Lab #7:Final Project

Name: Shaiv Ramdhani

In [44]:
```
import matplotlib.pyplot as plt
import matplotlib.image as mplimg
import csv
```

In [45]:
```
def InlineImage(img, width=8, height=8, caption=None, fs=24):
    # Author: BG
    # Renders an image with a caption underneath using matplotlib
    # Requires matplotlib.pyplot to be imported as plt and matplotlib.img to
    # img - string, the location of an image file
    # width, height - numbers, the dimensions of the figure the image is dis
    # caption - string, the caption to be displayed beneath the image
    # fs - number, the font size of the caption
    fig,ax = plt.subplots(figsize=(width,height))
    ax.imshow(mplimg.imread(img))
    ax.set_xlabel(caption,fontsize=fs)
    ax.tick_params(
        which='both',
        bottom=False,
        top=False,
        left=False,
        labelleft=False,
        labelbottom=False)
    ax.spines["top"].set_visible(False)
    ax.spines["left"].set_visible(False)
    ax.spines["right"].set_visible(False)
    ax.spines["bottom"].set_visible(False)
```

1. Experiment Description

1.1: Description

The objective is to build a circuit capable of setting and actively controlling the speed of an electric motor. The circuit built will be able to control the revolutions per minute of an electric motor by electronically adjusting the current supplied to the motor. The variable input will be a potentiometer set voltage.

Latch and Reset Generator

The first step was to build out CLK input wave for the D-Latch, as well as the delay circuit and then reset pulse. The latch output is desired to be a square wave of 5 Hz

oscillating between 5V and 0V. We began by contructing the circuit seen in Figure 1. This is shown built in Figure 2. The datasheet for the schmit-trigger inverters is seen in Appendix A.

In [46]:
```
InlineImage("InverterNew.jpeg",width=8,height=8,caption="Fig. 1: Timing Cont
```

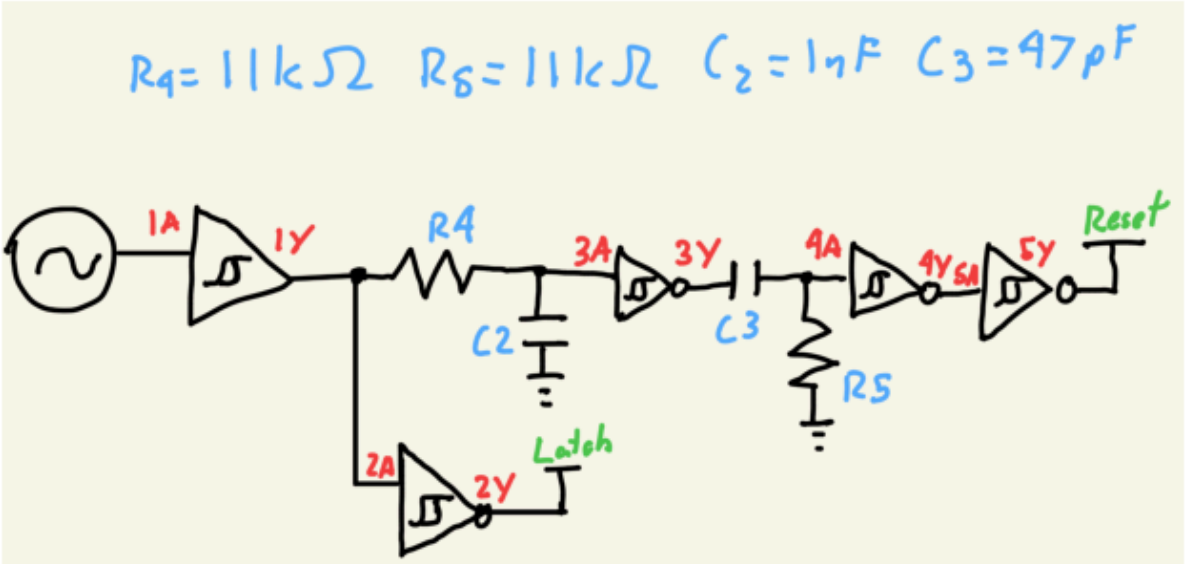


Fig. 1: Timing Control Circuit

In [47]:
```
InlineImage("InverterCircuitNew.jpg",width=8,height=8,caption="Fig. 2: Timir
```

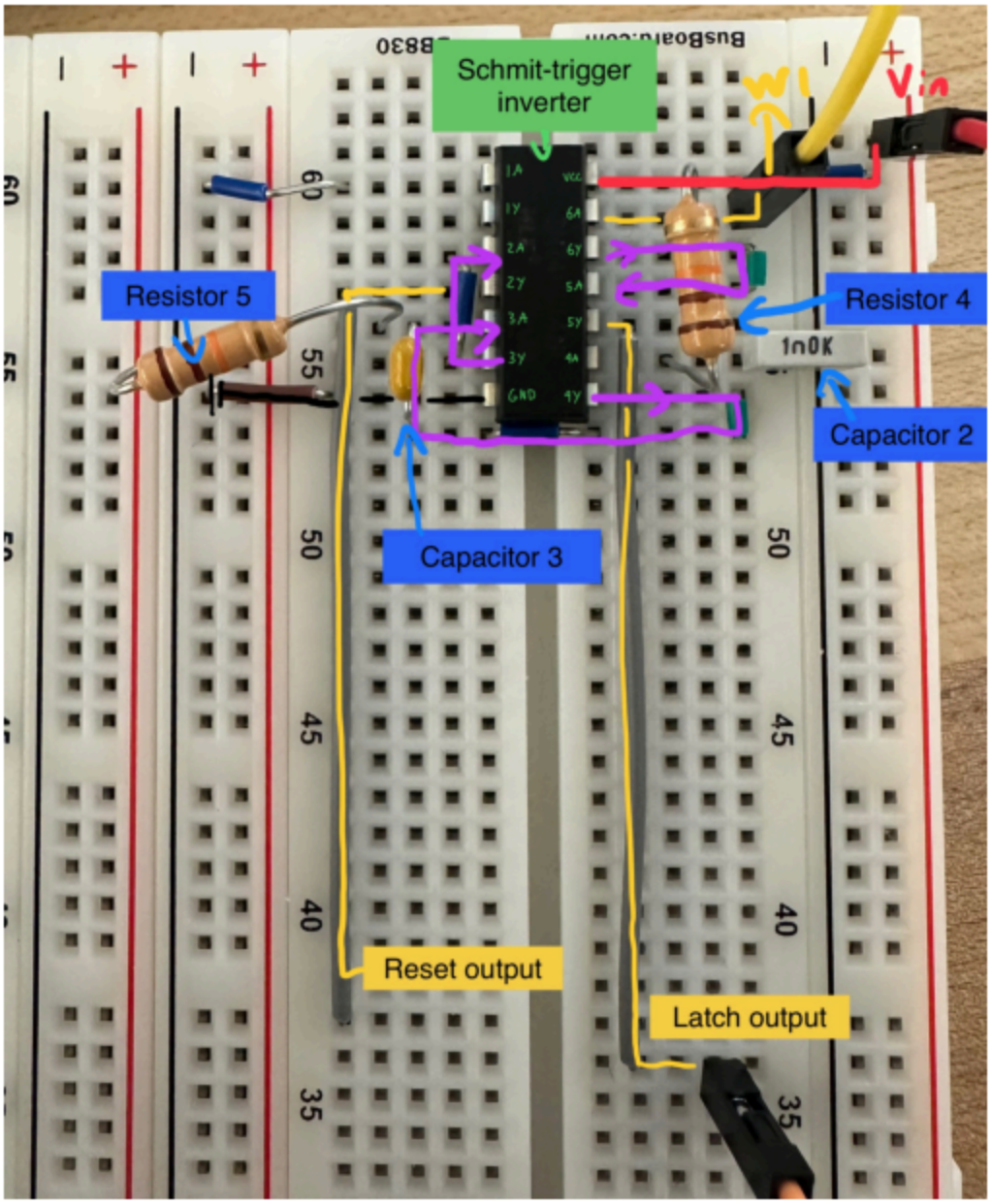


Fig. 2: Timing Control Circuit Photo

The delay circuit output can be seen in Figure 3. The purpose of this circuit is to introduce a small, predictable timing delay between the input signal and the output of the inverter. This delay is created by the RC network at the inverter's input. When the input square wave transitions, the capacitor does not change voltage instantly. Instead, it charges or discharges following an exponential RC curve.

Because the inverter is a Schmitt-trigger inverter, it does not switch the output immediately when the RC waveform begins to rise or fall. It waits until the RC voltage crosses its specific high or low threshold. This means the inverter only flips its state after the RC voltage has taken some time to reach that threshold. The result is a delayed, but still clean, digital transition at the output. In other words, the RC charging behavior

combined with the inverter's hysteresis produces a controlled timing delay between the input and output signals.

We can calculate the expected delay for this by using the RC discharge formula along with our hysterisis turn on voltage (Vt+). This was measured to be 3.1 ± 0.05 V. We can then use the formula $$t_{d}=-RC(ln(1-\frac{Vt+}{5}))$$. Plugging in our values with R being 11 kohms and C being 1 nF, this gives a delay of $11 \pm 1 \mu s$. When comparing to our actual value seen in Figure 3, we see a value of $12 \pm 3 \mu s$ which overlaps.

I then looked at the reset funcionality of the circuit by scoping the output.The reset pulse generator uses a capacitor to convert an input edge into a brief voltage spike, and a resistor to ground discharges that spike; the Schmitt-trigger inverter detects when the capacitor voltage crosses its threshold and turns that spike into a clean, narrow digital reset pulse.

This reset pulse should only occur on a rising edge. This is because when the capacitor is charging, the current is going to the resistor, creating a positive voltage at the inverter. However, when it is discharging, this terminal becomes negative instead.

TS - When I was originally trying to see the reset pulse, I wasn't able to see it. However, I realized that the pulse was incredibly short, and as a result I would have to zoom in quite a bit on the time scale to see a pulse of that small magnitude.

This can be seen in Figure 4.

In [48]:
```
InlineImage("FinalLabDelay.png",width=8,height=8,caption="Fig. 3: Delay Reac
```

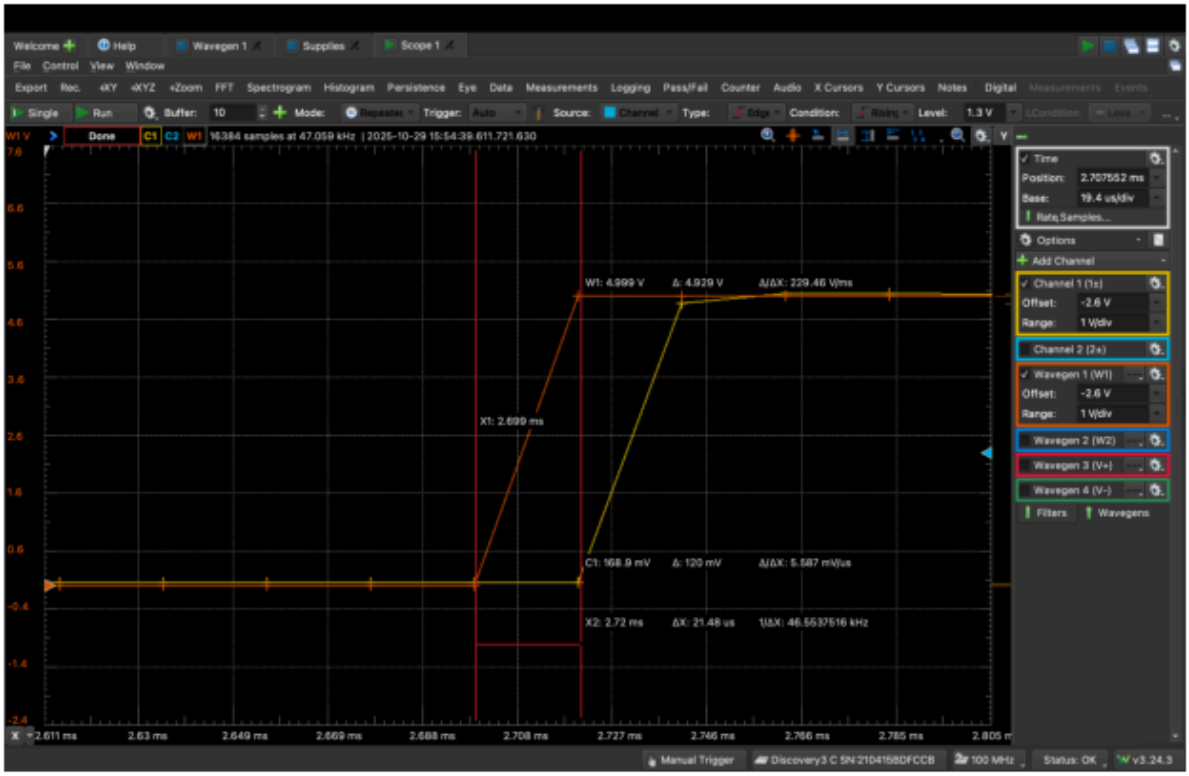


Fig. 3: Delay Reading

In [49]:
```
InlineImage("FinalLabResetPulse.png",width=8,height=8,caption="Fig. 4: Reset
```

