Closed-Loop Linear Motor Driver for Ultrasound Probe Stabilization

Julia Arnold (jul@mit.edu)

November 12, 2020

Abstract

This project developed a motor driver and controller capable of position control for a linear motor. PWM modulation creates sinusoidal phase voltages for the motor instead of square waves, which allows for much smoother movement and control. In future work, this could be used to provide constant, user-determined force on a surface regardless of hand position for ultrasound probe stabilization. This report includes accessible recommendations for future work in order to adjust to hardware limitations that have been met thus far.

1 Haptic Interfaces for Biomedical Applications

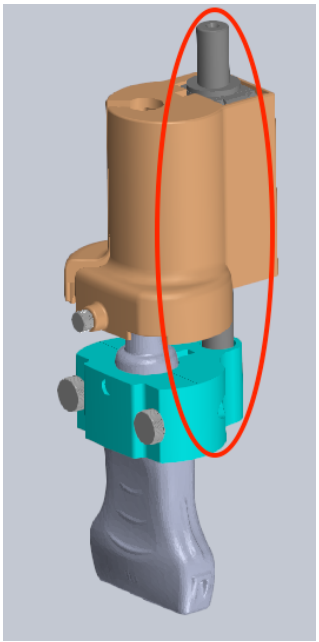

Figure 1: Proposed Stabilizing Handheld Ultrasound Probe System

When conducting ultrasounds, clinicians find it difficult to maintain constant probe force while also managing equipment settings.[1] A current MIT mechanical engineering Masters thesis project seeks to improve the quality of ultrasound images by developing an exterior device to steady the clinician's hand. The CAD model in Figure 1 shows the stabilizing product connected with a linear motor (circled in red). The clinician can hold the brown piece and the linear motor will exert a user-determined desired force through the green piece upon the ultrasound probe, regardless of the clinician's hand position. This technology will greatly improve ultrasound capabilities once deployed.

While the selected motor is sold with a control board (the MCLM 3003 P), the on-board microcontroller program is specific to the vendor, making it difficult to flex to the desired application. Moreover, it lacks the necessary flexibility and control needed for such a specific application; for example, there is no way to measure the actual motor current and factor this into a closed loop feedback system.

This project developed a motor driver and controller for a linear motor. In future work, this could be used to provide constant, user-determined force on a surface regardless of hand position. This report includes accessible recommendations for future work in order to adjust to hardware limitations that have been met thus far.

This presents the opportunity to make improvements in the medical field that could positively impact many patients, while incorporating power electronics, including a switched capacitor power supply, PSoC deployment, a closed feedback system, and a motor driver. This project builds on previous class work by implementing a closed feedback loop to make the motor exert a constant force at the output. Additionally, this uses a bidirectional motor which means the driver must have a way to force current to the motor in both directions.

2 Hardware and Software Description

At a high level, the constructed hardware is configured as shown in Figure 3. The switch cap converter divides the input 10V to power the switching logic of the buck converter. Next, the buck converter converts 10V to 5V to power the logic of the rest of the circuit. The L293 receives the two voltage supplies, as well as the control signals from the PSoC to output the three phase voltages to the linear motor.

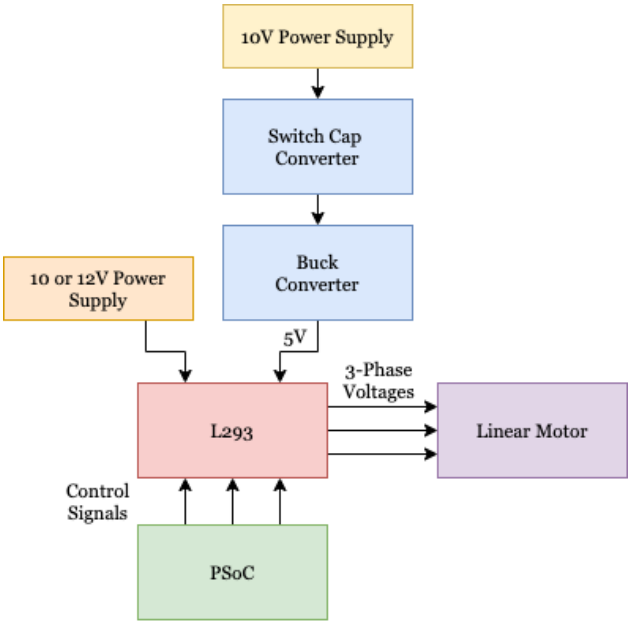

Figure 2: Hardware Block Diagram

In order for the PSoc to output these control signals, three PWM blocks running at 40 kHz have a duty cycle that varies from 0 to 1. A script varies this duty cycle in a sinusoidal pattern to create a smooth output control signal.

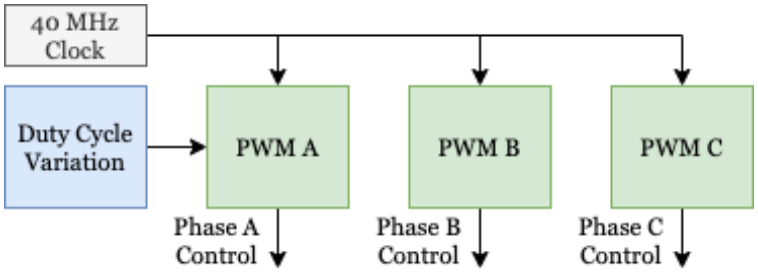

Figure 3: Software Block Diagram

This project used the following materials:

