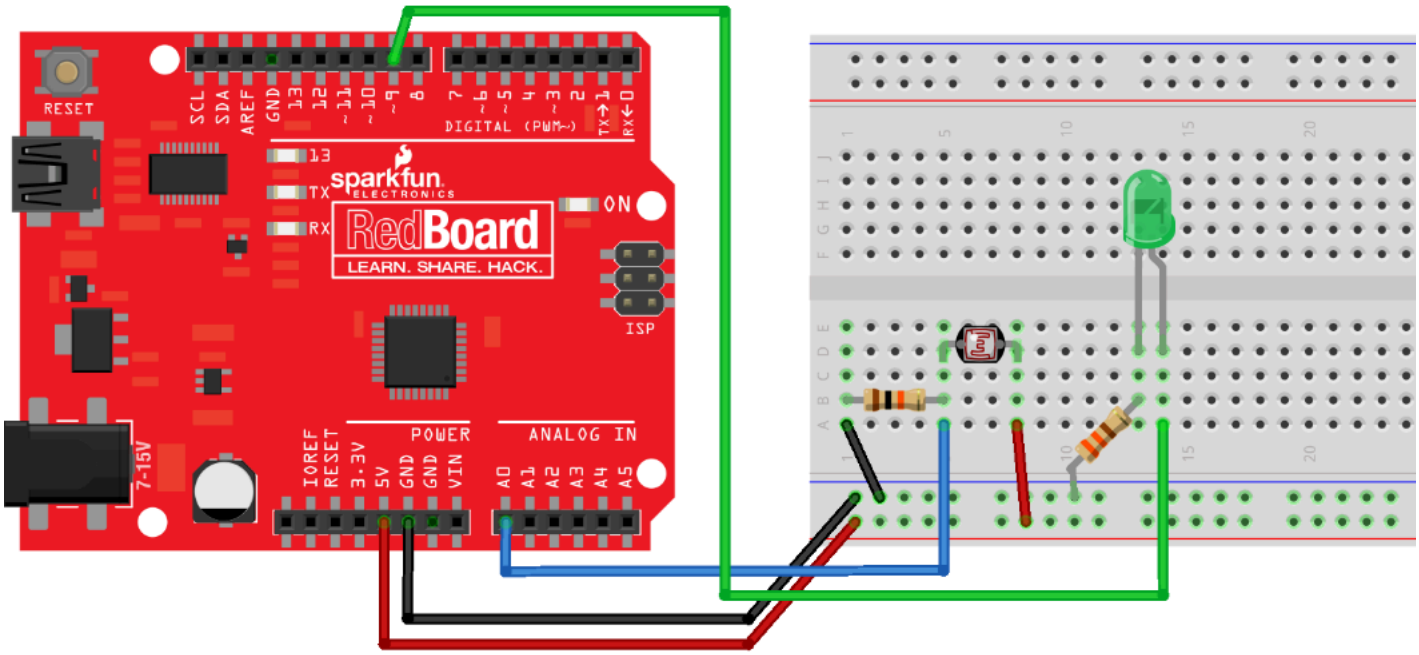
Part 1: Signal Calibration

Build an Example

We will be building the built-in Arduino example **Analog In Out Serial**

Build the Circuit

Build the following circuit. This time we're using a Photocell:



Upload the code

- Open the built-in **Analog In Out Serial** example
  - Go to **File / Examples / 03.Analog / AnalogInOutSerial**
- Save a copy as this part of the assignment
  - Include your title block
- Upload that code to the board

Experiment

- Open the Serial Monitor
- Try putting your finger over the Photocell
  - What level is it reading?
  - **Write it down.**
- Expose the photocell to the ambient light of the room
  - What level is it reading?
  - **Write it down.**
- Write down the minimum and maximum value you saw

    - You may have to increase the delay to be able to read these
- Notice the brightness change of the LED.
    - It should go completely off or go fully on

You can get more extreme values covering it fully in your hands or shining your phone light on it. The values we want for later though are the covering it with a finger and the ambient room light.

Understand the Example

We've already done this during Lesson 1.

Modify the Example: Signal Calibration

What we want to do is automatically determine the minimum and maximum input capabilities of the photocell and use those in mapping our output instead of 0 to 1023.

Modify the Circuit

We won't be modifying the circuit for this one.