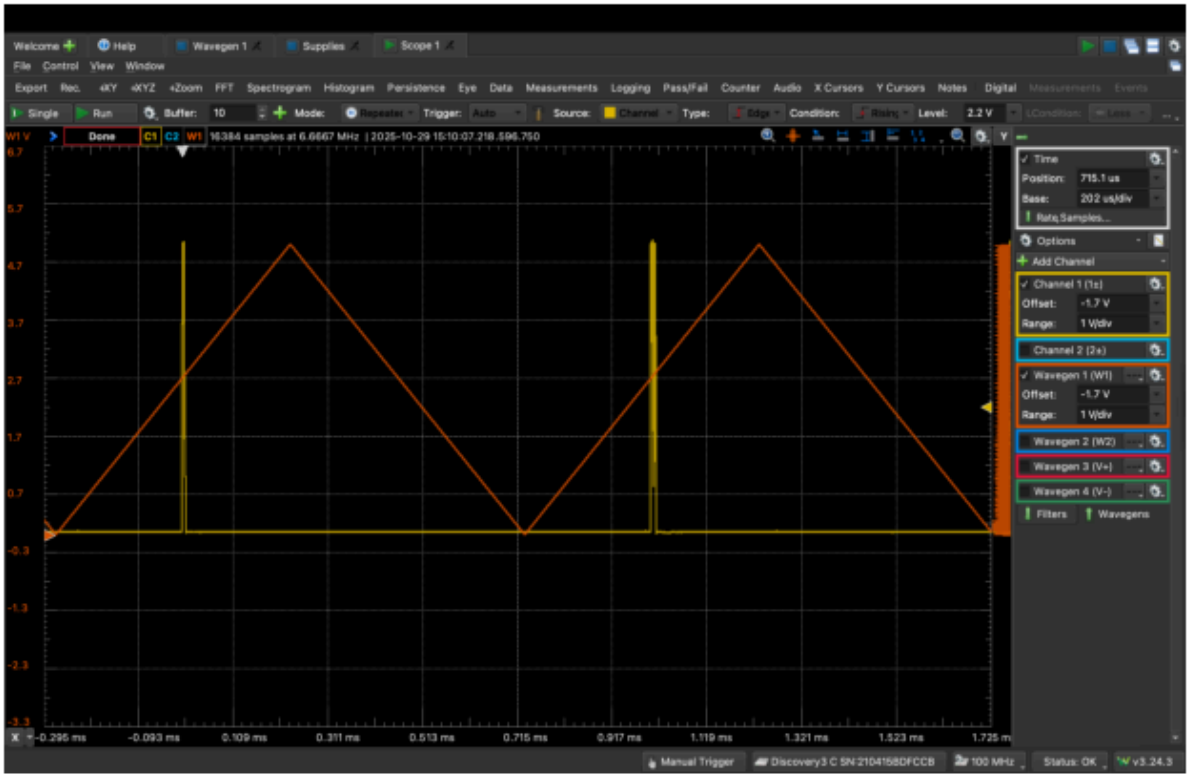


Fig. 4: Reset Pulse

Counter Based on the circuit in Figure 5, I wired up my divide 256 counter in the circuit photo below in Figure 6. The datasheet for the counter can also be seen in Appendix B. For testing, I used the logic analyzer and 8 D I/O pins. The least significant bit was set the output of 1QA, and the MSB was 2Qd.The resets were initially held low. As seen in the output photo Figure 7 , the counter could count count up by one up to a maximum of 255 before resetting back to 0. This is because the two cascaded ÷16 counters form an 8-bit binary counter, which can represent values from 0 to 255 ($2^8 - 1$). When the count reaches 255 (11111111_2), the next clock pulse causes a binary overflow, so all bits return to 0 and the counter continues from the beginning. This is due to the modulus behaviour of 256 and explains why the counter resets back to 0 after 255. I also tested the reset function by manually switching it in the Logic app using a DIO pin and manually changing it between high and low to trigger a pulse. For this, it was confirmed that resetting early before 255 set it back to 0.

In [50]:
```
InlineImage("Lab7Counter.jpg",width=8,height=8,caption="Fig.5: Divide 256 Co
```

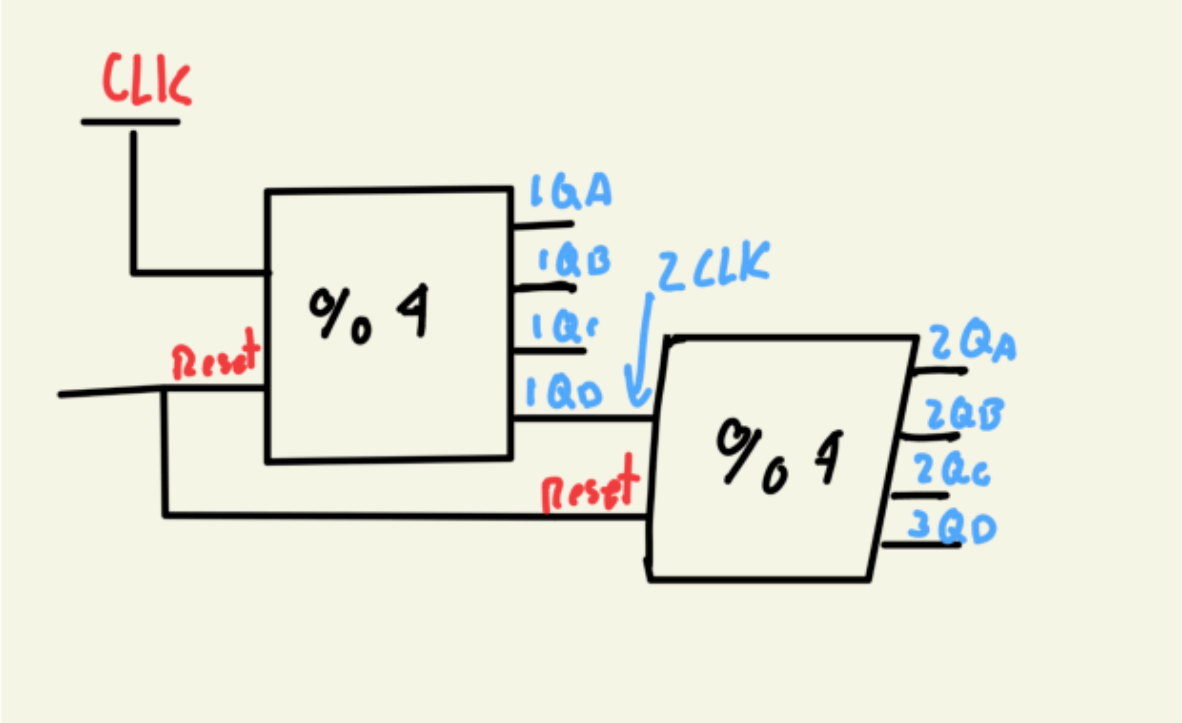


Fig.5: Divide 256 Counter

In [51]:
```
InlineImage("CounterPhotoNew.jpg",width=8,height=8,caption="Fig.6: Divide 25
```

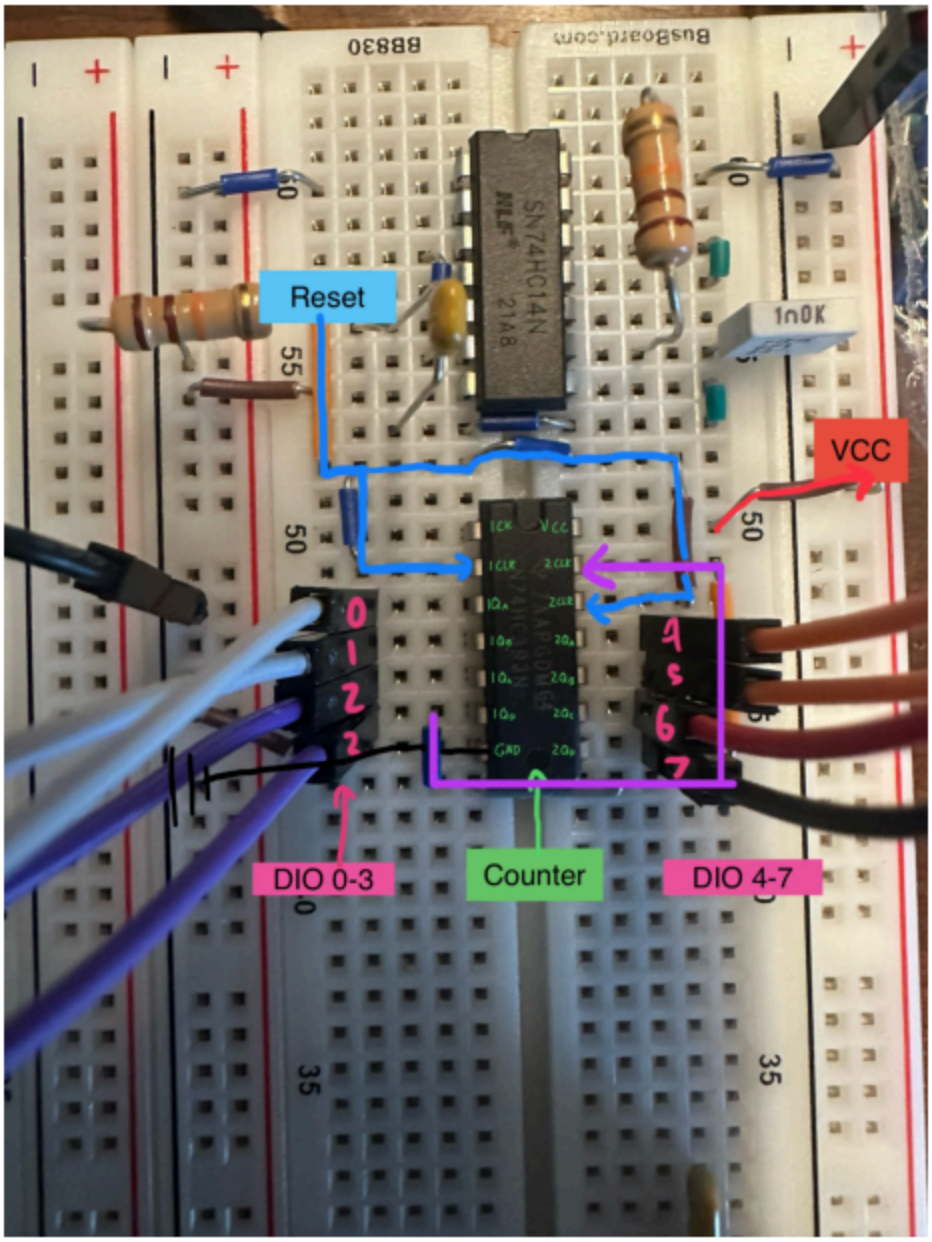


Fig.6: Divide 256 Counter Photo

In [52]:
```
InlineImage("Lab7Divide256Counter.png",width=8,height=8,caption="Fig.7: Divi
```

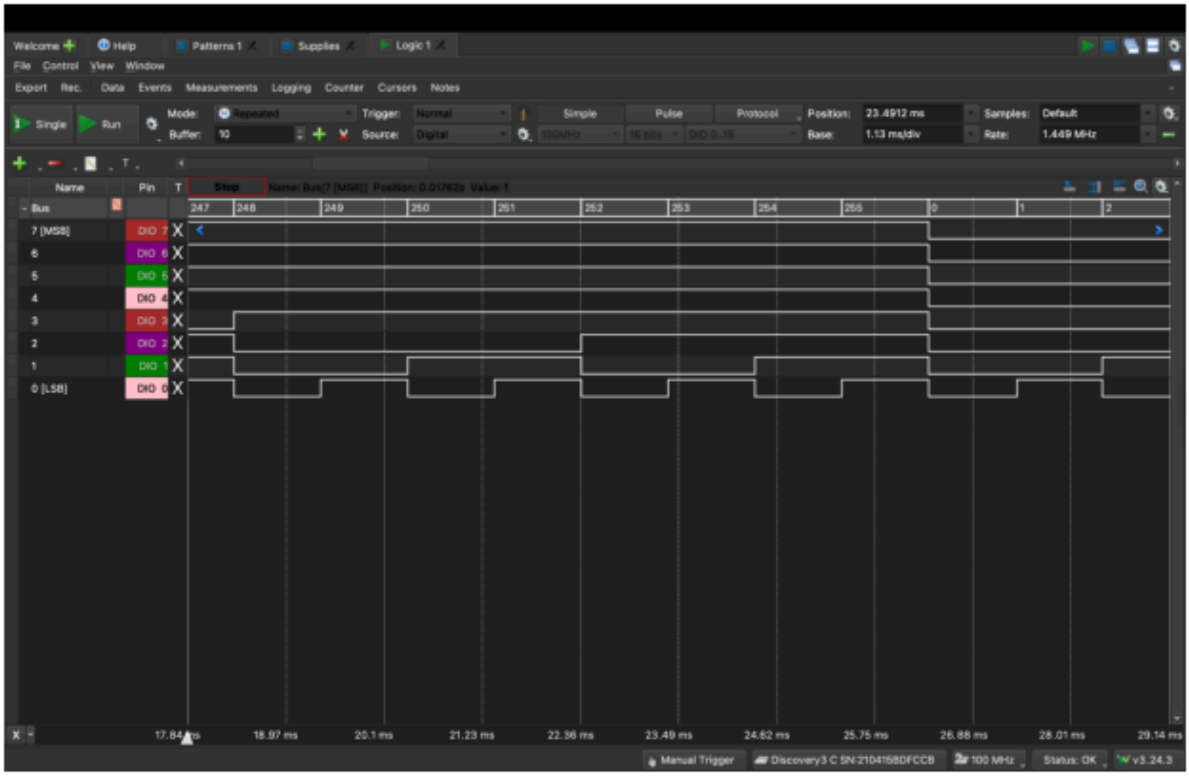


Fig.7: Divide 256 Counter

D-Latch

I next wired up the D-latch as seen in Figure 8, and built the circuit in Figure 9. The D-Latch pinout is seen in Appendix C. To test the functionality of the D-latch, I confirmed that I could "grab" a reading from the counter. This inovled adding a second wave pulse, directed into the clock of the counter. This was a 50 Hz square wave with a 2.5V amplitude and 2.5V offset. The idea behind this was that it should count the number of CLK pulses between RESET pulses and stores the result. Since the CLK pulses happen 10 times per every reset pulse (50 Hz / 5 Hz), the latch should store the value of 10. I then used 8 DIO pins on the output of the D-latch, from the first output being the LSB to the last being the MSB. From this I set up a bus on my logic analyzer.

TS - When initially trying to set this up, I was not getting the desired reading, but rather random fluctuations. To test this, I tested my counter first, testing the DIO pins both at the inputs and the outputs. I then tested my DIO pins at the inputs of the D-Latch rather than the outputs. This made it clear that I had

made a wiring error in between my counter and d-latch, which I was able to fix.

