Task 2.2.1

Tactile Stimulation Team

Microcontroller Coding Design Write-up

Jacob Sickafoose

February 27, 2022

1 Introduction and Overview

This document will serve as a walk through of all decisions made in the 2.2.1 task. It will serve as a guide as to how the decisions were made and everything that may need to be remembered and documented moving forward. It will be broken down into sub-tasks, and each sub-task will be explained as thoroughly as possible.

Each of these sub-tasks will start by describing the predetermined goal of it, as well as the estimated work time. It will then describe the process of going through the task, and the final result. Overall, this document will serve as a deliverable for each of the sub-tasks in the 2.2.1 task.

2 Sub-Tasks

I. Decide on a new Microcontroller (Est. Hrs: 10)

Description

The purpose of this sub-task was to narrow down what microcontroller(MCU) we will be utilizing for this prototype. If we were to to stick with the TeensyLC, we were going to specify with as great of detail as possible why we made the decision.

Walk through

We determined that the MCU we were using was going to be more than adequate enough to continue using for this prototype. The main issue seemed as though we needed to focus more on the electrode rather than the MCU of choice and it seems superfluous to be thinking about making the device wireless at this point. We also found that we can power the electrode using PWM, and the TeensyLC has amazing PWM support with a very high resolution giving us freedom in signal choice. It also has a whopping 10 PWM pins so if we decided to just do that, we could use 1 electrode per finger, each with their own PWM pin.

II. Create Block Diagram of Code Design (Est. Hrs: 2)

Description

The purpose of this sub-task was to create a new fully documented block diagram of the new code design.

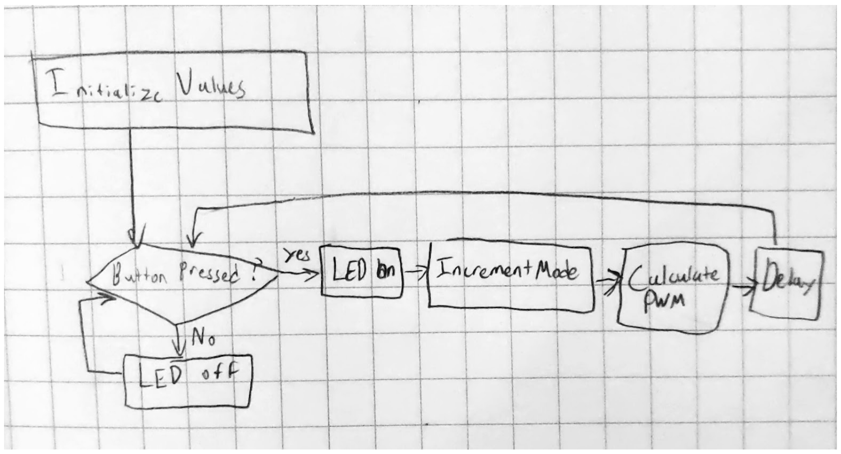
Walk through

I ended up with the following pseudo-code:

```
Initialize Pins

Loop
    if buttonPressed
        LED on
        frequencyMode++
        analogWrite(calculatePWM(mode))
        delay 5 seconds // Necessary for debouncing
    LED off
```

which translated into the following block diagram:



III. Complete the Code (Est. Hrs: 5)

Description

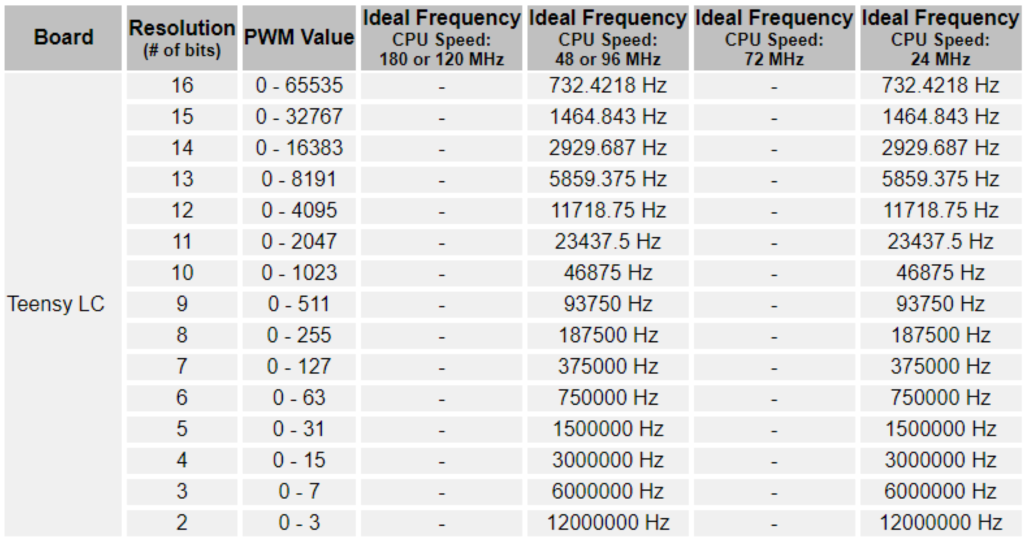
The purpose of this sub-task was originally to create each of the necessary helper functions we may need for the main code. However, the code once again seemed like it would be extremely simple and would be very clearly readable without any helper functions. I decided on making the main file to begin with, and then focusing on moving pieces to helper functions for legibility if necessary.

Walk through

Because we will now be using the TeensyLC to output a PWM controlled signal, I started by finding the built in library commands related to controlling the PWM.