- L293 Quadruple Half-H Driver (Digikey)
- ACS712 Current Sensor on a breakout board, 2 (Amazon)
- LTC1044A 12V CMOS Voltage Converter
- PSoC
- Faulhaber LM2070 Linear DC-Servomotor with Analog Hall Sensors in Figure 4

- Assorted capacitors, MUR120s, FETs and resistors

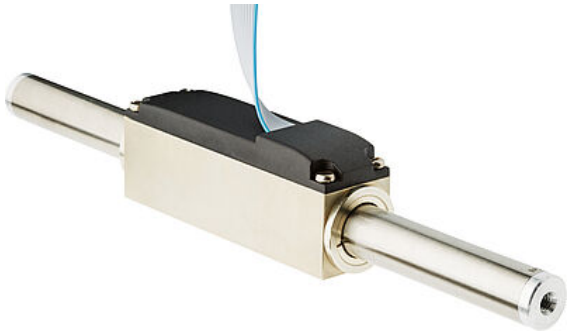

Figure 4: Faulhaber LM2070 Linear DC-Servomotor with Analog Hall Sensors

3 Switch Capacitor Converter

Switched capacitor converters, or charge pump circuits, can be used to invert a voltage or double it, as well as a wealth of other options introduced on the LTC1044A datasheet. The LTC1044A is a CMOS switched-capacitor voltage converter chip. It's good for supply voltages up to 12 volts, and down to 1.5 volts. It can invert the input voltage, double the input, divide it in half, or multiply it to a desired integer value. Not only does this chip simplify the process of changing a supply voltage, it is also very efficient, promising a minimum of 95% efficiency within typical operating points.

In general the switched cap converter works by charging one capacitor from the power supply, then switching its connection to the other side to transfer its charge to a load resistor and capacitor as shown in Figure 5. The current transferred per unit time (for a switching frequency f) is

$$i = f\Delta q = f \times C_1(V_1 - V_2)$$

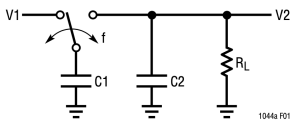

Figure 5: Switched Capacitor Converter Basic Theory/Model

The switching capacitor can be modeled as a resistor with a value equal to $\frac{1}{f*C_1}$. The loss is determined by the output impedance, and as frequency is decreased, the output impedance will be dominated by the equivalent resistance, which drops the efficiency. However, increasing the frequency increases switching losses, so the data sheet offers an efficiency vs. switching frequency curve to help find the sweet spot. The switching frequency can be raised, lowered, and driven from an external source using the OSC and Boost pins.

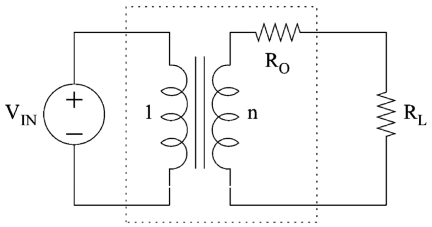

Figure 6: Model of an idealized switched-capacitor converter [3]

Also discussed this semester was how one can mathematically solve and understand this circuit and how it can perform such a miracle. Tellegen's theorem also applies, which shows that the sum of the voltage times the current in each branch will always sum to zero. In this application, the integral of the current is related to the sum the voltage and charge in the capacitors. With some clever algebra, it can be concluded that with the introduced switching frequency of the first capacitor, the circuit maintains an overall sum of 0 in all the branches of the system. This way power transfer can be modelled and shows that energy is neither created or destroyed.

The switch cap converter chip can also serve many other useful functions such as voltage dividing, and splitting a battery in half to a positive and negative rail, which may be useful later.

Brief research revealed the possibility of creating a more customized switched capacitor converter in order to specify the exact desired voltage relationship. Seeman and Sanders (2008) propose multiple topologies to

accomplish this. [3] However, the LTC1044A satisfied the specifications needed in the end, and it was not necessary to continue designing a customized switched capacitor converter.

This project leverages the voltage dividing capability of the LTC1044A in order to step the input voltage down to the logic level in order to power the logic circuitry of the buck converter that in turn powers the L293 and other components.

4 Design

The final schematic may be found at the end of this paper.

4.1 Switched Capacitor Converter

Figure 7 shows the suggested wiring of the LTC1044A from the datasheet. It promises .002% accuracy for up to 100 nA. While the buck converter has a higher current demand than that, the slip in accuracy observed upon wiring the chip for a test during design was not a concern since this is not used as a reference voltage. In summary, the schematic from the datasheet was used to create the splitter and provide 5V to the next stage of the design.

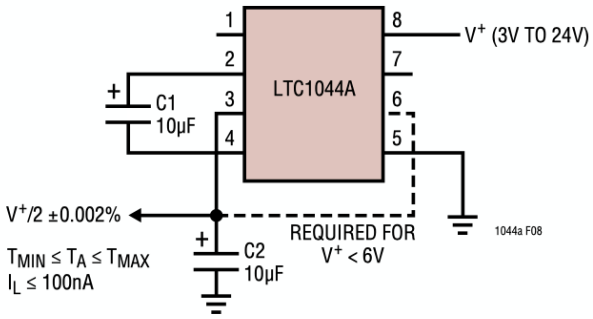

Figure 7: Switch Cap Splitter

4.2 Buck Converter

The buck converter was designed to step down 10V to 5V in order to operate the next stages of the hardware.

First we choose 250 kHz as a reasonable switching frequency because it is much higher than the audio range that we are processing so it will not introduce audible noise that we could hear at our output. It introduces a challenge because we must have semiconductors, like our diodes and MOSFETs, that are able to keep up with the switching frequency. Also, the capacitor on the 74HC14 gets smaller to make the frequency faster. We have a parasitic capacitance from the breadboard that will interfere with this value so we must remain an order of magnitude away from the 8 pF breadboard capacitance to avoid this affecting our actual frequency.