This output can be seen in Figure 10.

In [53]:
```
InlineImage("DLatchCircuit.png",width=8,height=8,caption="Figure 8: D-Latch
```

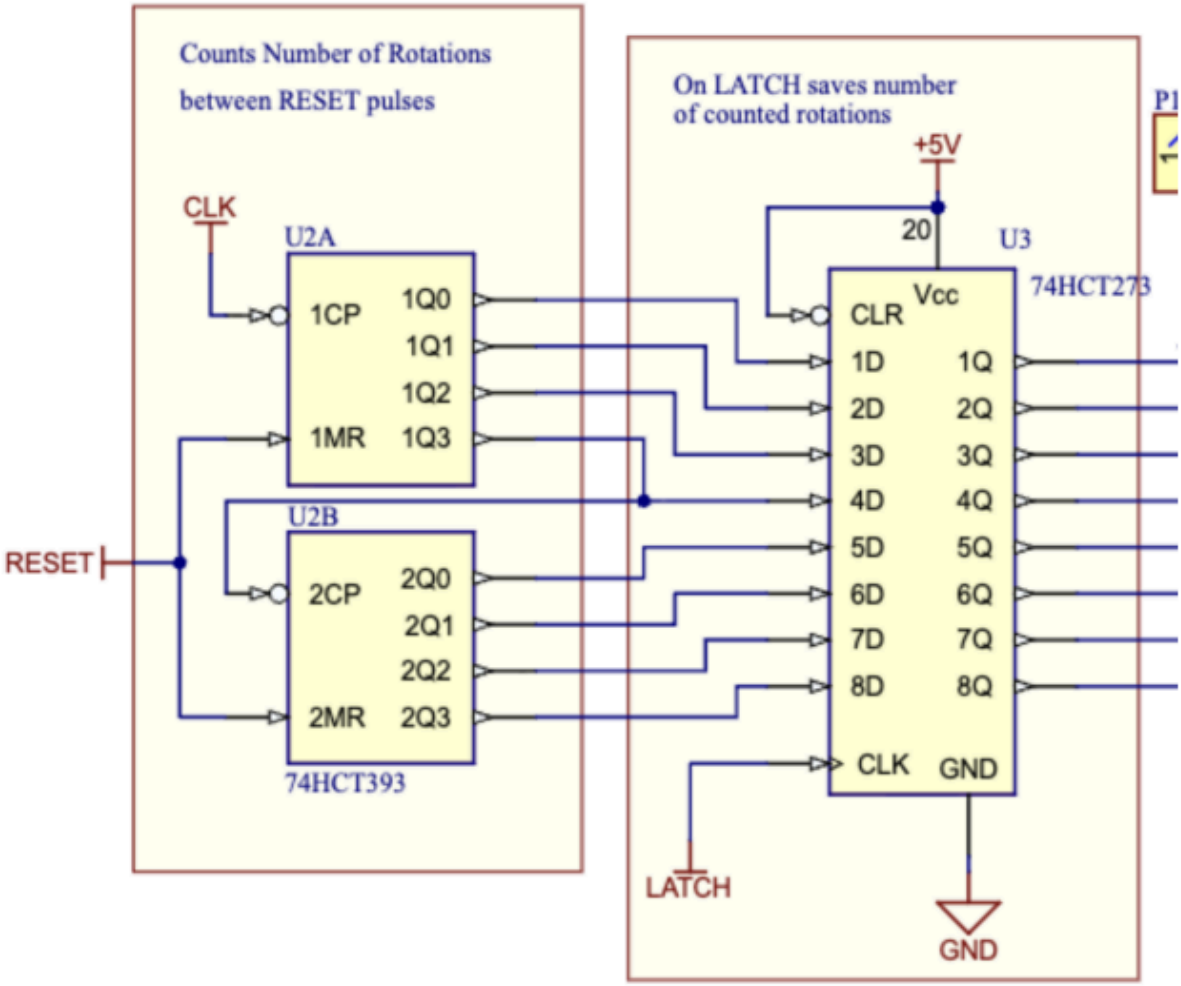


Figure 8: D-Latch Input Photo

In [54]:
```
InlineImage("DLatchPhoto.jpg",width=8,height=8,caption="Figure 9: D-Latch Ir
```

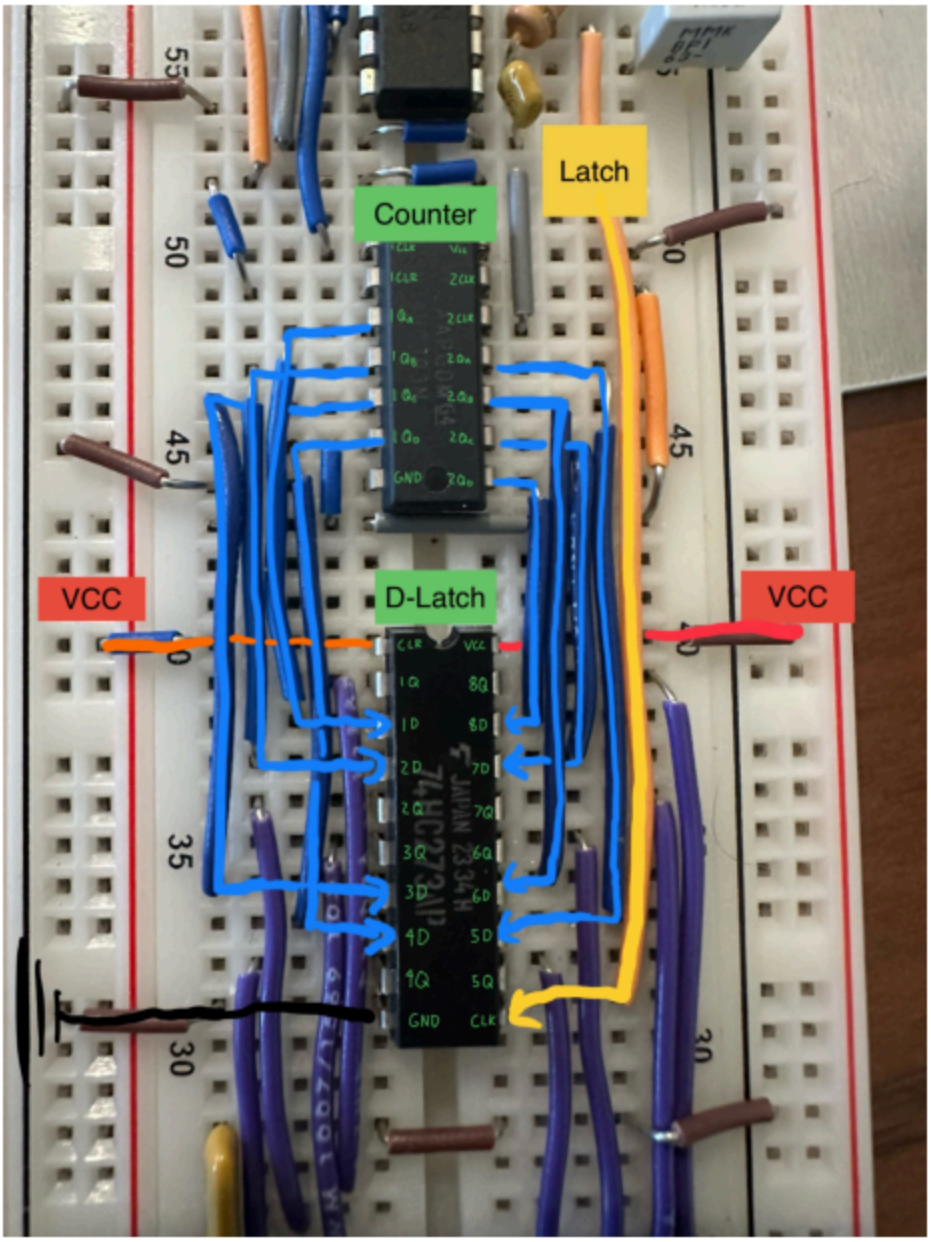


Figure 9: D-Latch Input Photo

In [55]:
```
InlineImage("D-Latch.png",width=8,height=8,caption="Figure 10:D-Latch Testir
```

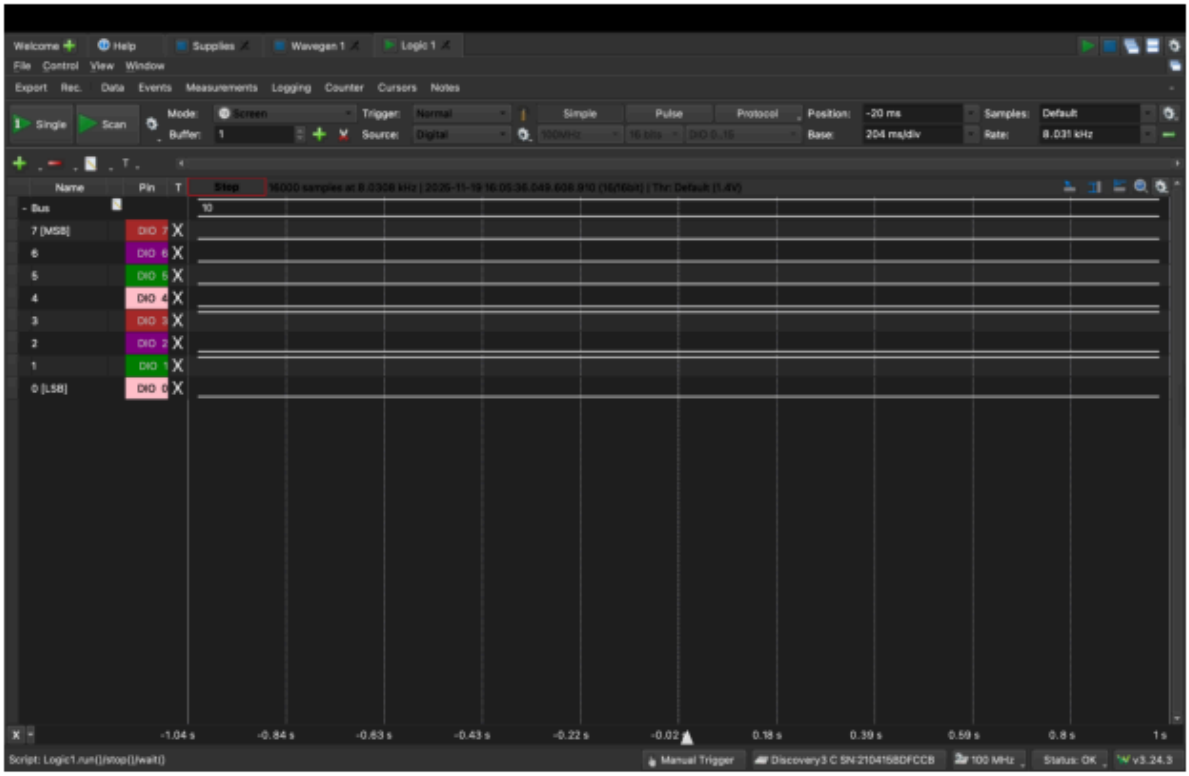


Figure 10:D-Latch Testing

DAC and Buffer

After the D-Latch, I set up the DAC R2R ladder in order to convert the latched binary/digital measurement into a an analog reading. Using the pinout in Appendix D and Appendix E for the DAC and Op-amp respectively, and the circuit diagram in Figure 11, I built the circuit and it can be seen in Figure 12.

In [56]:
```
InlineImage("DAC+BufferPhoto.png",width=8,height=8,caption="Figure 11: DAC+
```

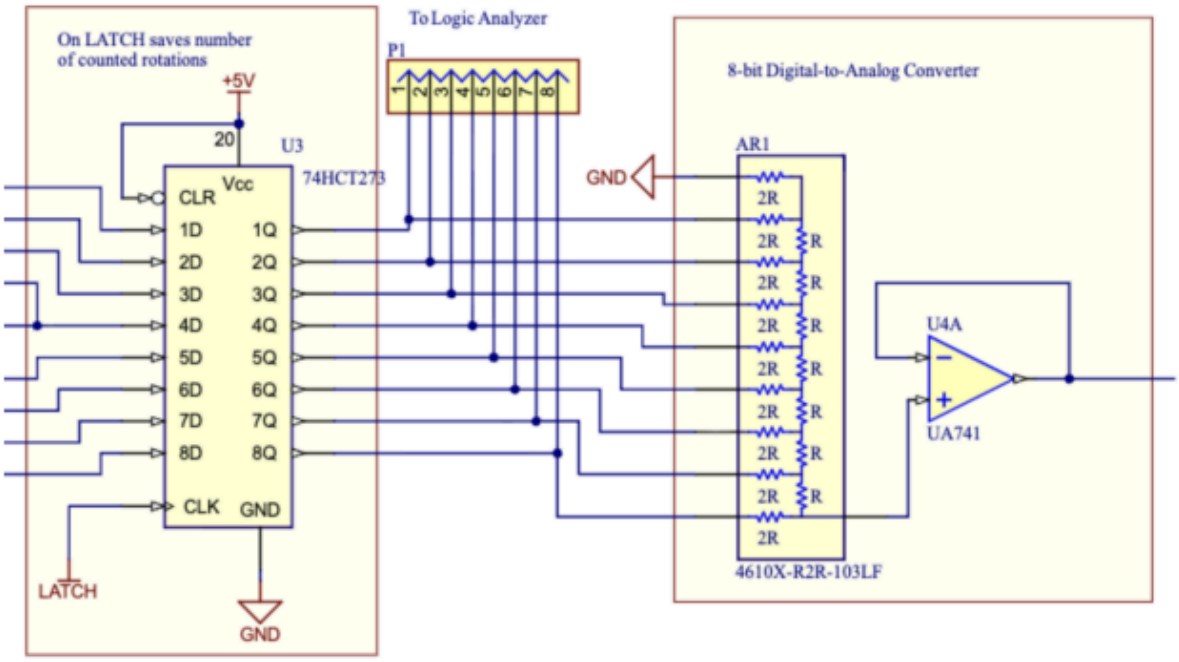


Figure 11: DAC+ Buffer Photo

In [57]:
```
InlineImage("DAC+Buffer.jpg",width=8,height=8,caption="Figure 12: DAC + Buff
```