I found, PWM is controlled with the analogWrite("pin", "value") function. "Pin" is used to select which pin will be used on the MCU and the function of the "value" field changes depending on which pin is selected. For the PWM pins, "value" is a number, 0 $\longrightarrow$ 256 which corresponds to the duty cycle. 128 $\longrightarrow$ 50%.

The frequency, or period is controlled by analogWriteFrequency("pin", "frequency") where the frequency is 488.28Hz by default. Here a value of "frequency" = 375000 corresponds to 275kHz. We can also change the resolution with analogWriteResolution("bits") with "bits" being any where from 2-16. Doing this changes the resolution for all the pins. Changing the resolution is where it can be tricky because raising the resolution, or how small the steps in the duty cycle are, lowers how fast of a frequency(or period) we can achieve. The following chart was included in the TeensyLC datasheet on PWM.



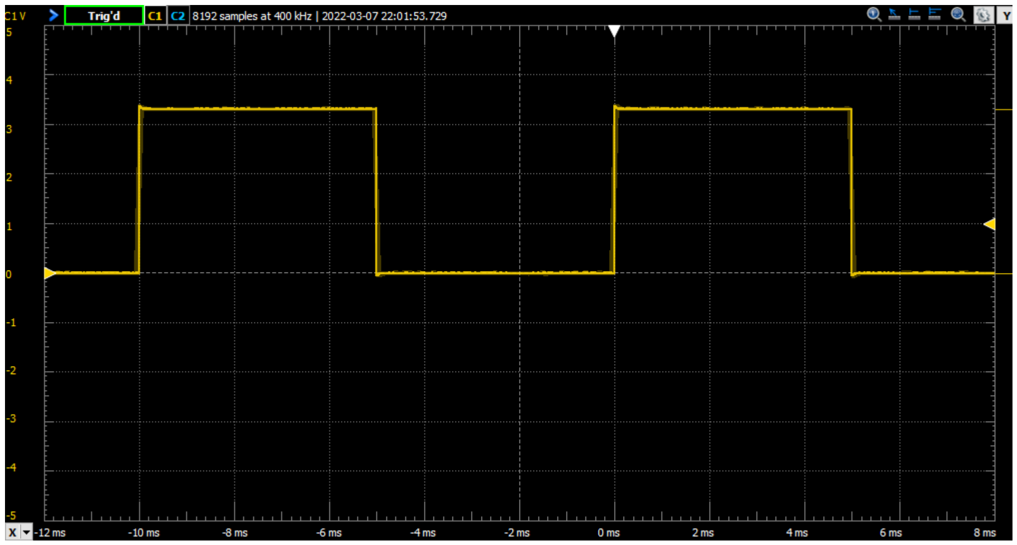
Board	Resolution (# of bits)	PWM Value	Ideal Frequency CPU Speed: 180 or 120 MHz	Ideal Frequency CPU Speed: 48 or 96 MHz	Ideal Frequency CPU Speed: 72 MHz	Ideal Frequency CPU Speed: 24 MHz
Teensy LC	16	0 - 65535	-	732.4218 Hz	-	732.4218 Hz
	15	0 - 32767	-	1464.843 Hz	-	1464.843 Hz
	14	0 - 16383	-	2929.687 Hz	-	2929.687 Hz
	13	0 - 8191	-	5859.375 Hz	-	5859.375 Hz
	12	0 - 4095	-	11718.75 Hz	-	11718.75 Hz
	11	0 - 2047	-	23437.5 Hz	-	23437.5 Hz
	10	0 - 1023	-	46875 Hz	-	46875 Hz
	9	0 - 511	-	93750 Hz	-	93750 Hz
	8	0 - 255	-	187500 Hz	-	187500 Hz
	7	0 - 127	-	375000 Hz	-	375000 Hz
	6	0 - 63	-	750000 Hz	-	750000 Hz
	5	0 - 31	-	1500000 Hz	-	1500000 Hz
	4	0 - 15	-	3000000 Hz	-	3000000 Hz
	3	0 - 7	-	6000000 Hz	-	6000000 Hz
	2	0 - 3	-	12000000 Hz	-	12000000 Hz

As described in the electrode signal characterization, we are aiming for a signal frequency of 100Hz which is fairly low. With a resolution of 16 bits, we can command the PWM value with 65535 steps. Converting frequency to period, 100Hz$\longrightarrow$ $10ms.10ms/65535steps = 15.3\mu s$. We want our duty cycle to go as low as $250\mu s$ so our PWM spec is surpassed with this resolution. The lowest resolution that could still achieve the 250μs minimum duty cycle at 10ms period is 6-bits, or 0-63 PWM steps.

Like in the last lab, I set it so that hitting an external button cycles through some signal outputs. For each of them, the frequency stayed at 100Hz. For the rest, I had modes 0-5. At the highest duty cycle mode, 5, the duty cycle was only 50% so I just had each mode add 10% to the duty cycle and it went back to mode 0 when the button was pressed and mode = 5. I also set it so that at mode 1, it has the period at 250μs

IV. Test Code (Est. Hrs: 2)

I measured and received the expected PWM output with the Analog Discovery 2. I was able to observe the signal changing as I hit the button. Here is the measured output when the mode was set to 50% duty cycle:


C1 V
Trig'd
C1
C2
8192 samples at 400 kHz | 2022-03-07 22:01:53.729
5
4
3
2
1
0
-1
-2
-3
-4
-5
X
-12 ms
-10 ms
-8 ms
-6 ms
-4 ms
-2 ms
0 ms
2 ms
4 ms
6 ms
8 ms