Modify the Code

Pseudocode for this assignment:

1. `PROGRAM Signal Calibration`
2. `In the SETUP:`
3. `Print to serial monitor calibration starting;`
4. `WHILE time < 5 sec`
5. `    Sense the sensor value (photoresistor);`
6. `    Update the minimum and maximum values as required;`
7. `ENDFOR`
8. `Print to serial monitor calibration finished;`
9. `ENDSETUP`
10. `In the LOOP:`
11. `Sense the sensor value (photoresistor);`
12. `Set LED output level relative to sensor value using min and max sensor value from calibration and full output range;`

13. Constrain the output to within allowable range;
14. Turn LED to the output level;
15. Print to serial monitor: Minimum and maximum levels, signal level & Output level;
16. ENDLOOP
17. END

Implementation

There are really only 2 new elements to this:

1. Adding a calibration phase to the setup()
2. Constraining the output to be within the allowable range

Element 1: Calibration

1. WHILE time < 5 sec
2. Sense the sensor value (photoresistor);
3. Update the minimum and maximum values as required;
4. ENDFOR

Update the minimum and maximum values as required;

This is a fairly common task and can be achieved by a simple algorithm.

1. Establish variable for min set very high, to the top of the expected range
2. Establish variable for max set very low, to the bottom of the expected range
3. In Setup:
4. WHILE time < 5 sec
5. Read SensorValue(SV)
6. IF SV < min
7. min = SV;
8. ELSE IF SV >max
9. max = SV;
10. ENDIF

11. `ENDWHILE`

This works by guaranteeing that the first reading will be less than the min then resetting that stored min to the new reading. This is because you set your min to something very high, like 1023, and since that's the top of the range, any reading will be below it. The next time around you are likely to have a reading that is greater than the max then resetting that stored max to the new reading. This is because you set your max to something very low, like 0, and since that's the bottom of the range, any reading will be above it. This will continue for 5 seconds as you adjust the exposure to the photoresistor by covering and uncovering it. By the end, you'll have a min and max set for the range of actual sensor values you'll see, not just the full range of possible sensor values.

```
WHILE time < 5 sec
```

For creating a while loop, you can use the Arduino resources to get the syntax. While there, you can also look up the command millis(). This command returns the number of milliseconds passed since the Arduino board began running the current program. Essentially, it's our timer. It returns in milliseconds so you will need to convert. So 5 seconds is 5000 milliseconds.

Once you have the minimum and maximum sensor value, you can update the mapping to be from that range rather than 0-1023.

Element 2: Constraints

Even though you've established the minimum and maximum, you don't want to have a situation where a sudden extreme occurs that exceeds that. That would set the output value to outside of the allowable range. Normally this would be ok, analogWrite would just default to 0 or 255 but there are some funny cases with negative values that will cause errors. To avoid that, you would want to '`Constrain` `the output to within allowable range`.' Arduino provides a limiting function for that case. Your challenge is to identify and implement it from the Arduino Resources. You want to add a command that **constrains** a number to be within a range, specifically 0 to 255 in this case. Go to the Arduino reference page to see the available Arduino functions and see which would achieve that goal.

End of P4H1:

Make sure that you use save as... to create a copy of your modified code. Using

- The Raw Code – The .ino file

- The PDF of your Code – The .pdf file
- The video of your working circuit – The .mp3 file
    - There will be chatter around the room. Do your best to speak audibly for your video. It will end up picking the noise of the room. We can not help that for the in-class lessons.

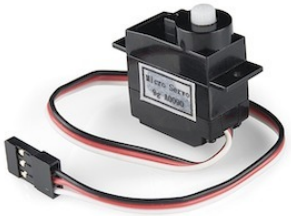
This is part 1 of 2. You must submit both parts of the assignment.

Part 2: "Figure It Out"

The name of this part of the assignment may sound flippant, but I assure you it's not and you're not being left to just do so from scratch. Very often with a new project, as part of a new course, or as your design requires it, both in academics and in industry; you'll be asked to explore new technologies and, more often than not, completely from scratch. This is less about figuring out a servo but with identifying how you go about learning how to use a new a bit of technology and establishing your own processes for going about this.

YOU WILL BE TOLD SOMEDAY ***"FIGURE IT OUT"*** BY AN AUTHORITY FIGURE IN A CLASS OR A JOB AND THAT'S ALL THE GUIDANCE THEY'LL GIVE YOU. That may end up being a sink or swim scenario, so I want you to have the skills to swim. Take this opportunity to start learning how you'd best handle that; what resources tend to be the most useful to you. There are some starting points below to handle this specific circumstance, *but I also advise* doing a little searching on your own to check you're own ability to find resources.