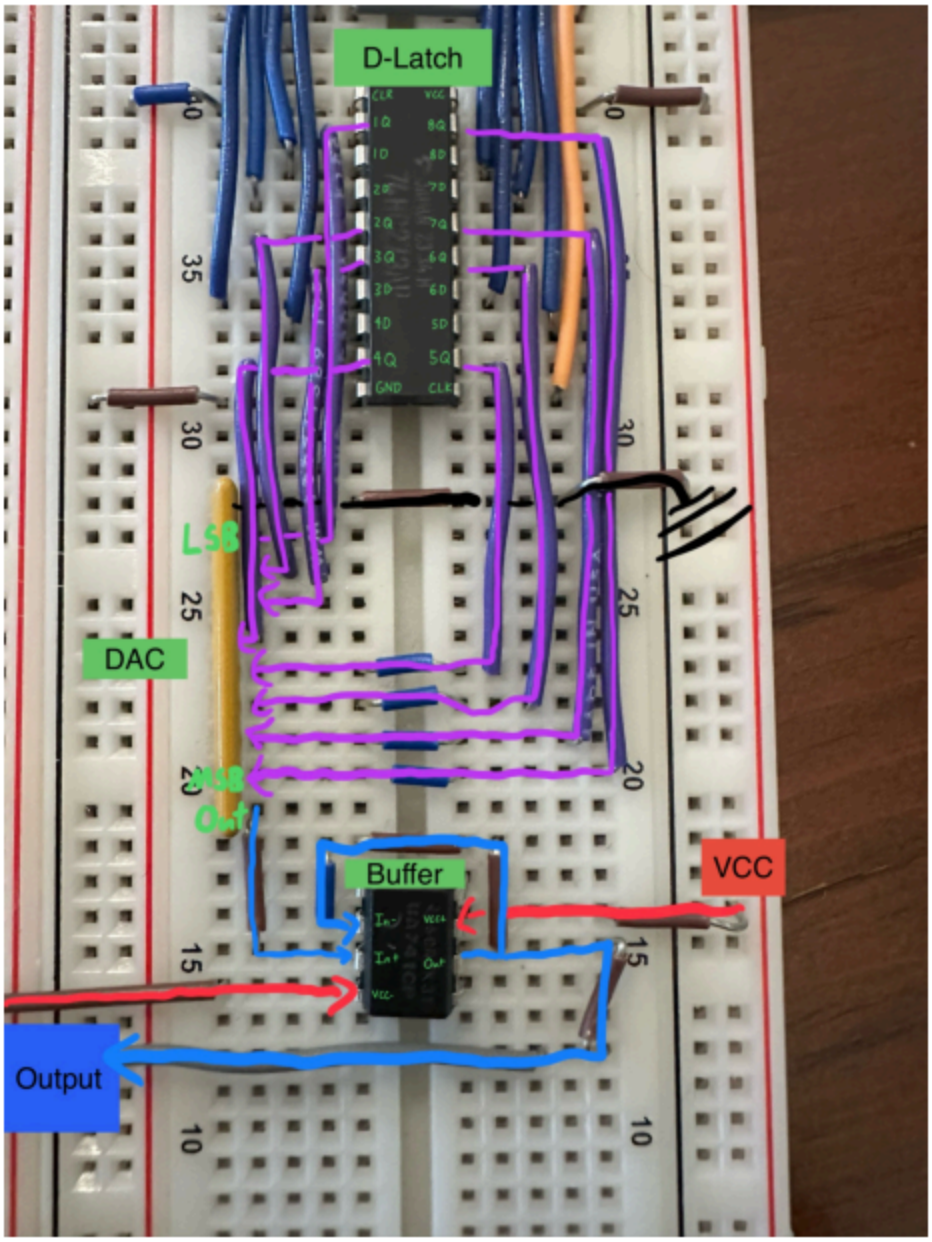


Figure 12: DAC + Buffer Circuit Photo

Since we are converting from digital to analog measurement using a R2R, we can use the formula we derived in a previous lab, where it is the latched number divided by the max number, and then scaled by the voltage level of a digital HI reading. This was 5 V in this case.

This becomes: $$ V_{out} = \frac{Latched Number}{255} * 5 V $$ Sticking with the 50 Hz wave provided to the counter, this meant our latched number was 10, resulting in a output voltage of 200 mV, verified below. The purpose of the buffer op-amp is to prevent any loading effects, meaning that the output of it should be the same as our DAC. This was confirmed, indicating this part of the circuit functioned correctly. A photo of this is in Figure 13.

In [58]:
```
InlineImage("op-amp output.png",width=8,height=8,caption="Figure 13:Buffer O
```

Figure 13:Buffer Output

Motor Sensor and Motor Driver

Error Signal Amplifier

The next part of the circuit was to construct the error singal amplifier as seen in the circuit digram in Figure 14. A photo of this can be seen in Figure 15. This potentiometer at the setpoint is what will later be used to control the motor speed.

In [59]:
```
InlineImage("ESACircuit.png",width=8,height=8,caption="Figure 14: Error Sign
```

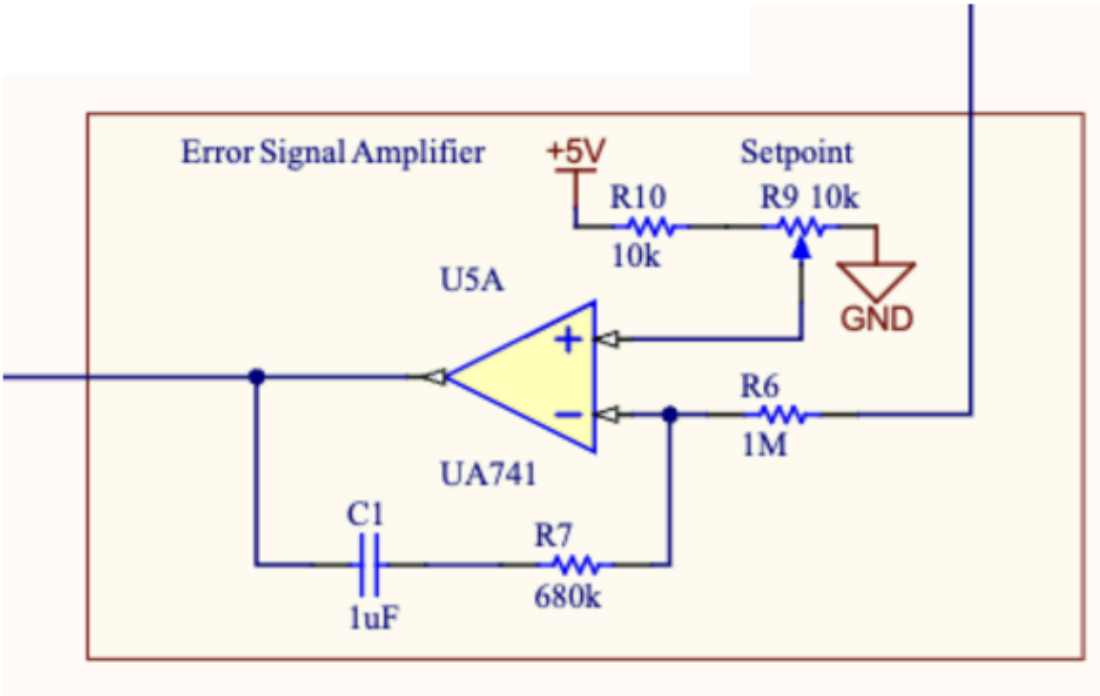


Figure 14: Error Signal Amplifier Circuit

In [60]:
```
InlineImage("ESAPhoto.jpg",width=8,height=8,caption="Figure 15: Error Signal
```

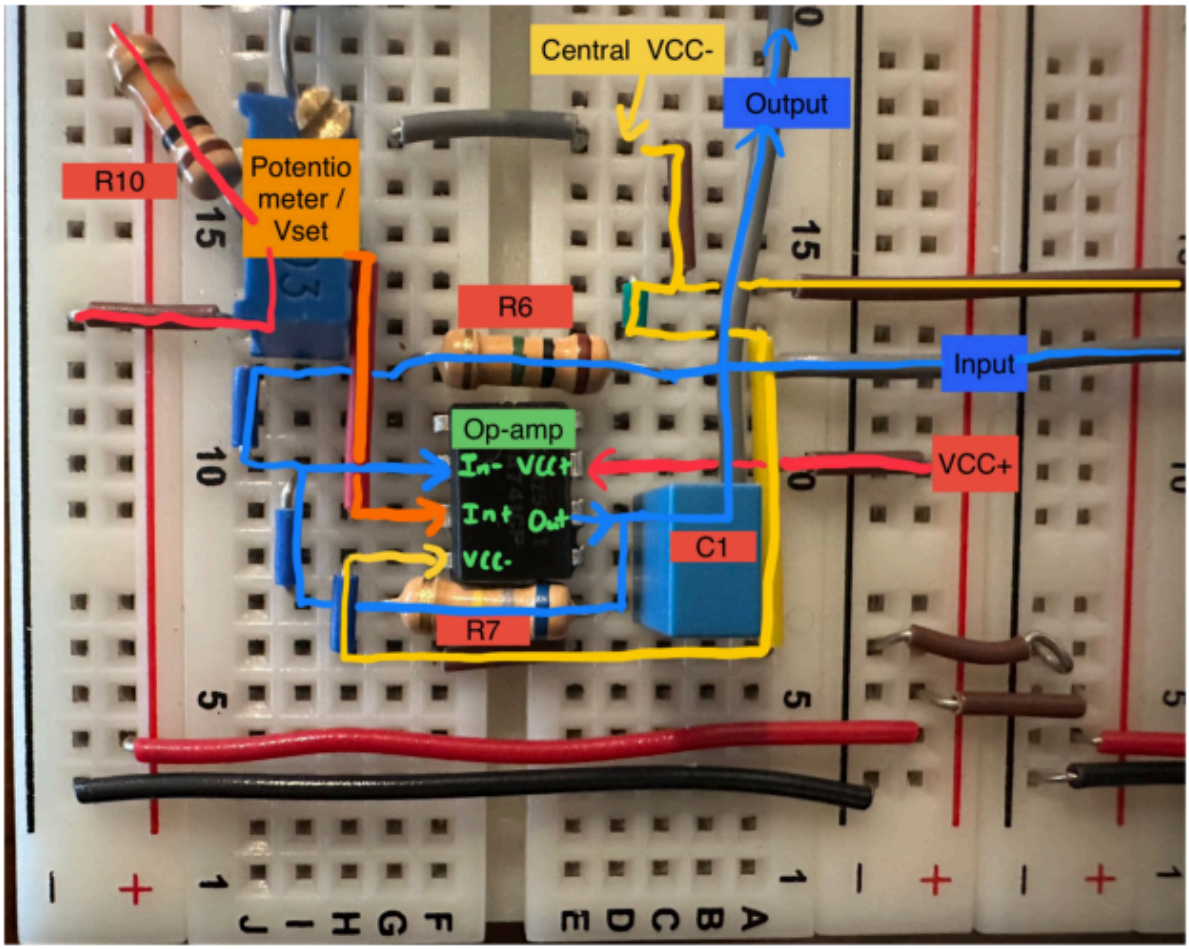


Figure 15: Error Signal Amplifier

In order to test this circuit works properly, it was important to derive an expression for the output voltage of this part. This was a function of the input voltage and the set voltage from the potentiometer. This was done below in Figure 16.

In [61]:
```
InlineImage("ESADerivation.jpeg",width=8,height=8,caption="Figure 16: Error
```

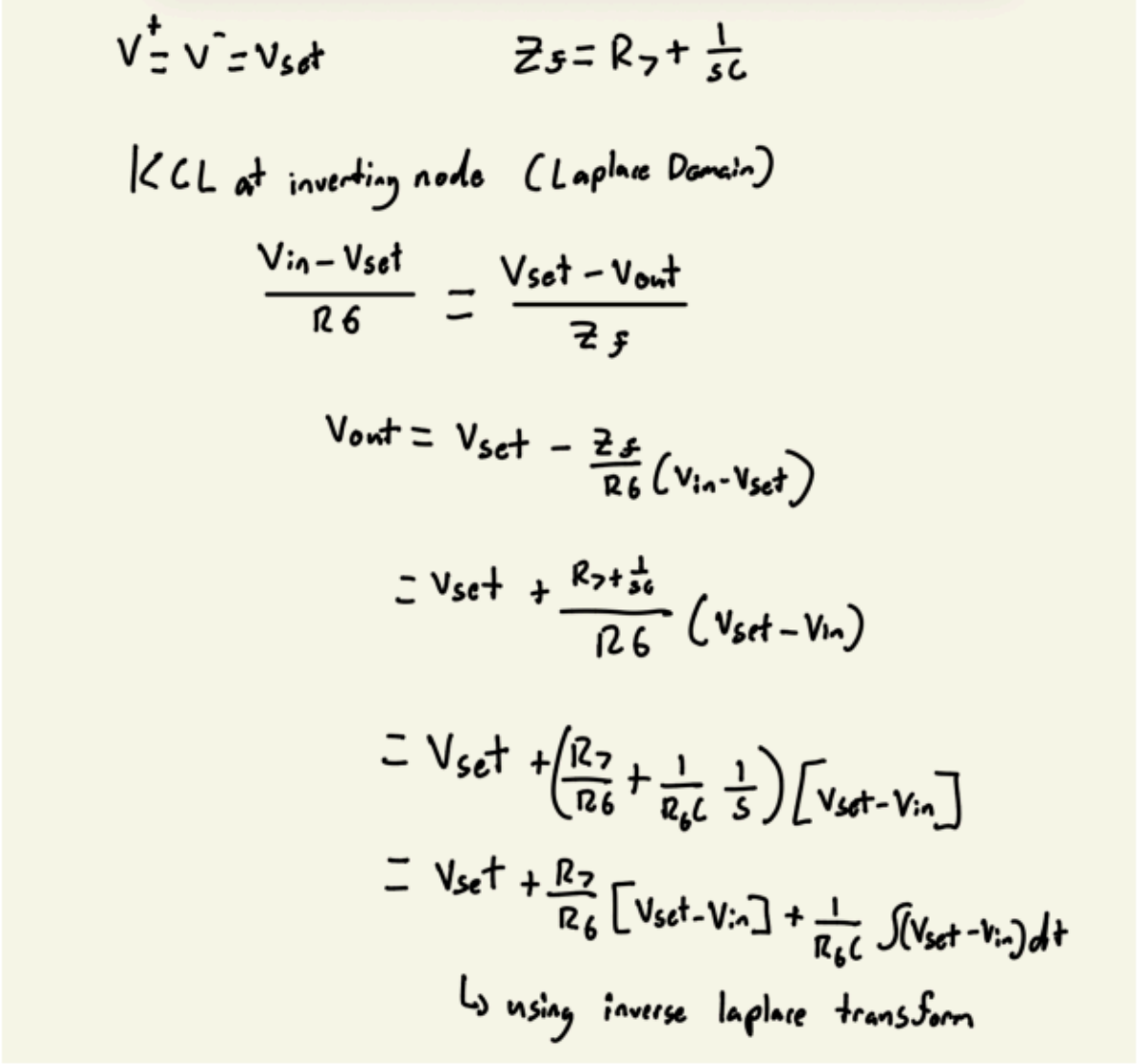


Figure 16: Error Signal Amplifier Derivation

The final expression is therefore $$ V_{out} = V_{set} + \frac{1}{RC}\int (V_{set} - V_{in})dt + \frac{R7}{R6}(V_{set}-V_{in})$$ The "error" signal that is being amplified is seen to be Vset-Vin, with both a linear and integral component.