Next we choose the PFET to be an IRF9540 since it is available and good for fast switching, has a max V_{DS} of -100 V, max V_{GS} of $\pm$20 V, and max I_{DS} of 19 A so it can withstand our voltage/current ratings. Even within the maximum safe operating area graph, -12 V can withstand -20 A with a 10 ms pulse. In practice, V_{GS} threshold voltage ranges from -2 to -4 V and an on-resistance of .2 Ω. Delay/rise times (dynamic and for body diode) are on the order of nanoseconds, and thus can switch faster than kHz.

While NFETs have a better cost/performance ratio, we choose a PFET due to complications with a reference voltage and the body diode. In order to have the diode the right way, the source of the NFET would be connected to the junction where the inductor and the diode meet up. Then the source would be connected at the diode junction which is a "floating node" whose voltage changes (sometimes 12, sometimes zero). To drive the FET, it gets complicated with a IR2125 and a way to energize the bootstrap capacitor. The PFET can use 12 V as its reference since we can make the drain a stable voltage and vary the source at a lower voltage than the drain.

For the diode, a MUR120 will work well because of the current tolerance. The MUR120 is rated for 1 A repetitive, 35 A non-repetitive.

In order to pick the inductor and capacitor, we consider maintaining CCM, the LC resonant frequency and the output voltage ripple.

In order to ensure that we maintain second-order filtering and linear output voltage versus duty cycle even at low volume levels that correspond to low output voltages and power levels, we check the following condition for Δi.

$$\Delta i = \frac{V_{out}(1-D)T}{L}$$

$$L = \frac{V_{out}^2(1-D)T}{2P_{out}}$$

$$L > \frac{1^2(1-.5)(1/250k)}{2*100mW} = 10\mu H$$

Next, when the load current is at its maximum, the output voltage ripple should never be larger than 10% of the maximum output voltage (5 volts) over an output current frequency range from 500Hz to 10kHz. We can model the impedance response including component ESR over frequency in Figure 8.

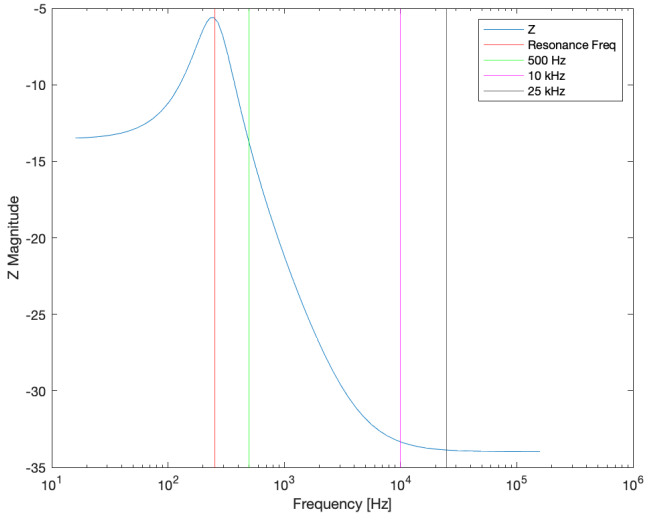

Figure 8: Z vs. Frequency of LC Filter

If we tried to place resonance between 10 kHz and 25 kHz, the impedance is larger, so we would have an undesirably larger ripple voltage. Instead it is easier to aim for below 500 Ohms. We ensure CCM and use this impedance check to see that the optimum inductance value is 200 uH and capacitance is 2000 uF with a 16V rating. Also included is a 1 uF capacitor to address a wider range of unwanted frequencies.

Lastly, we check the LC resonant frequency. This should be less than 10% of the switch frequency, which is 25 kHz. To check this:

$$\frac{1}{\sqrt{LC}} = 1580Hz < 2\pi 25kHz$$

Since this is an easy condition to satisfy, the choice was mostly driven by L and C values that were below 500 Hz. We want the resonance of the LC filter relatively low compared to the switch frequency of the buck converter because we want as smooth as possible of an output and to filter the higher frequency switching out [to adequately filter switch frequency ripple]. We also don't want the 30 kHz ripple from the totem driver. The goal is to produce as close to a DC output as possible so we remove all higher frequency ripples with the DC filer.

To create an inductor for circuit construction, a core and number of turns must be calculated. First we may compare inductance versus maximum current before saturation for each toroid. We assume that a "maximum" current is when the core permeability falls to 80 percent of its "no current" initial value. When this happens, H = 21 Oe. ℓ for T90 is 5.87 nm and ℓ for T106 is 6.49 nm, and I = $\frac{H\ell}{.4\pi N}$

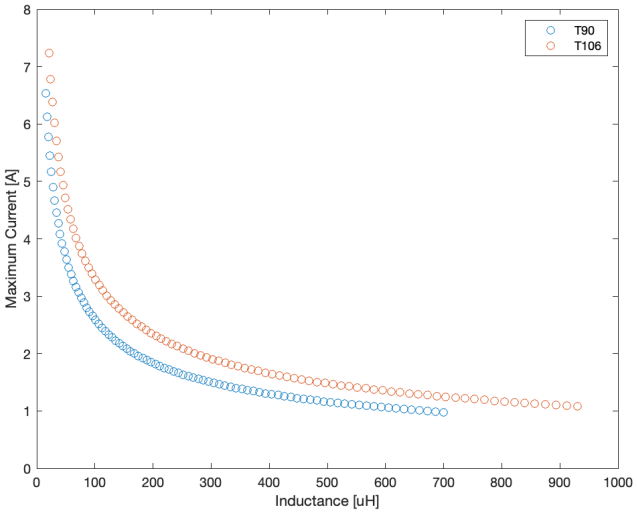

Figure 9: Maximum Current vs. Inductance

We see that with a T90 at 200 uH, a maximum current of 2 amps is possible before reaching saturation. While a larger toroid would lend more current tolerance and less turns, we can meet both of these requirements with the smaller, cheaper toroid. We only save 7 turns with the larger.

$$L = A_L N^2$$

where $A_L = 70$ nH/N^2 for T90. Thus, we need 54 turns for 200 uH.