Here are are readily available resources to figure out how to control a servo that might help:



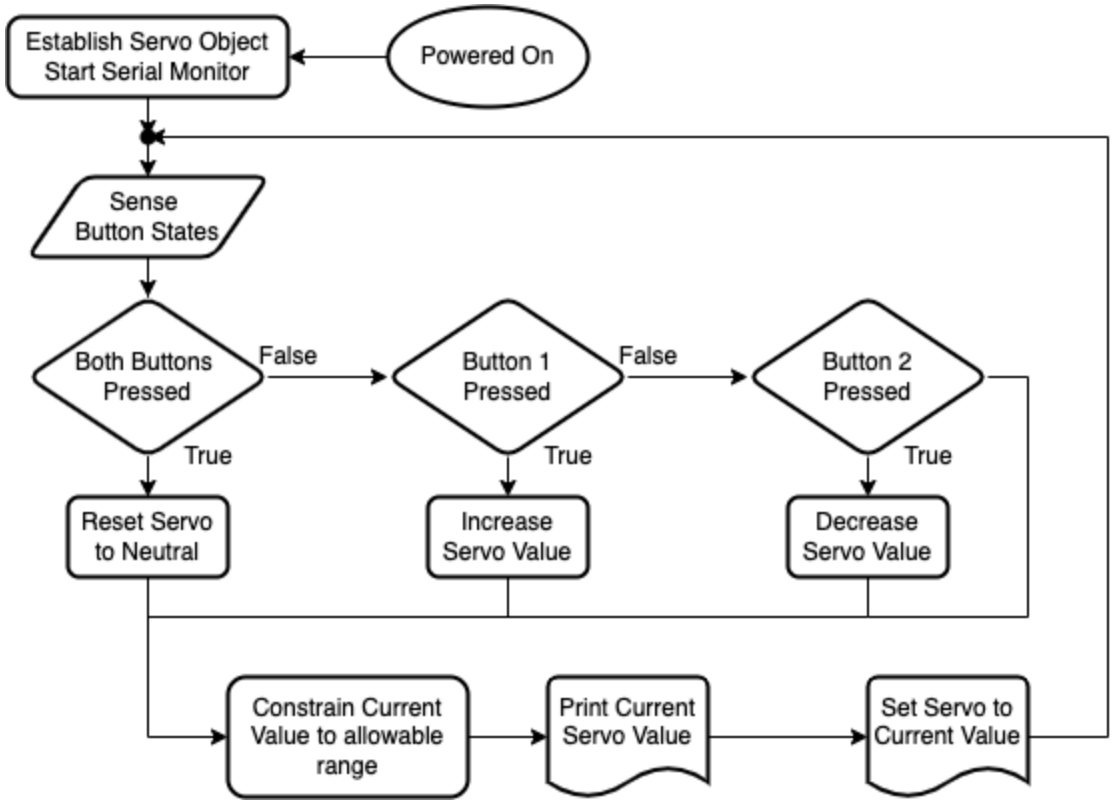
- Your Sparkfun Inventor's Kit Guide
- https://learn.sparkfun.com/tutorials/photocell-hookup-guide
- https://www.arduino.cc/en/Tutorial/Calibration
- https://learn.sparkfun.com/tutorials/hobby-servo-tutorial
- https://www.arduino.cc/en/Tutorial/Knob

NOTE: I will say that some tutorials will say you need a small capacitor. FYELIC has them but it shouldn't be necessary for the small servos in your kit. The capacitor is mainly to protect from

backlash when over-torquing the servo. You're not connecting anything to your servo that would mechanically lock it up so no need for a protective capacitor.

Requirements:

- Use a variety of sources to learn how to use the servo from your sparkfun inventor's kit
- Create a script or modify an existing example to create a project that behaves as indicated in the following flowchart:



- Include in your title block the citations for the resources used to understand this component and note how you found them useful (or not) as well as which type you believe you'll most likely use the next time you have to figure out a new component or technology

No Pseudocode or Flowchart for this part since you're given one (that your implementation must match). I just want you to be able to figure out a new component using readily available open-source resources outside of this course.

Assignment Summary

Part 1: Signal Calibration

Modify the Arduino example code **AnalogInOutSerial** to include calibration for a photocell's limitations.

Part 2: "Figure it out"

Utilize publicly available resources to learn about servos and implement the provided flowchart

Submission Requirements:

P3H1: Signal Calibration

- Raw code
  - Includes Title Block
  - Includes comments
  - Based on the provided pseudocode
- PDF
  - An image of your circuit
  - A copy of your raw code - Ensure formatting is preserved
- Video
  - Shows face or Student ID
  - Shows upload
  - Describes the running circuit
  - Upload raw video but adding to the assignment as a media comment is preferred

P3H2: "Figure It Out"

- Raw code
  - Includes Title Block
  - Includes comments
    - clarify what does what, tying the code back to the given flowchart
  - Ensure your title block includes
    - References to any sources used in learning your code (Use IEEE citations)
    - Description of their usefulness (or lack thereof)
    - A statement about what resources you will seek out first next time you have to figure out a new component or technology
- PDF
  - An image of your circuit
  - A copy of your raw code - Ensure formatting is preserved
- Video

- Shows face or Student ID
- Shows upload
- Describes the running circuit
- Upload raw video but adding to the assignment as a media comment is preferred

The following survey is for my reference to help improve future assignments. The results are not checked until after the semester is completed and participation is not required.