

System Report

1.0 Scope of Work

1.1 Behavior Description

The goal of homework/project 4 is to incorporate analog and digital devices into the AVR Butterfly's interface. The user can press the up or down buttons of the integrated pushbutton to select multiple modes for the device to take measurements with. The Up Button triggers *Follow the Light* mode, which sends a PWM signal from the Butterfly to the servo attached to PORTB4. The Light Dependent Resistor is then used by ADC1, which takes measurements at every 20-degree interval up to 180 degrees inclusive and stores them in an array. These values are compared, and the largest value is the brightest location of the 180-degree sweep and displayed to the user. From there, the servo rotates in order to take measurements at the primary angle, the (angle - 10 degrees), and the (angle + 10) degrees. The same method is used to store and compare these values. The largest of these three values is then the brightest location, and that angle is displayed to the user on the LCD. These two behaviors are considered *FULLSWEEP* and *LOCALSWEEP* respectively and use their own functions in the C file. The respective mode to *Follow the Light* mode is *Avoid the Light* mode, which follows all of the same steps with the only difference being that the comparisons are to find the smallest value for each of the sweeps. The *LOCALSWEEP* function is designed to avoid the edge cases of the brightest or dimmest position being either 180 or 0 degrees. An *if else* block is used for comparing the primary angle and only performing a local sweep on the two angles that don't exceed the bounds. As stated in the document, interrupts are disabled when in *FULLSWEEP* mode, so no button presses can stop the measurements being taken. As it was not stated to do this during *LOCALSWEEP*, I kept interrupts enabled during this function.

1.2 Testing and Debugging

Testing has been done to find the proper timing for the servo so that it moves in both 20- and 10-degree increments, this took a majority of my debugging time. I used the provided ADC skeleton file to ensure my LDR was working properly and have pointing the LDR at and away from different light sources during the sweeps to test for accuracy. As these resistors are low accuracy and only work for big shifts in light movement, my testing seems relatively conclusive and I feel confident that my arrays are storing and comparing the data properly. A demo video has been included to show the final implementation of the LDR and Servo. Because I was not able to make my servo go clockwise using PWM, my LDR is not attached to the servo itself, as the wires would very quickly quick twisted.

1.3 State of Completeness

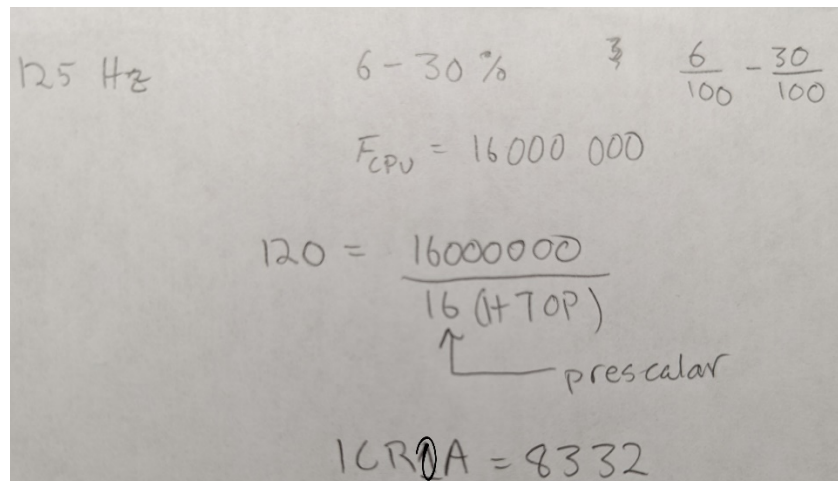
The overall design of this project is functional. The only content inside of the *while(1)* loop in *main()* is the commands to print the text "PROJ 4" on the LCD panel. All other behavior is controlled through the interrupts from PINB6 and PINB7. As a way to

complete my full and local sweep behaviors with only being able to rotate one direction, I have the servo complete another 180 degrees turn while not collecting any data. This sets the servo position to its original location. Once it is in its original location, the localsweep function rotates the servo to the position 10 degrees less than the brightest/dullest angle. It then rotates in two 10-degree intervals (edge cases withstanding). Once the ADC data from the three positions has been collected, the servo rotates the rest of the way around the 180 degrees. While it is not preferable to have a program that drives a servo omnidirectionally, this was the best solution I could find after iterating through almost all ranges of OCR0A values and *for loop* parameters.

2.0 Project Overview

2.1 Design Decisions

I chose to complete this project in small chunks by writing and testing modules for each part of the system separately. I started by only including the necessary functions to initiate and drive the ADC so that I knew my diode was working properly and interfacing consistently. I then worked on creating PWM pulses that met the criteria and were able to rotate ~180 degrees in 9 pulses as well as increments of 10 degrees at one pulse each. Using a scalar of 16 for the counter, I used the formula below to find my ICR0A value. Once I was able to precisely move the servo, I integrated the two devices into one project file and started working on the data portion of the project. This involved passing the ADC values to the array at each step of the servo sweep, then using comparisons to find the extreme values for *Avoid* and *Follow*. Once all three were tested and working, I was able to test the full system and make the necessary tweaks to get it to this point for submission.



Handwritten calculations on a piece of paper:

$$125 \text{ Hz} \quad 6-30\% \quad 3 \quad \frac{6}{100} - \frac{30}{100}$$
$$F_{CPU} = 16000000$$
$$120 = \frac{16000000}{16 (\text{+TOP})}$$

↑
prescaler

$$ICR0A = 8332$$