Moving on to the switching logic, we choose the LM311 pullup resistor to be low because R is proportionally related to the time constant. The higher R is, the slower the output can switch.

LM311 needs 5 V to operate, but the output (which is just an open circuit when off inside the circuit) needs 10 V so that we can switch the PFET down the line faster. With more current, we better keep it out of a dissipative state.

The 1k resistor is a backup pull up for the output of the BJTs. In theory they should switch right at the same time, but in the case that both are off for a short time, the 1k provides a path to high V for the current returning from the PFET. When either is open, the 1k presents greater resistance than either of the BJTs so it will not have an effect during normal operation. In this way, we avoid having to deal with overcharge at the PFET gate each time the circuit switches.

Taking into account the power consumption of the LM311 output, the driver and the main body, I estimate P_{out}/P_{in} = 6W/8.39W = 72%.

4.3 Motor Driver

The next step, after establishing a power source for the motor driver, was to design the actual driver. Since the control is all taken care of on the PSoC, all that is needed is the L293 to provide voltage to the motor. The L293 can provide more current (up to 1A) than the L293D, making it more suitable for the higher powered motor in this project. This meant that physical diodes were needed, instead of being included in the package.

Additionally, a simple RC filter was added to each signal input in order to provide smoothing for the PWM input, as will be discussed next.

4.4 Control Scheme

For this project, a more complex control scheme was used to create a sinusoidal input to for the motor instead of a square wave. It also made switching the direction of travel very simple. The LTSpice model in Figure 10 demonstrates the concept. A sine wave, with three versions 120 degrees out of phase with each other, is compared with a reference ramp voltage.

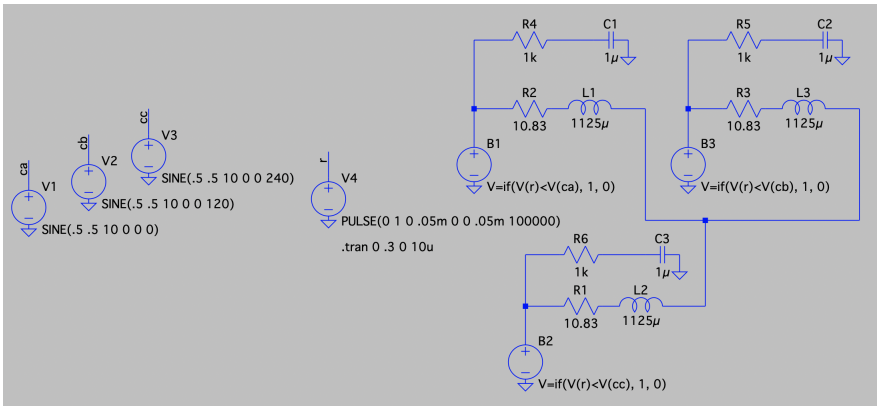

Figure 10: LTSpice Model of Control Scheme

The result is a sinusoidally varying pulse, which when filtered and put through a model of the motor's impedance, results in a smoothed sine wave as seen in Figure 11. This makes control simpler since we only need to adjust sine wave to change aspects of our motor output. The frequency controls the speed of the motor's movement. When this dips negative, it reverses motor direction. Current roughly correlates with the amplitude of the wave, also known as the depth of modulation.

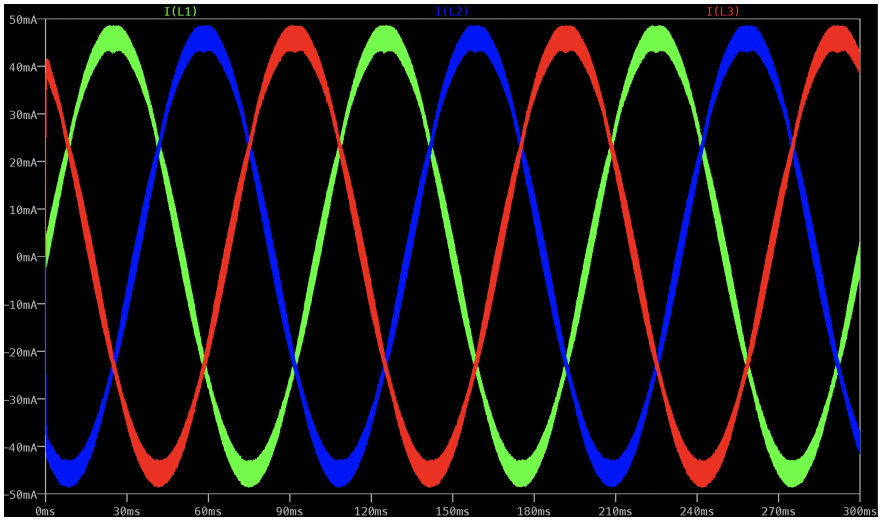

Figure 11: LTSpice Simulation Result of Current over Inductors

In order to leverage the PSoC capabilities and make the control even smoother, the next step was to use PWM blocks with varying duty cycles to create the effect of a sinusoidal control signal. Normally the duty cycle would vary sinusoidally with an amplitude of the depth of modulation and offset of 1/2. This gives a constant velocity that is dependent on speed.

