Laboratory Assignment 2

September 22, 2015

Austin Huck

Taylor Okel

# Objectives and Problem Description

Take analog inputs, interpret them, and control a servo motor based on the inputs. Two problems were investigated in this lab: Using a potentiometer as the position sensor of a steering mechanism, use a servo motor to set the rudder position indicated by the input. Using a potentiometer as a windshield wiper variable speed selector, use a servo motor to act as the wiper and respond to speed inputs.

# Procedure

To begin the RC times for the potentiometer used through all parts of this lab were determined. Using the circuit in figure 1 and the RCTIME function, these were found to be 1us at the lowest setting and 543us at the highest setting. This servo circuit in figure 2 was added to the existing potentiometer circuit and would serve as the full circuit for the remaining two parts. The summary of components and connected microcontroller pins can be found in tables 1 and 2.

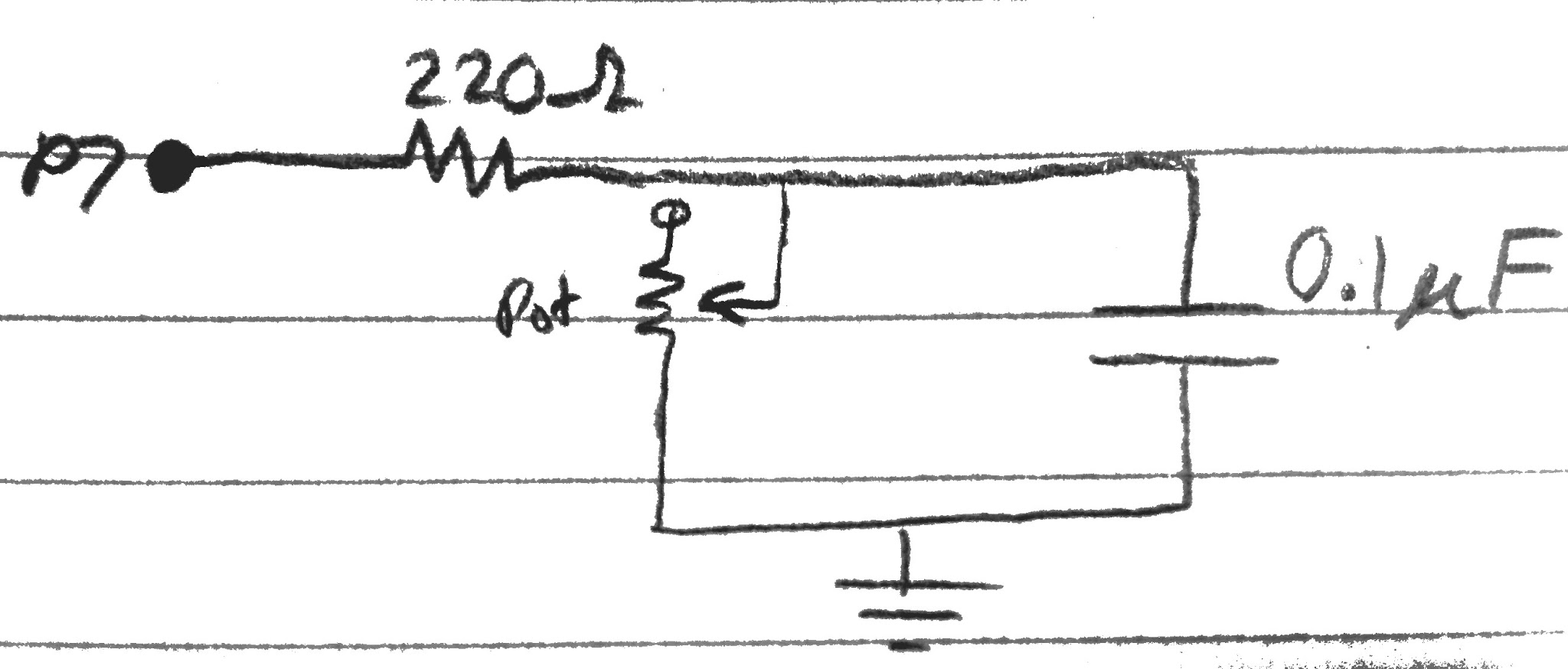


Figure 1: Potentiometer Circuit

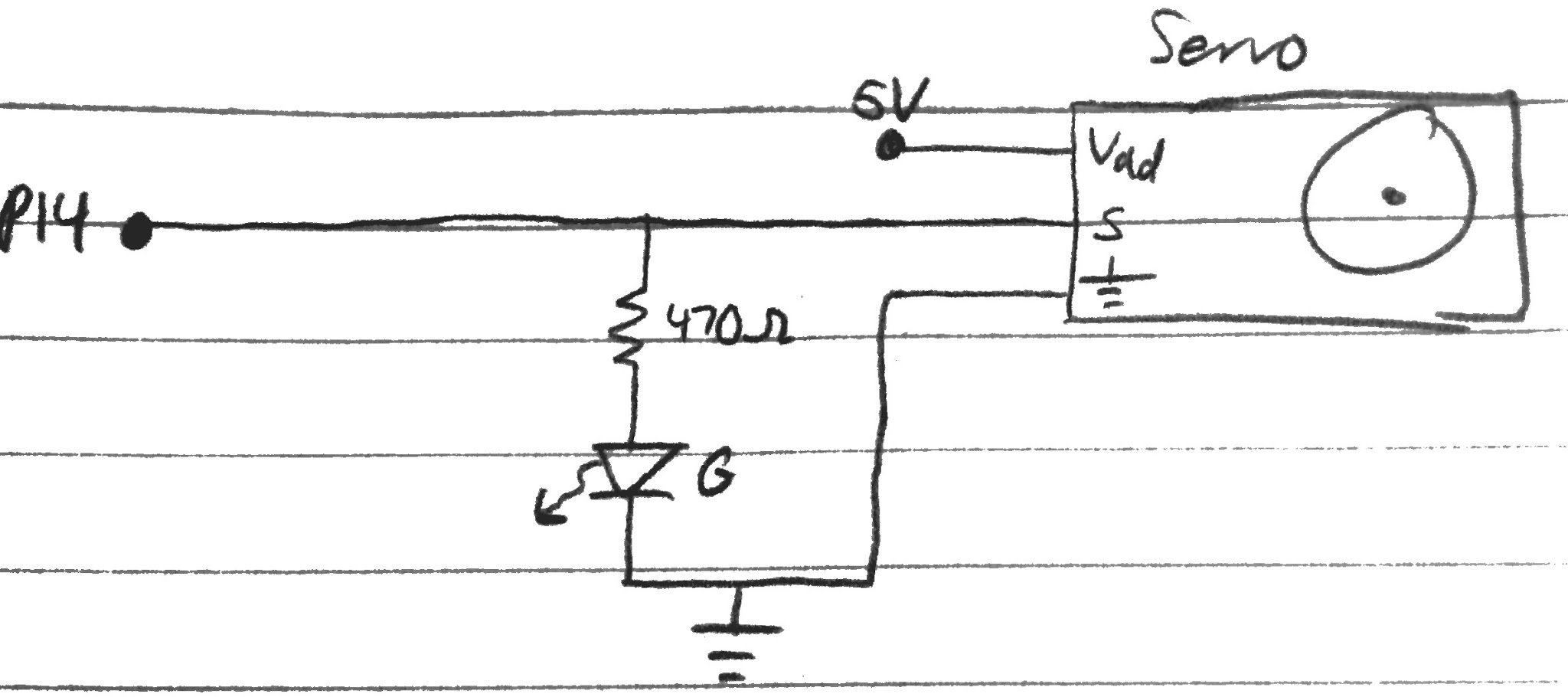


Figure 2: Servo Circuit

|  |  |  |  |
| --- | --- | --- | --- |
| Components | | | |
| Resistor - 220Ω (1) | Potentiometer 10KΩ (1) | Capacitor - 0.1uF (1) | Servo Motor (1) |

Table 1: Full Circuit Components

|  |  |
| --- | --- |
| Pins | |
| 7 (in) Potentiometer | 14 (out) Servo |

Table 2: Full Circuit Pins

Following the design of the potentiometer and servo circuits, a PBASIC program was designed for the rudder controller. The program flow, as depicted in figure 3, was three steps. First, read the input potentiometer value using RCTIME. Second, map the input value (0 - 543) to desired output value (500 - 1000 for PULSOUT). Third, send a high pulse to the servo using PULSOUT with the mapped value as a parameter and repeat.