I then set the Vset to 1 V before testing. To test the behaviour of the circuit, I applied a small square-wave waveform centred around the setpoint (Vin switching between 0 V and 2 V with a frequency of 1 Hz). I chose a square wave because its abrupt transitions clearly separate the proportional and integral responses. The proportional term causes an immediate step in Vout whenever Vin transitions, while the integral term produces a linear ramp between transitions. This should theoretically result in a triangular-shaped output. Since Vset = 1 V, a positive error (Vin < Vset) caused Vout to ramp upward, and a negative error (Vin > Vset) caused it to ramp downward.

The waveform output is shown in Figure 17. The observed waveform didn't quite match the expected response.Although there was the the clear jump and integrating term, I saw that the the output kept on the drifting to the positive rail and getting clipped. This is

likely because of the noise / slight variation in my Vset, resulting in it not being exactly 1.0000 V. Any small error would be integrated, and over time this would make the waveform output get skewed. Nonetheless, based on the visual observations made during the period where the output wasn't at the rail yet, I confirmed the validity of this circuit.

In [62]:
```
InlineImage("esa.png",width=8,height=8,caption="Figure 17: Error Signal Ampl
```

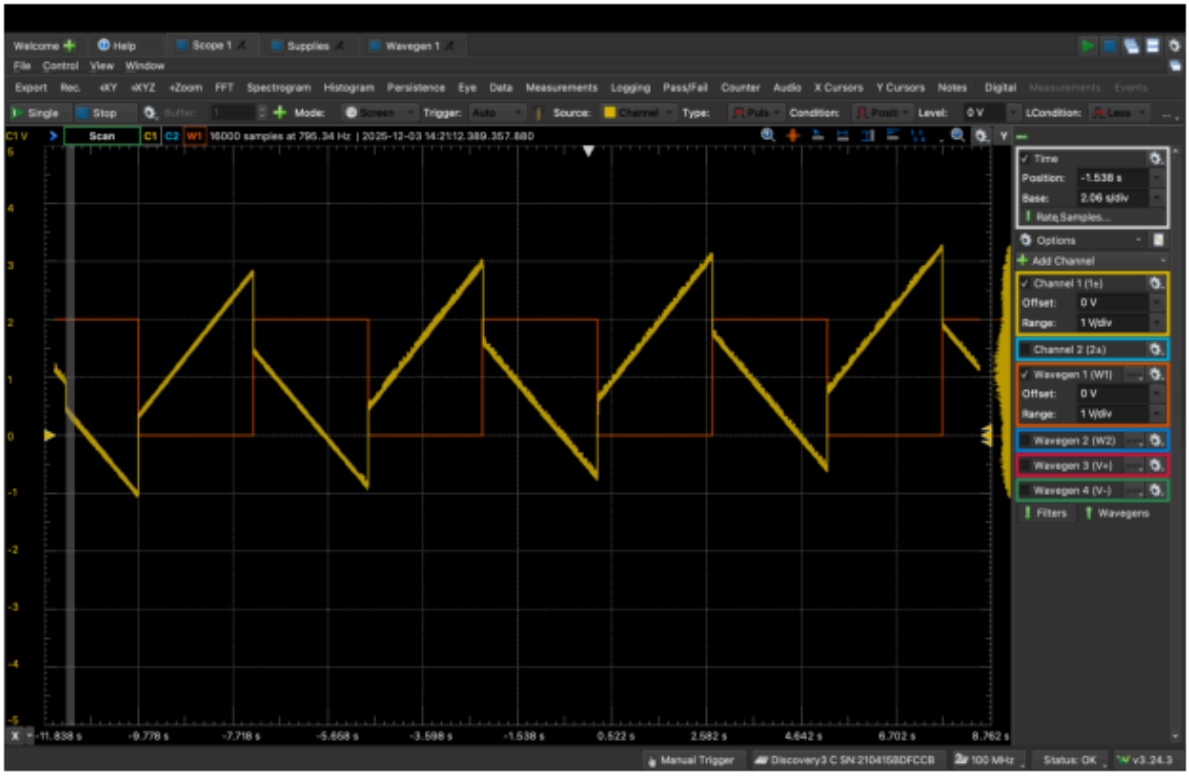


Figure 17: Error Signal Amplifier Testing

Motor Testing

I tried to test my motor by setting it up as seen in Figure 18, and then scoping the output of my CLK. I tried manually spinning the motor, and this produced the output in Figure 19.

In [63]:
```
InlineImage("MotorPhoto.jpg",width=8,height=8,caption="Figure 18: Motor Set
```

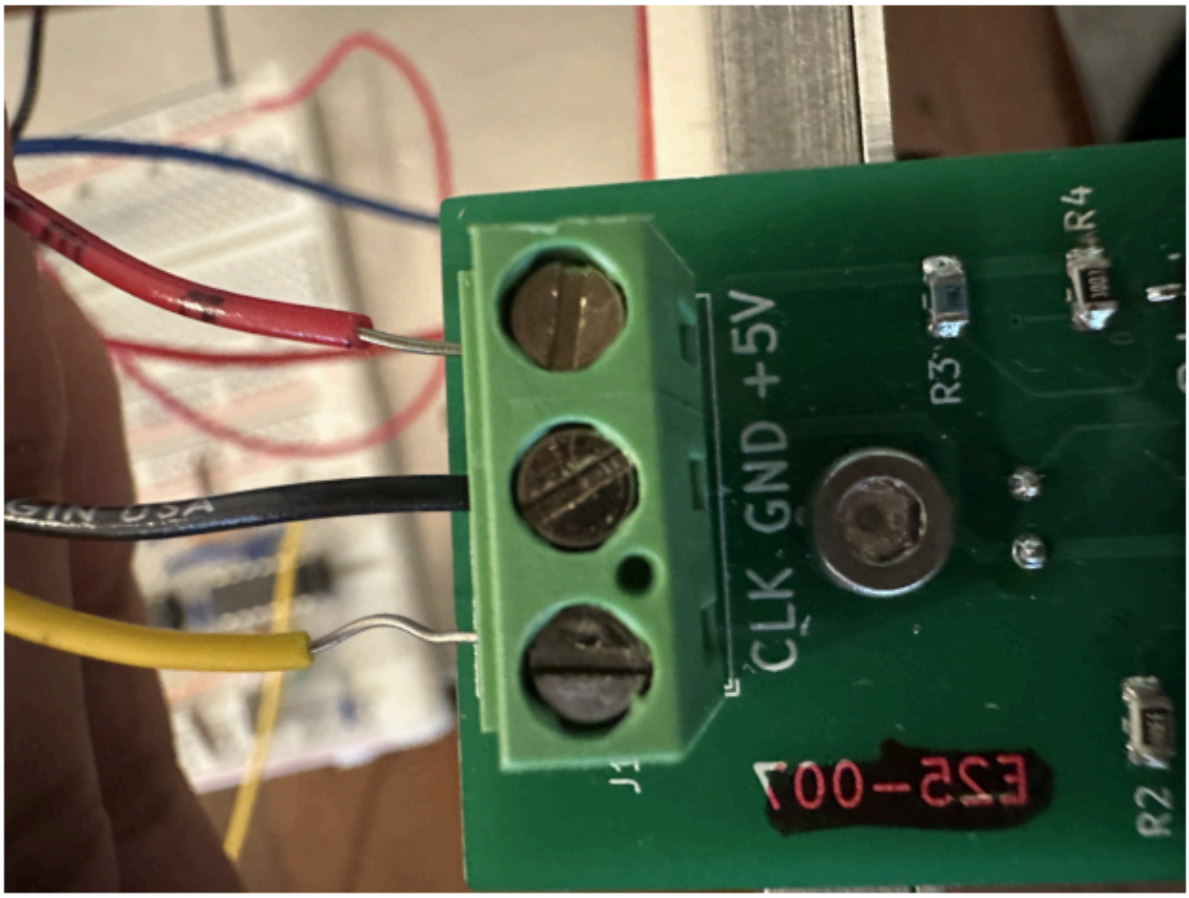


Figure 18: Motor Set Up

In [64]:
```
InlineImage("clockTroublehooting.png",width=8,height=8,caption="Figure 19: C
```

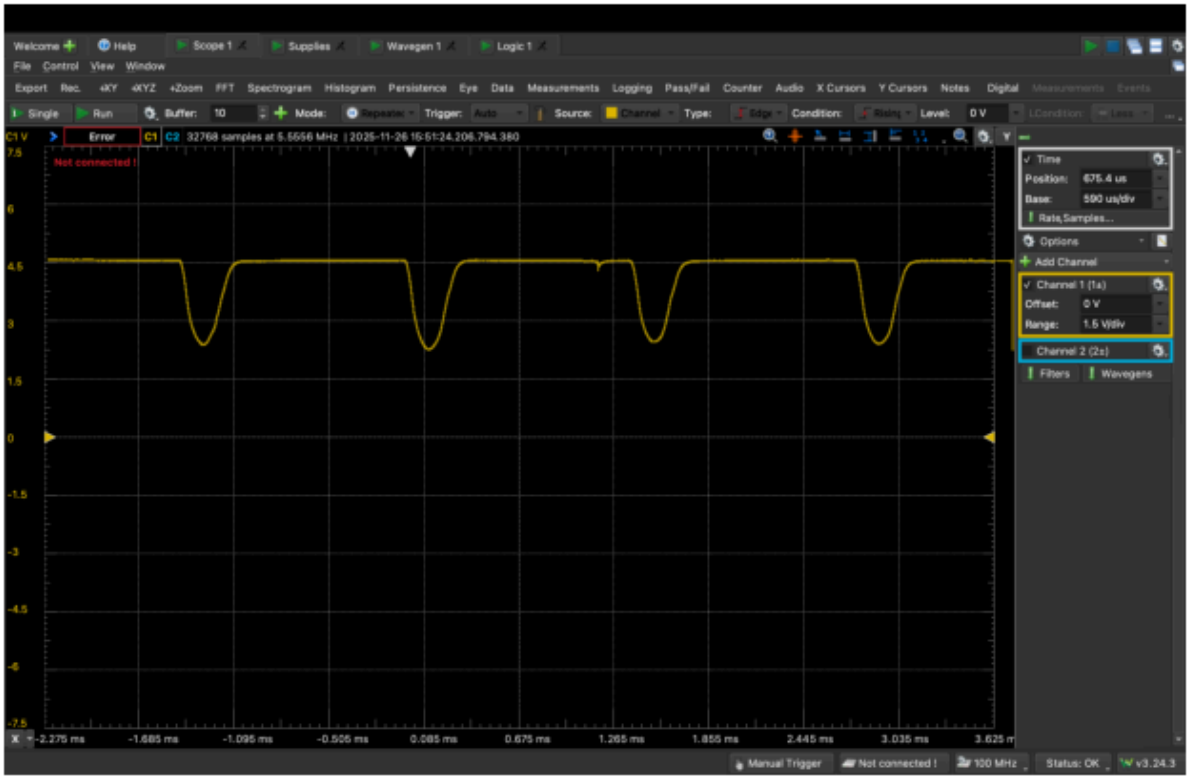


Figure 19: CLK output

When trouble shooting my motor, I noticed that the output of the schmitt-trigger inverter (and therefore my input into my counter) was just 0. I traced this back, and I realized that my clock pulse didn't drop the voltage low enough to trigger the inverter due to the hysterisis properties.

This led me to the introduction of the comparator in my circuit. The idea was compare the clock pulse to a set voltage, in order to trigger a set 5 V high low that always triggers. The comparator pinout is seen in Appendix F, and a circuit diagram can be seen in Figure 20. I knew I wanted to compare to the lowest voltage possible that still consistently triggered, and after examining the behaviour of the clock pulse for a bit I decided on around 2.6 V. This ensured that I would trigger the inverter, and also limit any noise from affecting my signal. I built this in Figure 21.

TS - This initially did not work, but was instead producing a 0 output still. The solution was to use a pull up resistor, pulling the output of the comparator to high. This would then only drop to low when the CLK was below 2.6 V.