In order to oscillate position of the motor, the duty cycle advancement was varied sinusoidally in order to create this effect. The resulting PWM wave duty cycle was simulated in MATLAB and is shown in Figure

12. The periodic reversing of the duty cycle is what makes the motor smoothly reverse direction as desired. The PSoC code for this is discussed in the Software section.

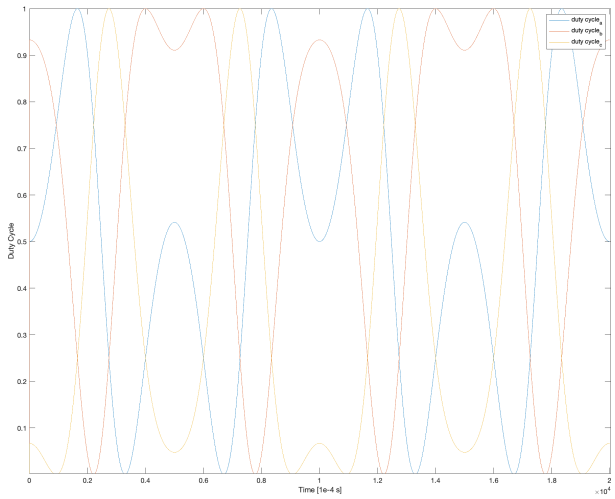

Figure 12: Duty Cycle Variation in MATLAB

5 Analysis

5.1 Switched Capacitor Converter and Buck Converter

Figure 13 demonstrates the constructed buck converter working to generate the desired voltage. The switch cap converter held its voltage well in order to power the buck converter logic. When measured at steady state, it was outputting 3.4 mA. With a 47 Ohm load, the output current is 105 mA, which gives us almost 5 watts at the output. The next stage draws 43.4 mW when on, and 51 mW when the PSoC is not outputting any signal. We expect this because the L293 datasheet says the chip draws up to 60 mA when all outputs are a logic low, which is more than the 22 mA maximum at logic high.

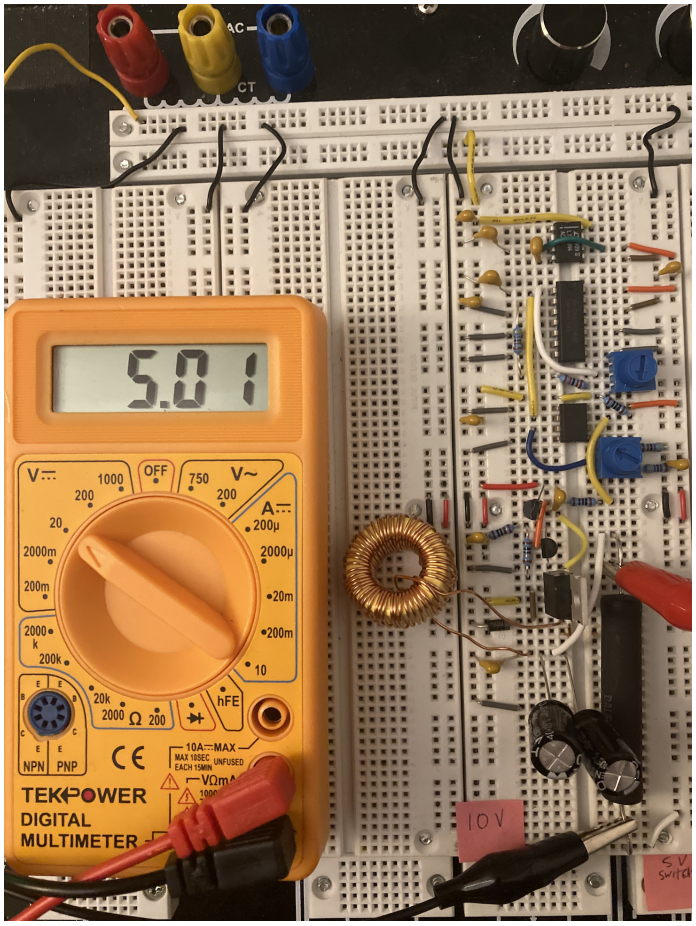

Figure 13: Constructed Switch Cap and Buck Converter

Figure 14 shows the switching of the PFET at the drain. There was a ringing on either side of the signal that was not optimal but still functional. The duty cycle is measured as 56%.

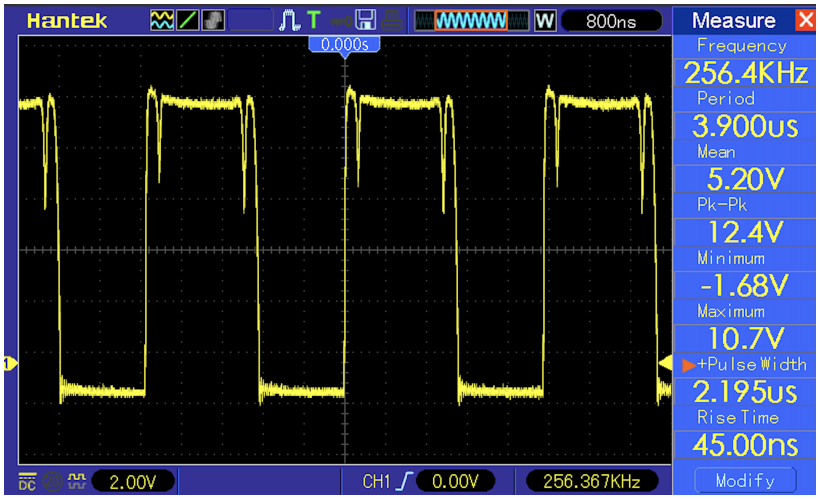

Figure 14: Buck Converter PFET Drain

Figure 15 shows the noise at the output of the buck converter. The ripple is less than 1V, which is 10%

of the input voltage.