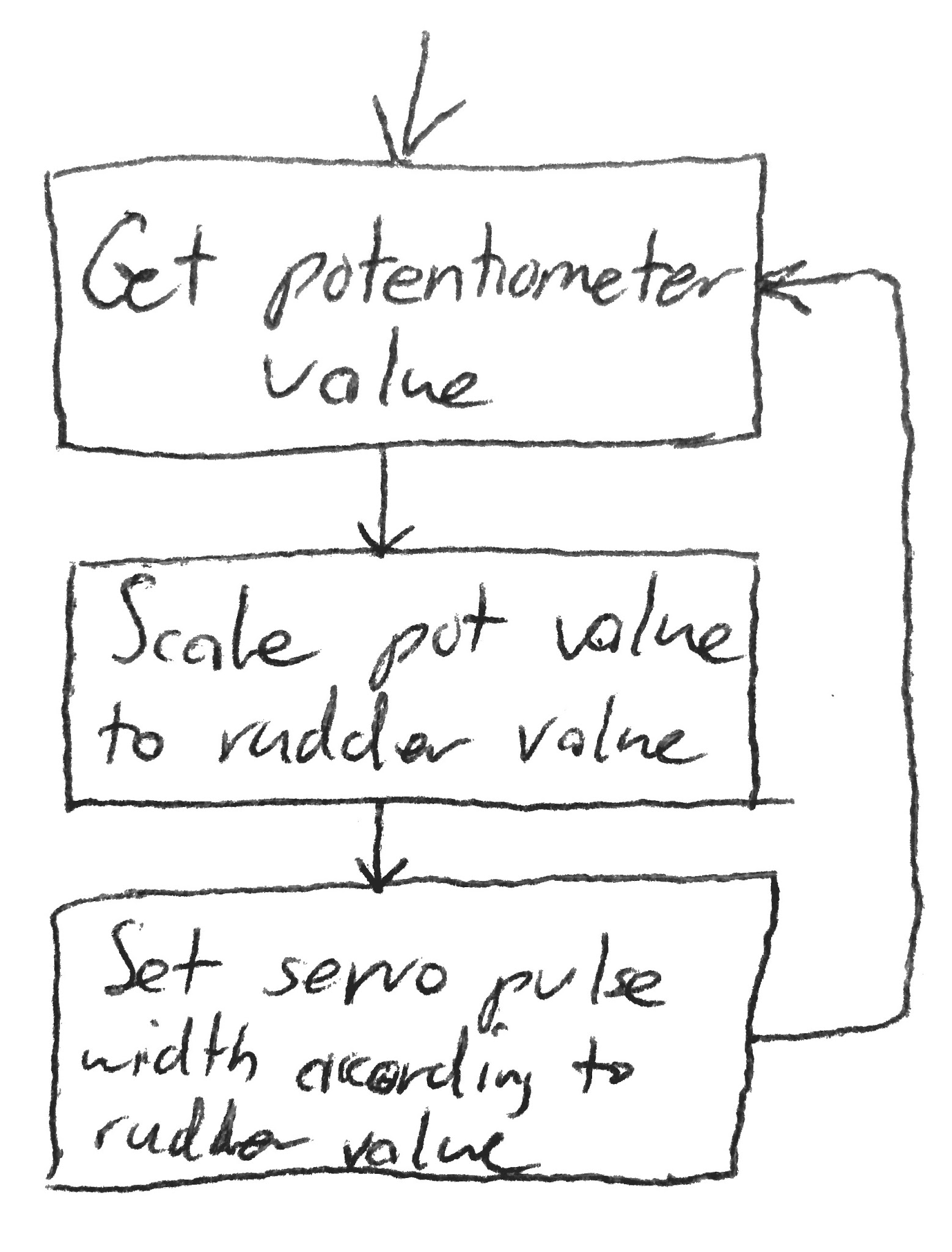


Figure 3: Rudder Controller Flow

For the Windshield Wiper activity, the same circuit will be utilized; the only changes to the design will be made in the code. Figure four depicts the new logic for the design.