In [65]:
```
InlineImage("ComparatorCircuit.jpeg",width=8,height=8,caption="Figure 20: Cc
```

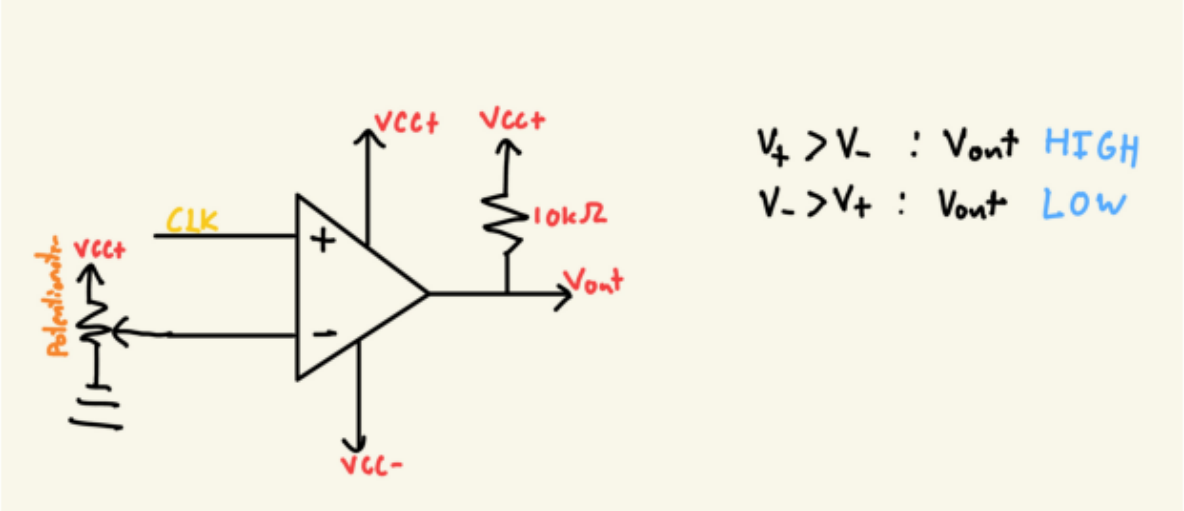


Figure 20: Comparator Circuit

In [66]:
```
InlineImage("ComparatorPhoto.jpg",width=8,height=8,caption="Figure 21: Compa
```

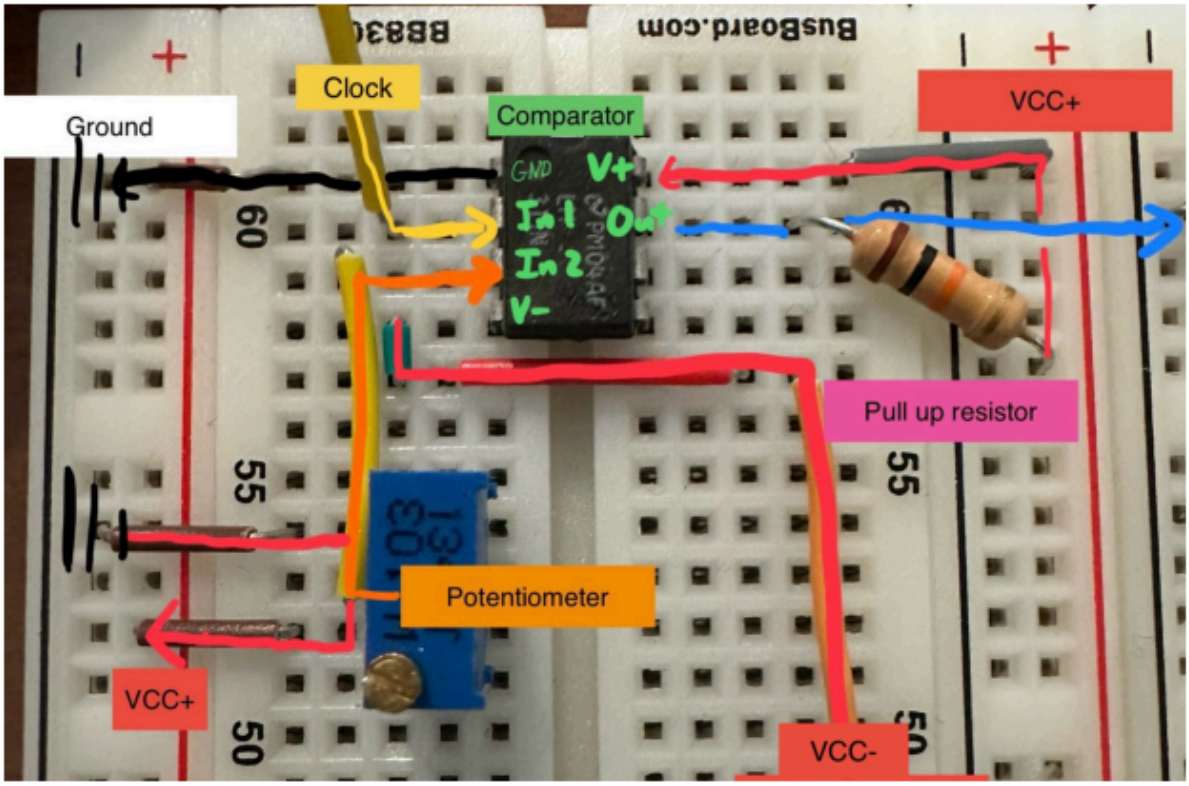


Figure 21: Comparator Photo

The next step was to wire up the motor and test that the counter and latch was updating with the motor speed. I set up the 8 DIO pins on the latch, and wired up the motor so that it was going directly through the power 3W resistor from 5 V to -5V. As the motor spun, the bus updated and the relative rotation speed of the wheel was updated on the latch.

TS - Initially this wasn't working, but I was advised that I needed to plug my AD3 into the wall to make sure I could provide the current that I needed for the motor.

BJT

I then finally set up my my BJT and integrated the whole circuit as seen in Figure 22 and 23.

In [67]:
```
InlineImage("BJT+DiodeCircuit.png",width=8,height=8,caption="Figure 22: BJT+
```

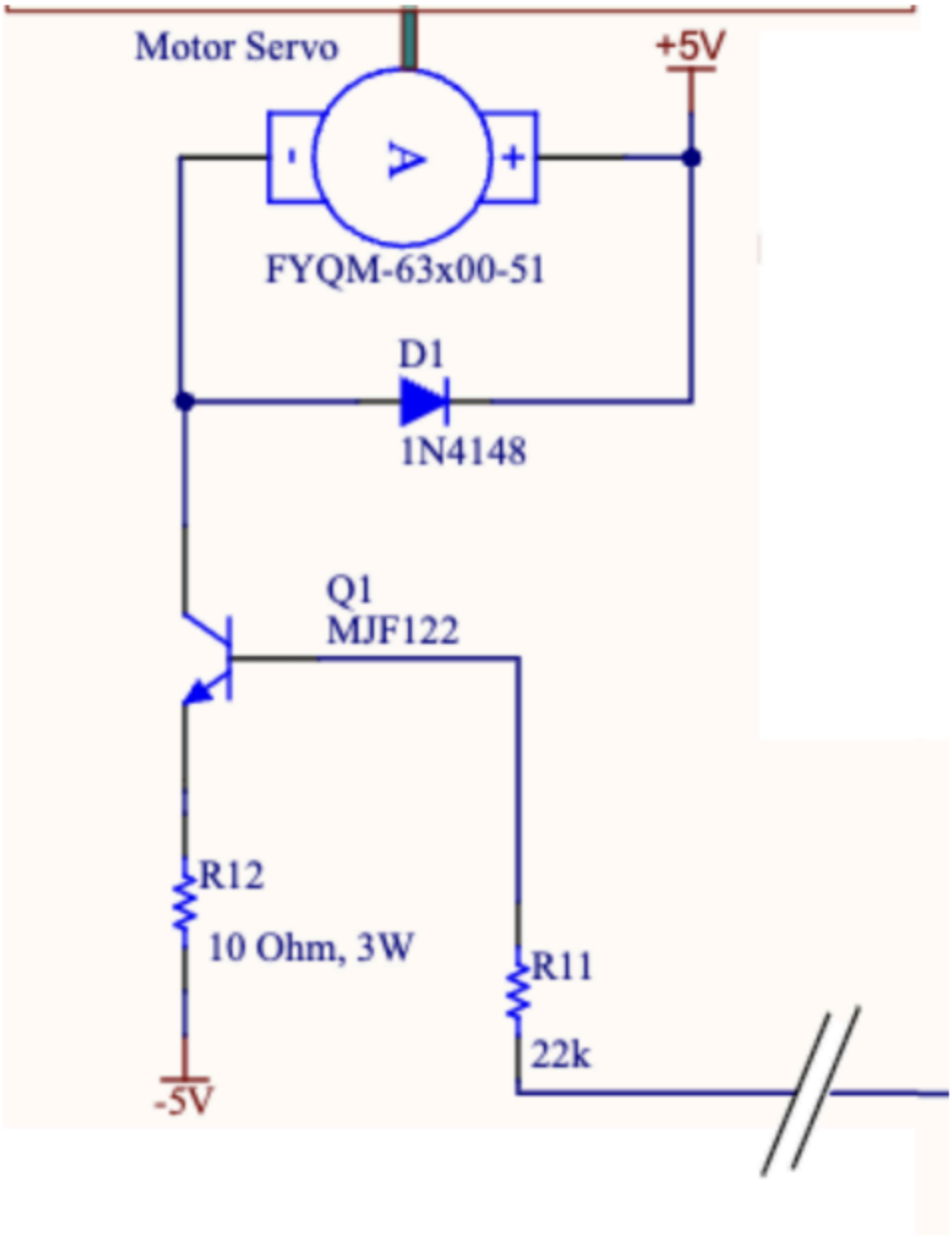


Figure 22: BJT+ Motor Circuit

In [68]:
```
InlineImage("BJT+Diode.jpg",width=8,height=8,caption="Figure 23: BJT+ Motor
```

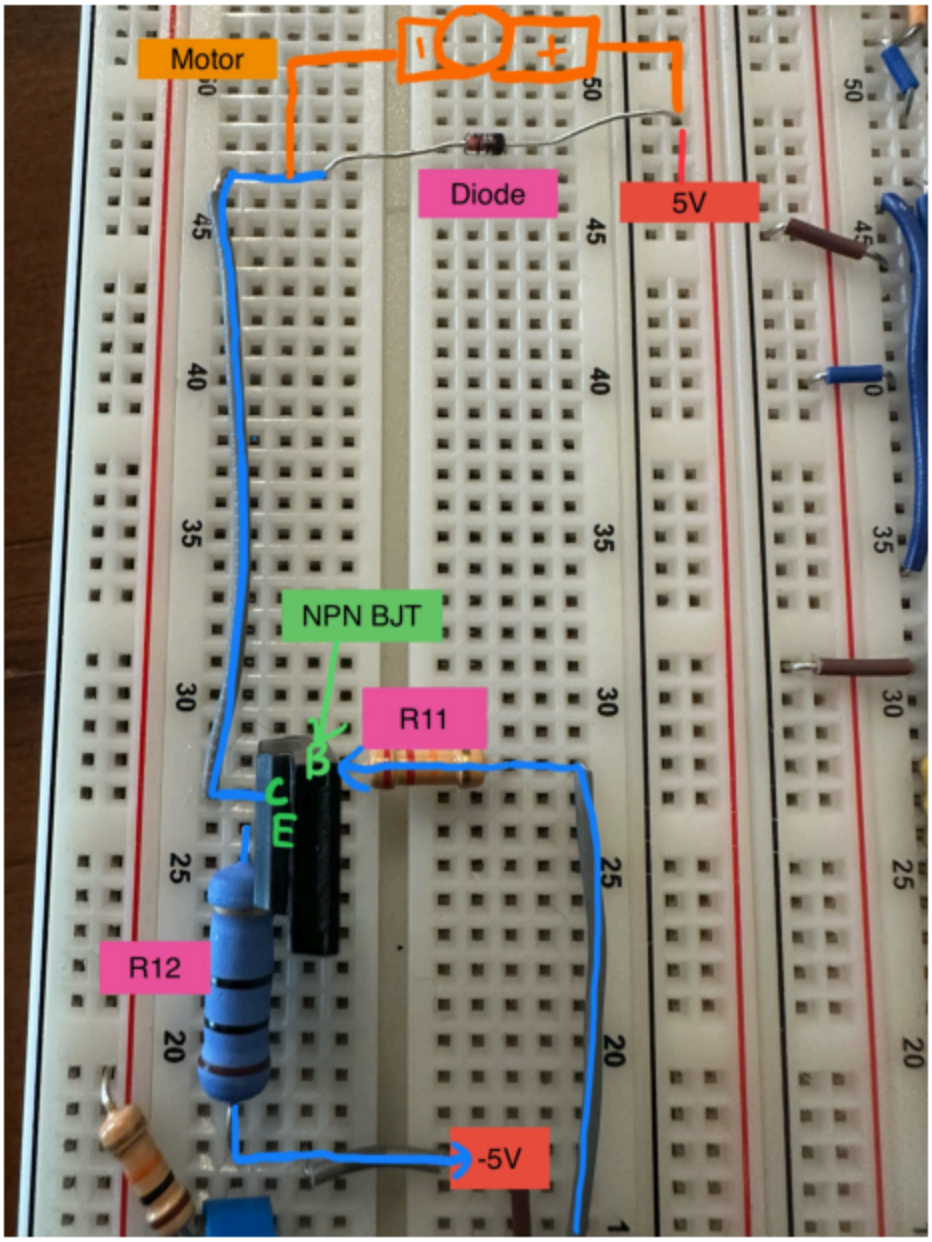


Figure 23: BJT+ Motor Photo

Troubleshooting + Conclusion

After wiring everything up my circuit still unfortunately didn't work, but I wasn't really sure what was wrong. I individually tested each component of the circuit over and over again, following the same steps I used to test when I initially set it up. I did the following steps in isolation (without the motor). I scoped my latch pulse, as well as my reset and verified those were intact. I then checked my counter by itself and it worked fine. I then used the 50 Hz wave into my counter and verified that my D-Latch was still latching. Then, I tested out my error signal wave again. At this point, all my components worked individually so I wasn't really sure what the problem is. The only part I didn't really test with was the motor. The next step I took was to test my circuit with Lauren's motor. To

my suprise, it ran fine and adjusted properly as I changed the potentiometer. I then signed out a new motor and my circuit worked! This new motor also didn't need the comparator, so I took it out for my final circuit.