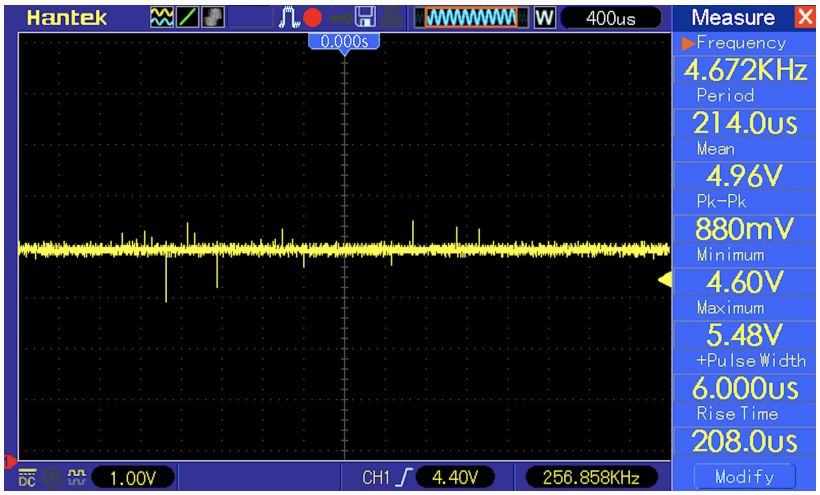

Figure 15: Buck Converter Output Noise

5.2 Motor Driver

The motor driving circuit is successfully powered by the buck converter in Figure 16. VCC2 of the L293 chip can be powered either by the variable 10V source or the side 12V source. Often the 12V source was more convenient as the variable source was used for testing other components, but overall the circuit could be powered from one voltage source with high enough current output.

The motor pulls about 450 mA of current at 10V when running, separate from the 5V converter source.

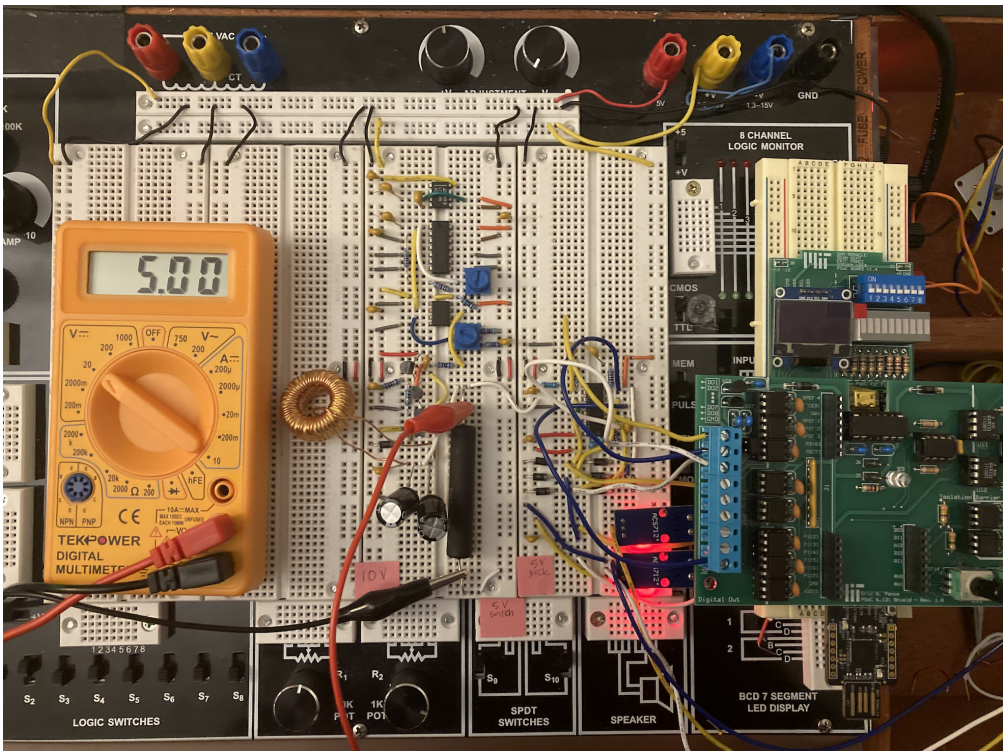

Figure 16: Constructed Driver Circuit

5.3 Control Scheme

Figure 16 also shows the PSoC connected to the L293 and the current sensors. The current sensors were a bit noisy and could have used an analog RF filter had their use been pursued further. The handy PSoC Counts to Volts feature made reading the voltage easier but not perfect.

As shown in Figure 17 which shows two phases of the PSoC output of a PWM wave with the duty cycle varying sinusoidally, constant velocity control worked very well.