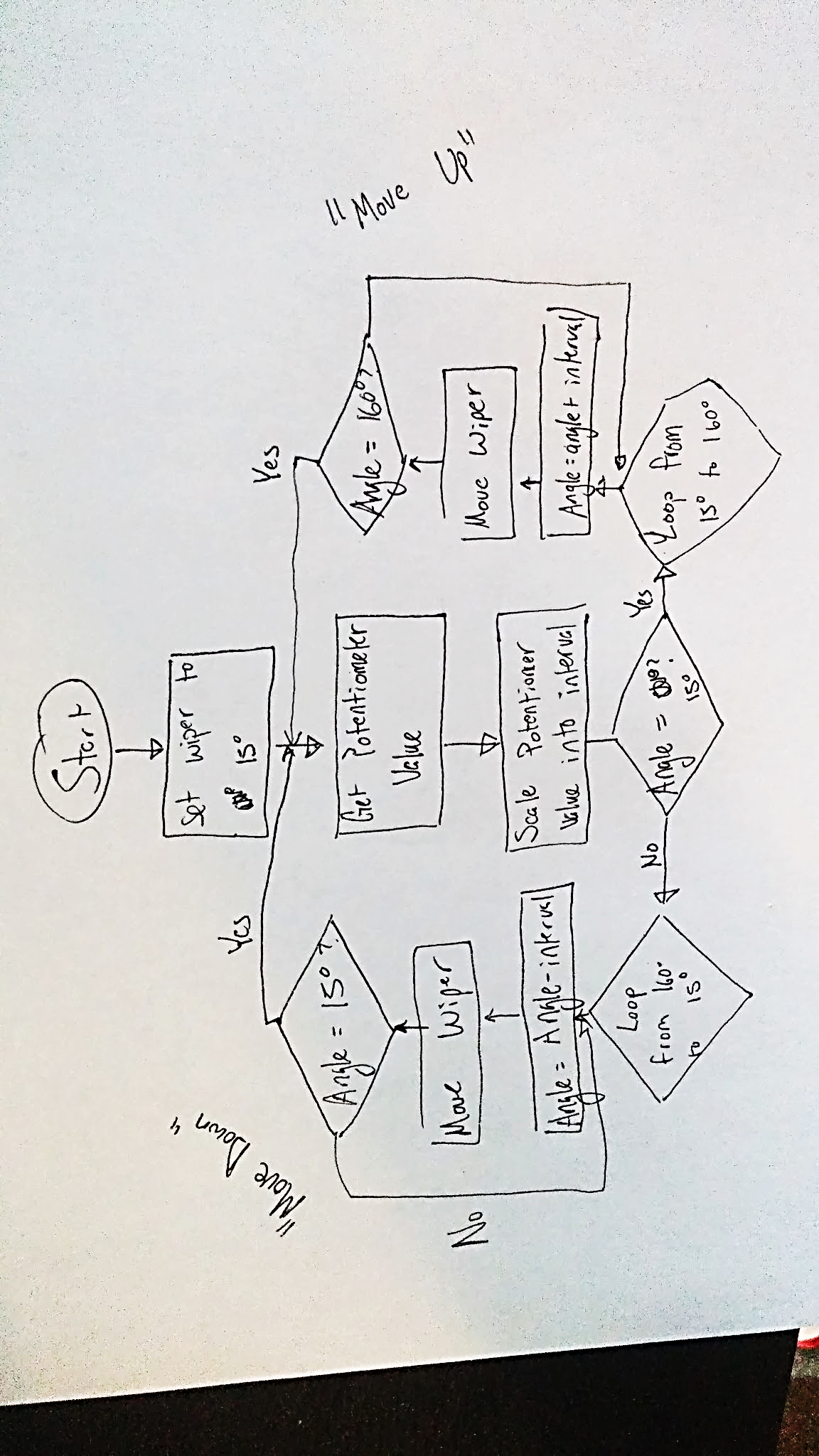


Figure 4: Wiper Controller Flow

This portion of the lab requires a translation between a potentiometer value to angular velocity of the wiper. We found the scale by plotting the potentiometer range versus the desired interval, which the interval was found via arbitration and guess-and-check techniques to find the cleanest interval scale.

Figure 5: RCTime to Interval Scaling

As the microcontroller in use does not have floating point operations, we must instead use the star-slash (\*/) operation, as well as round the 19.8 into a clean 20.

Equation 1: Star-Slash ratio calculation

Therefore,

*Interval* =0.1476\* *RCTime* +19.852≈ *RCTime* \*/ 38 +20

Equation 2: Interval Calculation

Using this formula, we can accurately convert RCTime into Interval, which will directly control the angular displacement per cycle – in turn, the wiper will appear to move faster with a higher potentiometer value. Inserting a pause between each cycle will aid in smoothing out the wiper movement.

All that is left, then, is to add logic to switch direction at corresponding angles. Using a for loop on the duration of movement, stepping by the interval, up to a maximum of 160° or minimum of 15°, we can ensure that the wiper will move the full range. The program will step out of the loop, and be redirected to the other direction. Of course, this interval can be controlled by the potentiometer during operation.

# Expected Results

**Rudder**

It was expected that the servo motor moved smoothly and responded correctly to the potentiometer input. Specifically for the rudder controller, the server should move from 45° to 135° in a linear relation with the potentiometer being turned from fully left to fully right.

**Windshield Wiper**

Expectations of the wiper simply include a motion from 15° to 160°, then returning back to 15° and looping forever. During operation, the user will be allowed to alter the speed of the wiper by means of a provided potentiometer.

# Experiment and Design Revisions

**Rudder**

The code was written and compiled into the microprocessor, meanwhile the circuit was designed as shown in figure five. After these steps, we ran the code and found that our wiper was slightly offset. This was fixed by unscrewing the servo fan and readjusting it. Afterwards, the system worked as designed.

**Windshield Wiper**

The circuit was left the same, as shown in figure five, the only changes involved the code. Once completed, the program was loaded onto the board and run.

Here, we encountered several issues. First and foremost, we found that the battery was not at a suitable charge to run the servo at the best quality. This issue was resolved by hooking the board into a DC power supply finely tuned to simulate a nine-volt battery. This way, we had a steady supply of power to run the system.

In addition, we also had issues with our timing and interval. The servo moved very sporadically and had too great a range of speed – from very fast, yet incomplete, revolutions to oppressively long rotations. Thus, we experimented and found a balance of pause timings and interval scaling that worked – these final findings are the numbers and formulas that were discussed in the procedure

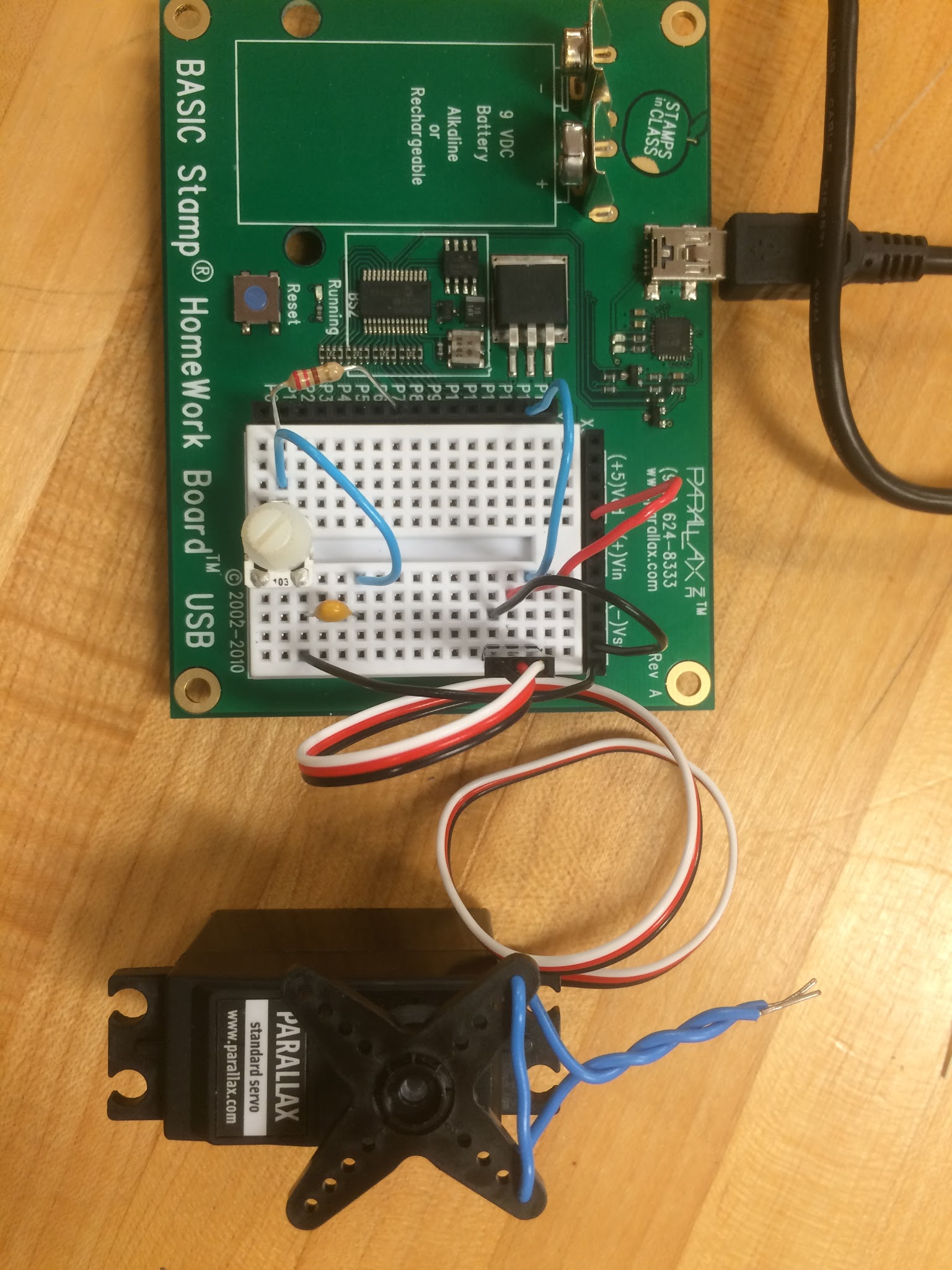


Figure 5: Completed Circuit

# Observations

Over the course of the two portions, the biggest observation made included the sensitive relationship between the servo movement and pause timings. It was noticed that, given too short of a pause, the servo could not position itself fast enough before the next input. This led to issues like sporadic jerks of motions, to getting locked in an undesirable loop outside of the operating range (windshield wiper). It is hypothesized that the servo could not achieve the given position before the next signal was received, thus the servo had conflicting commands that would cause it to behave unpredictably. Further, it was also hypothesized that sending multiple square waves of the same position would aid in the accuracy of movement by allowing the servo time to reach position and adjust to the appropriate angle.

# Discussion

This lab provided an extensive insight into the world of servos. The new concepts of servo control were utilized fully to provide solution to the problem statement. Further, the servos themselves were a first look into working with equipment; specifically, we found that the equipment will not always work as designed in the code. This lab taught us that care needs to be taken when working with these pieces, as they do not operate as fast or with as much precision as the microcontroller does – thus we must design systems that take into account the slowest piece of the puzzle.