To confirm the operation, I used my 8 DIO pins in a bus on the output of the latch, as well as a scope on both the set voltage and the DAC output. Picures of the output are seen in Figure 24 and 25 below. When I changed my Vset, the output of the DAC would follow it and match it. The DIO pins on the latch would also increase proportionally with the change, as I increased the Vset, the number being displayed also increased. This was up to a maximum of 110, at which point the circuit would kind of break and my output wouldn't go back down even if I brought my Vset back down.

In [69]:
```
InlineImage("MotorSpeed.png",width=8,height=8,caption="Figure 24: Motor Spee
```

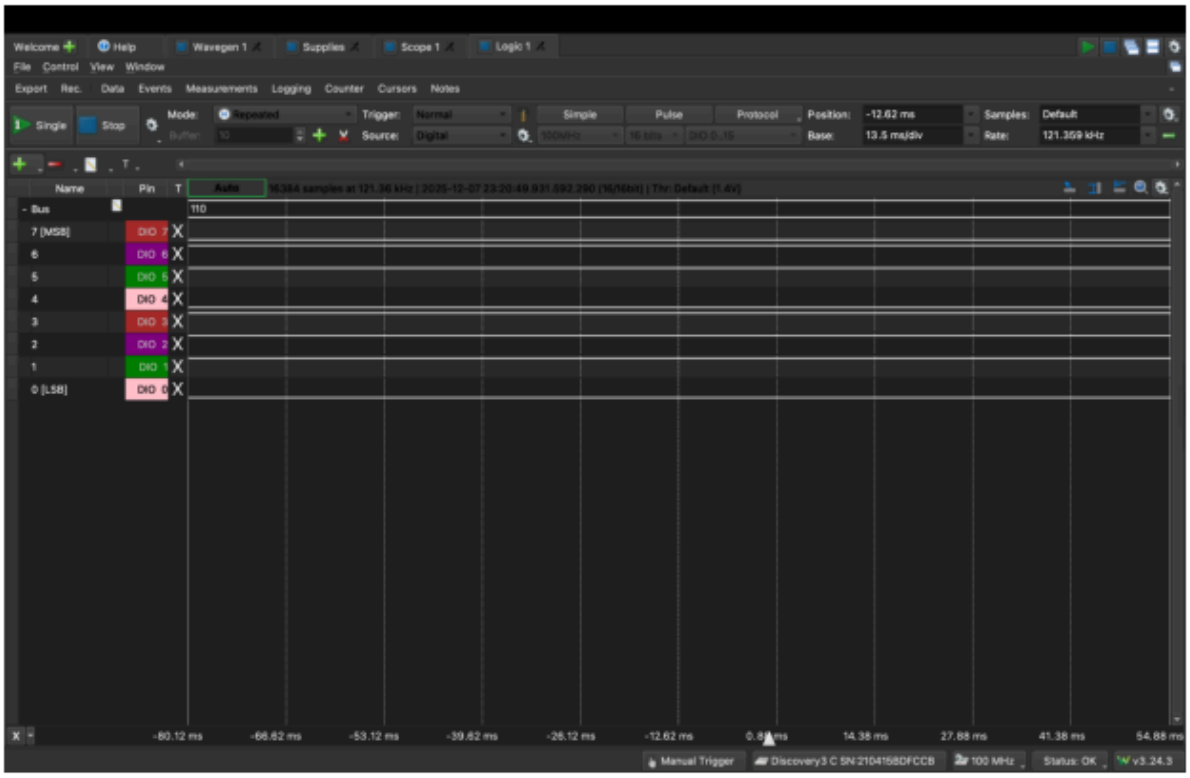


Figure 24: Motor Speed

In [70]:
```
InlineImage("OutputScope.png",width=8,height=8,caption="Figure 25: DAC + Vse
```

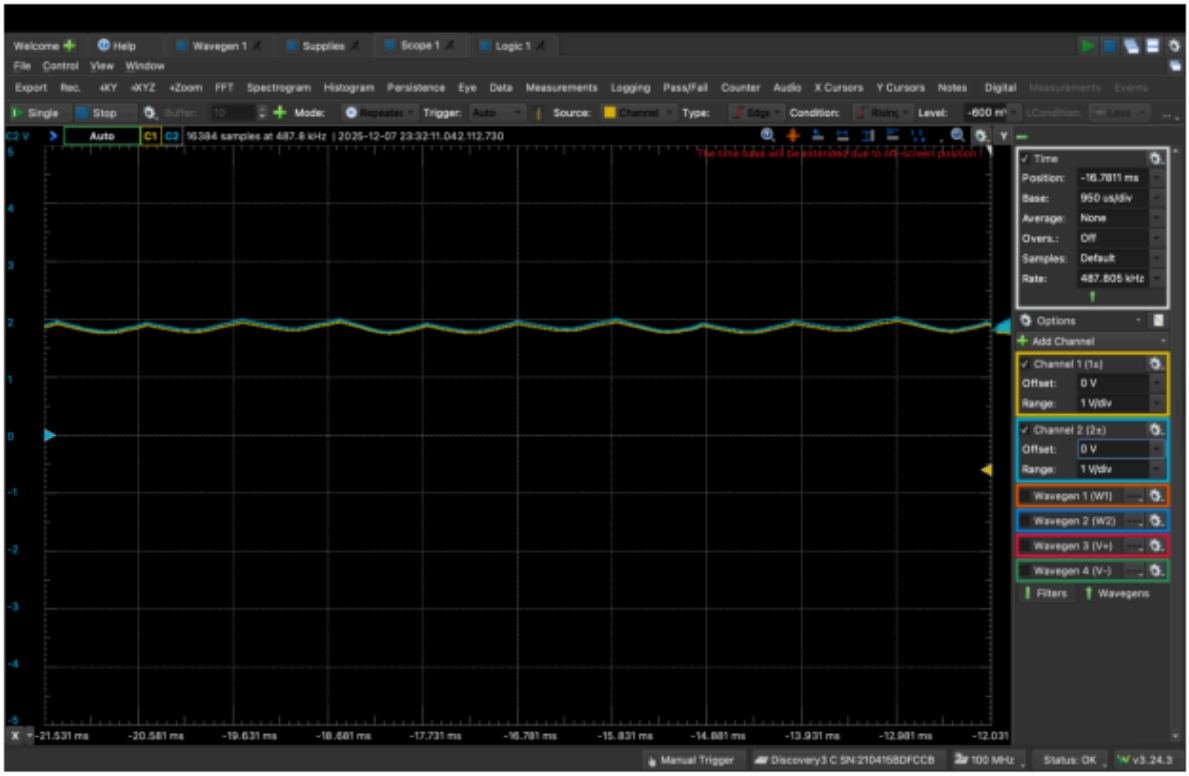
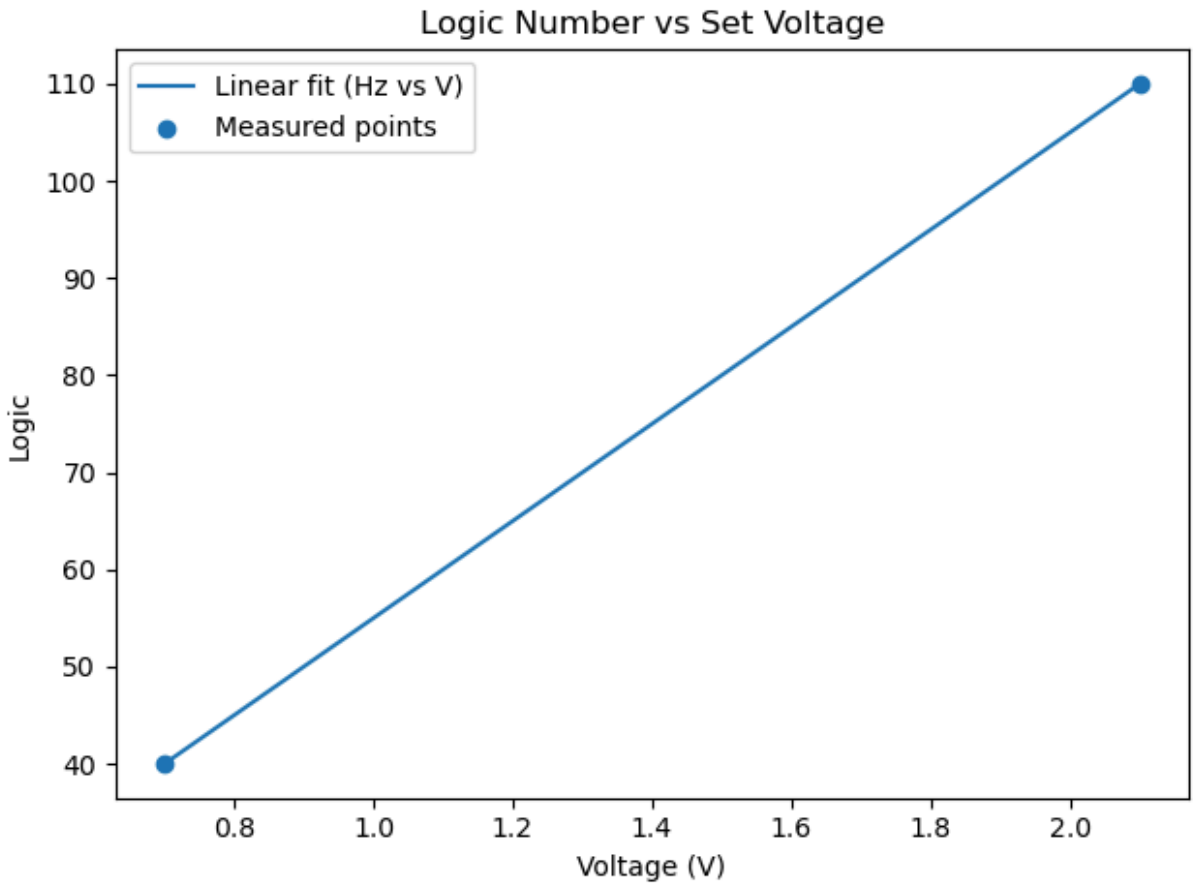


Figure 25: DAC + Vset

Speed and RPM

It was seen above that the maximum number was around 110. To change this to RPM, we would need to use the counter value multiplied by 30. This is because the counter is counting how many clock pulses there are at a frequency of 5 Hz. This would be every 0.2 seconds. Since there are 10 pulses per a rotation of the disk, it would become $ Motor speed = Counter Output * 5 * 60 / 10$. This means that my motor had a maximum speed of 3300. A table converting the two as well as a plot can be seen below.

In [76]:
```
import matplotlib.pyplot as plt
import matplotlib.image as mplimg


V_line = np.linspace(v_min, v_max, 200)
F_line = m * V_line + b

plt.figure()
plt.plot(V_line, F_line, label="Linear fit (Hz vs V)")
plt.scatter([v_min, v_max], [f_min, f_max], label="Measured points")
plt.title("Logic Number vs Set Voltage")
plt.xlabel("Voltage (V)")
plt.ylabel("Logic")
plt.legend()
plt.tight_layout()
plt.show()
```

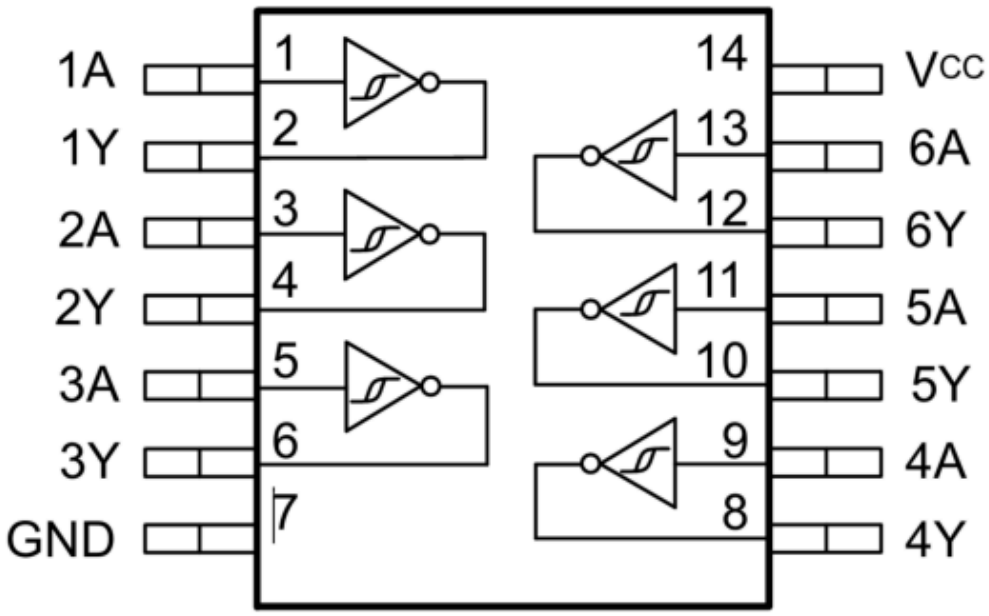


In [77]:
```
v_min, f_min = 0.7, 40
v_max, f_max = 2.1, 110
rpm_factor = 30


m = (f_max - f_min) / (v_max - v_min)
b = f_min - m * v_min


def linspace(a, b, n):
    if n == 1:
        return [a]
    step = (b - a) / (n - 1)
    return [a + i*step for i in range(n)]


V = [round(x, 3) for x in linspace(v_min, v_max, 8)]
F = [round(m*x + b, 3) for x in V]
RPM = [round(rpm_factor * fx, 1) for fx in F]


print(f"{'Voltage_set_V':>14}  {'Logic_value_Hz':>14}  {'RPM':>8}")
for v, f, r in zip(V, F, RPM):
    print(f"{v:14.3f}  {f:14.3f}  {r:8.1f}")
```