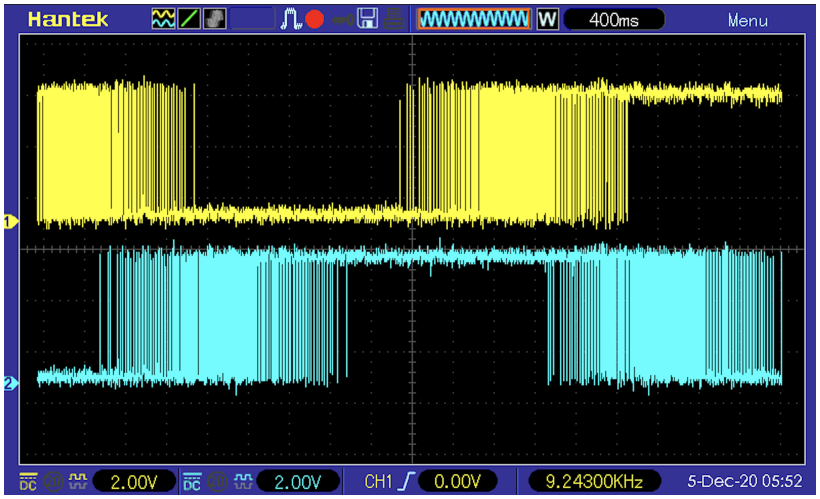

Figure 17: Constant Velocity Control, Phase A and B

After getting far along in the project, it was discovered that the PSoC did not have enough analog inputs in order to implement the intended control scheme. Since the motor is synchronous, knowledge of current position is required to adjust the phase angle of current being applied. However, this requires a Clarke transform of the 3 Hall sensors from the motor, and the PSoC only has two, even before including the current sensors.

Thus, the next best goal was to demonstrate position control. This was done by oscillating the frequency of PWM modulation wave, which worked very well. This gives an output as in Figure 19. If we call back to Figure 12, the duty cycle is varying in the expected sinusoidal pattern. The motion was slightly less smooth than before, perhaps due to the computation resolution.

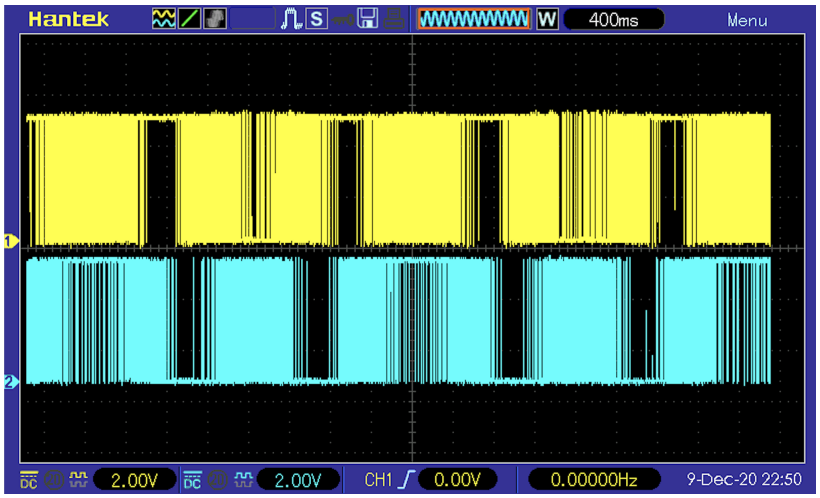

Figure 18: Oscillating Position Control, Phase A and B

6 Hardware Schematics

The schematic for all of the circuitry may be found on the last page.

7 Software

The documentation for the PSoC (Top Design and Code) in the implementation of position control is as follows. The use of the UART module to implement printf was invaluable in debugging this project.

```
#include <project.h>
#include <math.h>
#include <time.h>
#include <stdbool.h>
#include <stdlib.h>
#include <stdio.h>
#include <util.c>

int pwm_period = 600;
int interrupt_freq = 10000; //Hz

double mod_frequency = 10; //constant, in Hz
double mod_amplitude = 20; //constant
double dom = .5; // Depth of modulation between 0 and 0.5

double pwm_frequency = 0; //in Hz
double pwm_angle = 0;
double mod_angle = 0;

double d_a; //PWM duty cycle
double d_b;
double d_c;

CY_ISR(isr_1_handler) // Will run at a fixed 10kHz interrupt frequency
{
    d_a = .5+dom*sin(pwm_angle); //PWM duty cycle range from 0 to 1 around .5 at pwm_frequency
    d_b = .5+dom*sin(pwm_angle + 2.094);
    d_c = .5+dom*sin(pwm_angle + 4.189);

    // PWM duty cycle = period*duty cycle
    PWM_1_WriteCompare((uint16)(pwm_period * d_a));
    PWM_2_WriteCompare((uint16)(pwm_period * d_b));
    PWM_3_WriteCompare((uint16)(pwm_period * d_c));

    pwm_frequency = mod_amplitude*sin(mod_angle); //sign of pwm freq controls direction

    //increase the angle counter for overlayed modulation freq
    mod_angle = mod_angle + mod_frequency/interrupt_freq*6.283;
    //move the angle of the pwm duty cycle
    pwm_angle = pwm_angle + pwm_frequency/interrupt_freq*6.283;

    if (mod_angle >= 6.283){
        mod_angle = mod_angle - 6.283;}

    if (pwm_angle >= 6.283){
```

```
        pwm_angle = pwm_angle - 6.283;}
    else if (pwm_angle <= 0){
        pwm_angle = pwm_angle+6.283;}
    printf("%G, %G, %G \r\n", d_a, d_b, d_c);
    printf("pwm_angle, mod angle, mod freq, d_a: %G %G %G %G %G \r\n",
            pwm_angle, mod_angle, mod_frequency, d_a, d_b);
}

int main()
{
    CyGlobalIntEnable; /* Enable global interrupts. */
    PWM_1_Start();
    PWM_2_Start();
    PWM_3_Start();
    Clock_1_Start();
    Clock_2_Start();
    UART_1_Start();

    isr_1_StartEx(isr_1_handler); //start interrupt

    for(;;){}
}
```

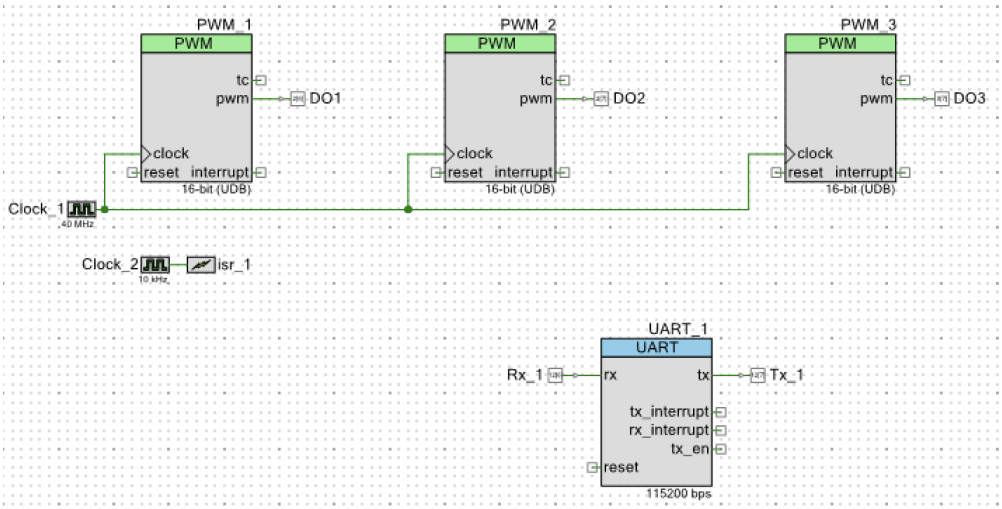

Figure 19: Top Design of Position Control Project

Code was also written to read current using the ADC CountsToVolts function and the ACS712 datasheet to convert voltage to current. This would have been used in the event that there were enough analog pins to do closed loop feedback.

```
    unfil_bit_a = ADC_SAR_1_GetResult16(); //void to int16
    unfil_bit_c = ADC_SAR_2_GetResult16(); //void to int16

    Filter_1_Write16(Filter_1_CHANNEL_A, unfil_bit_a);
    fil_bit_a = Filter_1_Read16(Filter_1_CHANNEL_A);

    Filter_1_Write16(Filter_1_CHANNEL_B, unfil_bit_c);
    fil_bit_c = Filter_1_Read16(Filter_1_CHANNEL_B);
```

```
printf("bits a: %i  %i \r\n", unfil_bit_a, fil_bit_a);
printf("bits c: %i  %i \r\n", unfil_bit_c, fil_bit_c);

voltage_a = ADC_SAR_1_CountsTo_Volts(fil_bit_a); //bit/V*filter scale
voltage_c = ADC_SAR_2_CountsTo_Volts(fil_bit_c);

 printf("voltage: %G  %G \r\n", voltage_a,voltage_c);

//voltage to current conversion from ACS712 datasheet
if (voltage_a < 1.5) {
    current_a = -5;}
else if (voltage_a > 3.5) {
    current_a = 5;}
else {
    current_a = (voltage_a-2.5)/.2;}

if (voltage_c < 1.5) {
    current_c = -5;}
else if (voltage_c > 3.5) {
    current_c = 5;}
else {
    current_c = (voltage_c-2.5)/.2;}

current_b = -current_a-current_c;
```

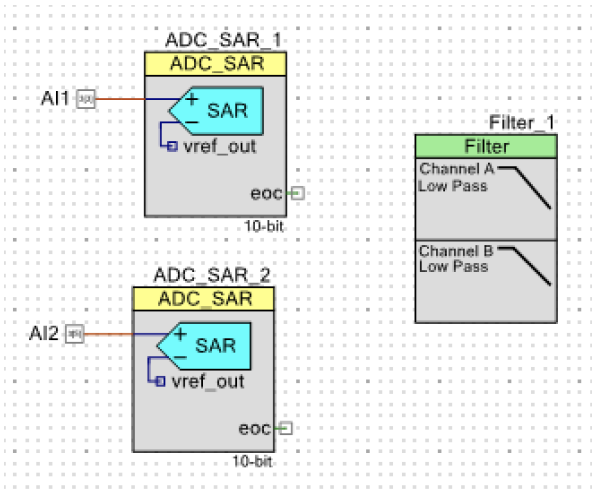

Figure 20: Top Design of Proposed Closed Loop Controller

8 Future Work

Due to limitations of the PSoC shield in the number of analog inputs, closed loop force control was not possible for this project. The motor is synchronous, and thus we need the three Hall sensor outputs to determine the current motor position to update the command current. The command current would vary between 0 and 90 degrees ahead in order to proportionally scale force. However, an Arduino script that implements closed loop feedback by producing a scaled PWM wave with duty cycle related to the desired voltage has been acquired. Bringing together the driver and microcontroller capabilities will take a bit of time but is generally straightforward based on the work that has been done here.

9 References

1. Chavez, Y., 2020, "System Identification and Control of a Miniature External Mechanical Vibration Device towards Clinical Ultrasound Shear Wave Elastography," Thesis, Massachusetts Institute of Technology.

2. Gilbertson, M. W., 2014, "Electromechanical Systems to Enhance the Usability and Diagnostic Capabilities of Ultrasound Imaging," Thesis, Massachusetts Institute of Technology.

3. Seeman, D. & Sanders, S., 2008, "Analysis and Optimization of Switched-Capacitor DC–DC Converters," IEEE Transactions on Power Electronics, Vol. 23, No. 2, p 841-851.