Voltage_set_V	Logic_value_Hz	RPM
0.700	40.000	1200.0
0.900	50.000	1500.0
1.100	60.000	1800.0
1.300	70.000	2100.0
1.500	80.000	2400.0
1.700	90.000	2700.0
1.900	100.000	3000.0
2.100	110.000	3300.0

Appendix

In [73]:
```
InlineImage("Schmit-TriggerPinout.png",width=8,height=8,caption="Appendix A:
```

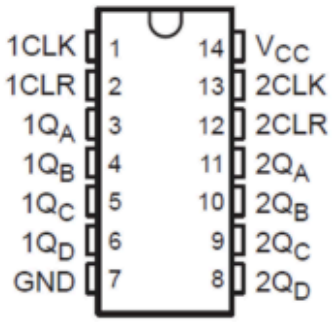


Functional pinout

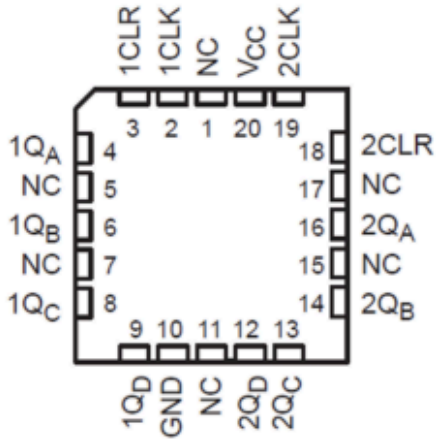
Appendix A: Schmit-Trigger Pinout

In [74]:
```
InlineImage("CounterPinout.png",width=8,height=8,caption="Appendix B: Counte
```

3 Pin Configuration and Functions



SN54HC393 J or W Package, 14-Pin CDIP or CFP; SN74HC393 D, DB, DYY, N, NS, or PW Package; 14-Pin SOIC, SSOP, SOT-23, TVSOP, SOP, or TSSOP (Top View)



A. NC - No internal connection

SN54HC393 FK Package, 20-Pin LCCC (Top View)

Table 3-1. Pin Functions

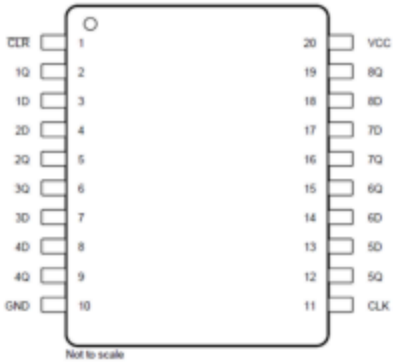
PIN		TYPE[1]	DESCRIPTION
NAME	NO.		
1CLK	1	I	Counter 1 Clock Input
1CLR	2	I	Counter 1 Clear Input
$1Q_A$	3	O	Counter 1 A Output
$1Q_B$	4	O	Counter 1 B Output
$1Q_C$	5	O	Counter 1 B Output
$1Q_D$	6	O	Counter 1 B Output
GND	7	G	Ground
$2Q_D$	8	O	Counter 2 D Output
$2Q_C$	9	O	Counter 2 C Output
$2Q_B$	10	O	Counter 2 B Output
$2Q_A$	11	O	Counter 2 A Output
2CLR	12	I	Counter 2 Clear Input
2CLK	13	I	Counter 2 Clock Input
V_{CC}	14	P	V_{CC}

1. I = Input, O = Output, I/O = Input or Output, G = Ground, P = Power.

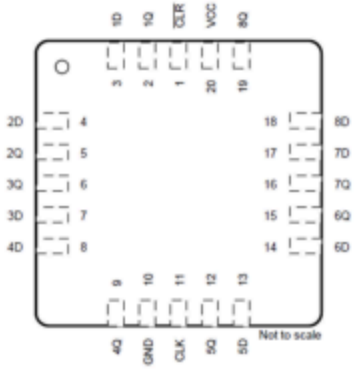
Appendix B: Counter Pinout

In [8]:
```
InlineImage("DLatchPinout.png",width=8,height=8,caption="Apendix C: D-Latch
```

5 Pin Configuration and Functions



J, W, DB, DW N, NS, or PW Package,
20-Pin CDIP, CFP, SSOP, SOIC, SO, PDIP, or TSSOP
(Top View)



FK Package,
20-Pin LCCC
(Top View)

Table 5-1. Pin Functions

PIN		TYPE[1]	DESCRIPTION
NO.	NAME		
1	CLR	I	Active low clear input
2	1Q	O	Output 1
3	1D	I	Input 1
4	2D	I	Input 2
5	2Q	O	Output 2
6	3Q	O	Output 3
7	3D	I	Input 3
8	4D	I	Input 4
9	4Q	O	Output 4
10	GND	—	Ground
11	CLK	I	Clock input
12	5Q	O	Output 5
13	5D	I	Input 5
14	6D	I	Input 6
15	6Q	O	Output 6
16	7Q	O	Output 7
17	7D	I	Input 7
18	8D	I	Input 8
19	8Q	O	Output 8
20	V_{CC}	—	Power

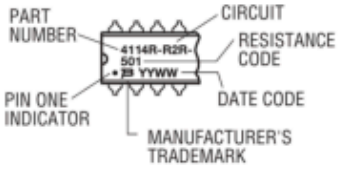
(1) Signal Types: I = Input, O = Output, I/O = Input or Output.

Apendix C: D-Latch Pinout

In [9]:
```
InlineImage("DACPinout.png",width=8,height=8,caption="Apendix D: DAC Pinout"
```

Typical Part Marking

Represents total content. Layout may vary.

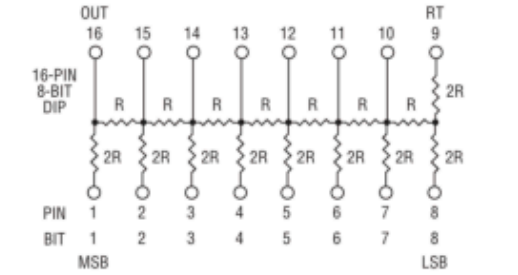


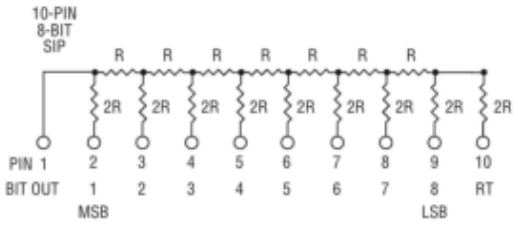
Product Dimensions

See applicable model data sheet for complete product dimensions.

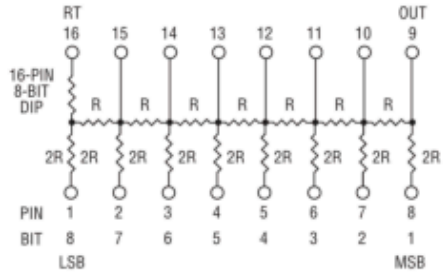
Resistor Schematic

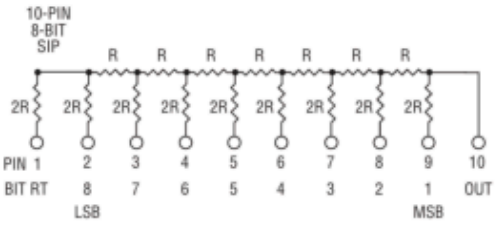
R2R Circuit





121 Circuit





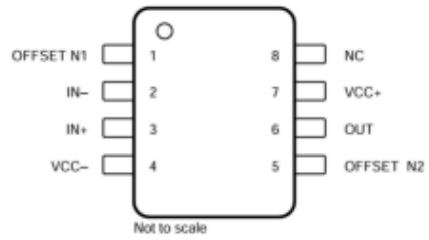
Power Derating Curve

Apendix D: DAC Pinout

In [10]:
```
InlineImage("op-ampPinout.png",width=8,height=8,caption="Apendix E: Op-Amp P
```

5 Pin Configurations and Functions

uA741C D, P, or PS Package
8-Pin SOIC, PDIP, SO
Top View



Not to scale

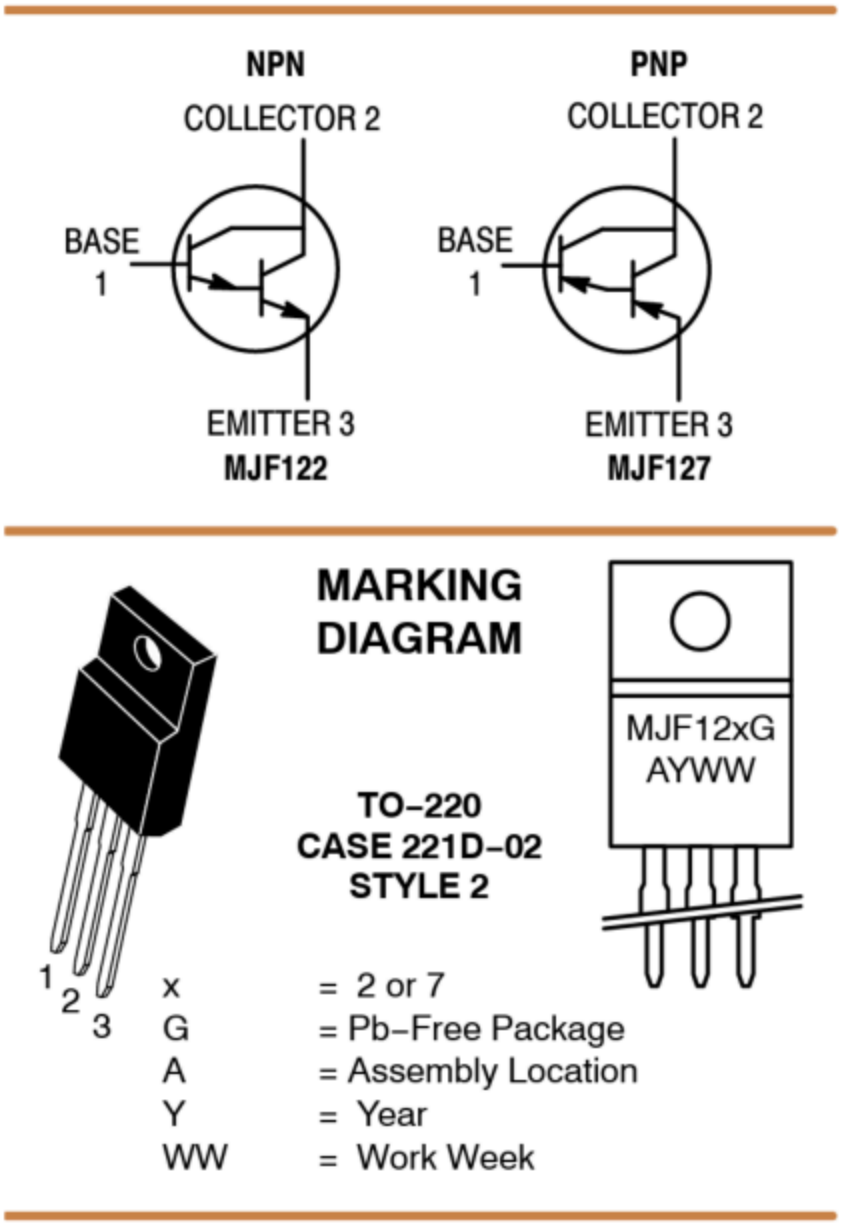
NC- no internal connection

Pin Functions

PIN		I/O	DESCRIPTION
NAME	NO.		
IN+	3	I	Noninverting input
IN–	2	I	Inverting input
NC	8	—	No internal connection
OFFSET N1	1	I	External input offset voltage adjustment
OFFSET N2	5	I	External input offset voltage adjustment
OUT	6	O	Output
VCC+	7	—	Positive supply
VCC–	4	—	Negative supply

Apendix E: Op-Amp Pinout

In [12]:
```
InlineImage("BJTPinout.png",width=8,height=8,caption="Apendix F: BJT Pinout"
```



Apendix F: BJT Pinout

In [4]:
```
InlineImage("ComparatorPinout.png",width=8,height=8,caption="Apendix G: Comp
```

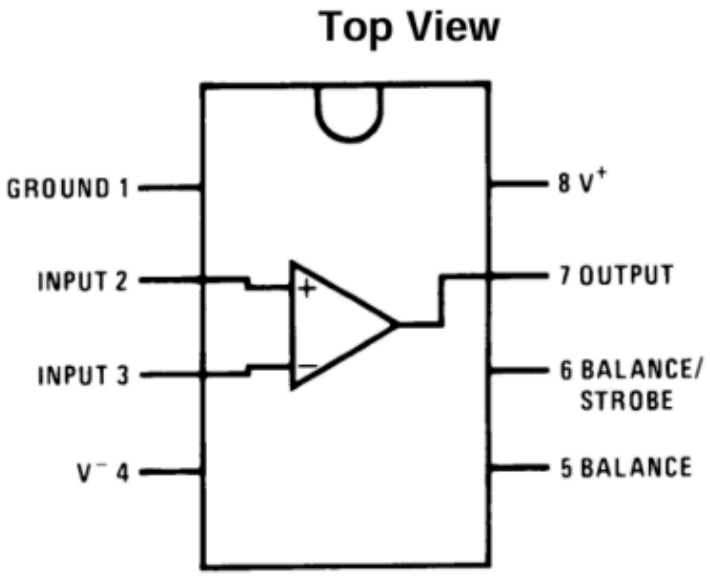


Figure 60. 8-Pin CDIP (See NAB Package)
8-Pin SOIC (See D Package)
8-Pin PDIP (See P Package)

Apendix F: Comparator Pinout

In [ ]: