The Final Project: Servo Motor

December 10, 2023

Name: Shao, Nora

Teammates at Table: [redacted]

Student Number: [redacted]

Lab Section: L7A

[94]:
```
# @title
%%capture
from IPython.display import Image
from google.colab import drive
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

drive.mount('/content/drive/')
path = 'drive/My Drive/UBC ENPH Y2/ENPH259/Lab7/'
file_name = 'The Final Project: Servo Motor.ipynb'

assert file_name in os.listdir(path)

def img_path(img_name):
  return path + img_name
```

1 Introduction

1.0.1 Wire colour code

- Orange: I/O from AD2
- Blue: internal connections between pins of in an IC or other circuit components OR DIO pins from AD2
- Purple: connections between different circuit components
- Red: power (5V)
- Black: GND
- Blue: $V_{CC}-$

1.0.2 Experiment Overview

In this lab, we build and a test a breadboard-based circuit that lets us control and read the speed of a servo motor via a potentiometer and the Analog Discovery 2 board. I'll be building the individual circuits clockwise on the two breadboards.

For organizational purposes, I attached the two breadboards together and connected all the power and gnd rails (separately), so just one power and gnd input from the AD2 breakout board would power all the power rails and ground the grounding rails.

[95]:
```
Image(filename = img_path("L7A_complete schematic.png"))
```

[95]:

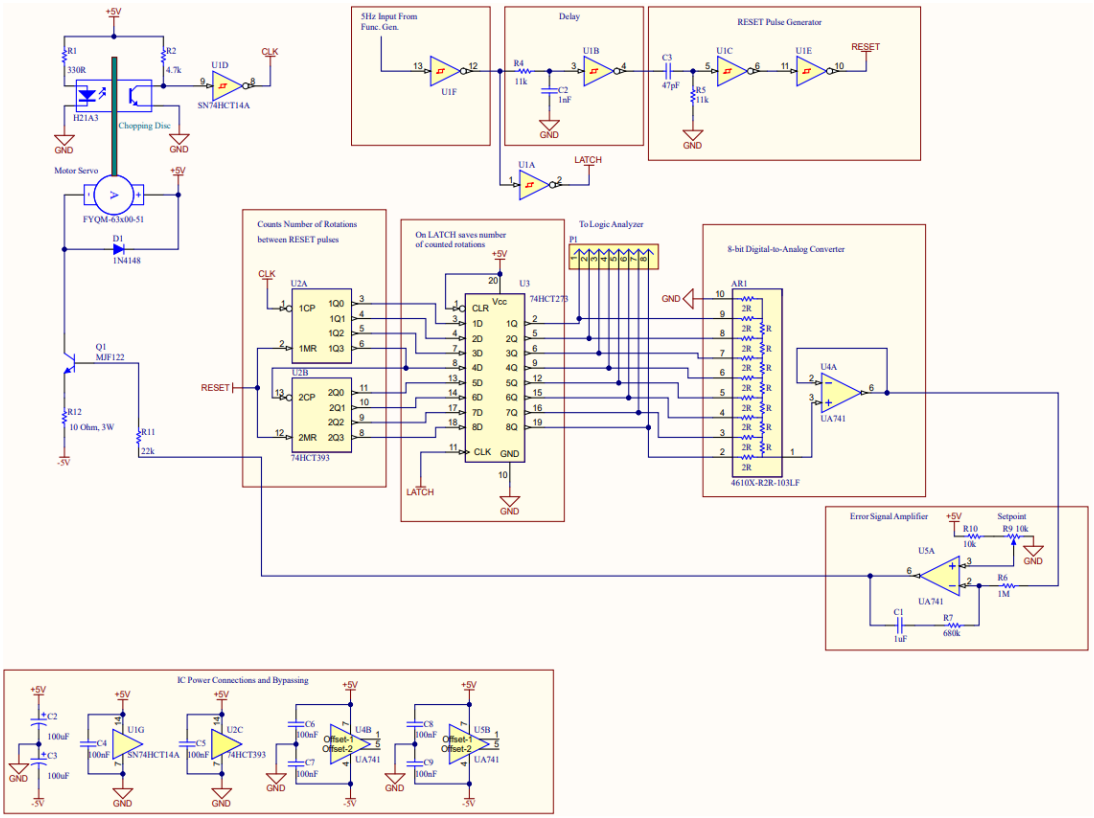

Figure 1: Functional diagram of entire circuit controlling servo motor

2 Experiment

2.1 *LATCH* and *RESET* generator

The latch takes a signal from a 5 Hz square wave input going from 0 to 5 V, and a reset pulse is generated with a Schmitt-trigger inverter by the series of capacitors and resistors that first create a delay from the input signal, then generates the brief reset pulse.

The reset pulse is what tells the counter IC to clear and start counting from 0 again, and the latch pulse tells the latch to read and store the values from the counter.

The delay is necessary because it ensures the pulse sent that clears the counter always occurs briefly (12 s) after the latch pulse, which tells the latch to pull the current value from the counter and stores it. The delay is short enough that there is almost no change the counter will increment in the time between the latch storing its previous value and the clearing of the latch, so there won't be a small discrepancy between the counter and latch values.

2.1.1 Delay Circuit

Procedure

- I'll use the 74HC14 Schmitt-Trigger Inverters as our inverters for the latch and reset generator and follow this datasheet: https://www.mouser.com/datasheet/2/308/74HC14.REV1-34947.pdf
- I built the delay and reset generator circuits around one inverter IC, like in Figure 3 and with the setup in Figure 4
- I test the operation of the delay by inputting the square wave into the input of the U1F inverter and reading the output of the U1B inverter.
  - I used a BNC cable connected to the Wavegen 1 Output of the AD2 breakout board
  - I used BNC cables connected to the Scope 1+ and Scope 2+ inputs on the breakout board to read the output of the U1B inverter (delay circuit) and the input, respectively.

Calculations:

Note: See below in the "Understandings" section where we got the V_{T-} and V_{T+} values.

In a series circuit, we have the time constant

$$\tau = RC,$$

where R and C are respectively the resistance and capacitance of the circuit. Here, we get

$$\tau = R4 \times C2,$$

or

$$\tau = 11k\Omega \times 1nF,$$

which evaluates to $\tau = 1.1 \times 10^{-5}s$.

The signal from the delay circuit (after U1B) goes HI when the input signal to U1B drops below V_{T-}, 1.65 V.

$$V(t) = V_0 e^{-t/RC},$$

so

$$1.65V = 5Ve^{-t/\tau},$$

and

$$t = -\tau \ln(\frac{1.65}{5}),$$

and we plug in our values to get

$$t = \tau \ln(\frac{1.65}{5}).$$

We evaluate to get

$$t = 12.2\mu s.$$

Thus, we expect our delay to be around 12.2 s, since there is some variation in what our actual V_{T+} and V_{T-} values are.

Troubleshooting

#TS1: I set my input as a square wave from 0 to 5 V with am amplitude and offset of 2.5 V, but when I read the input with the scope, I saw that it was going from 0 to 2.5 V instead of 0 to 5 V. After inspecting my circuit with the TA, I found that I had wired the input of U1F to the output, which is bad, since the output state of an inverter should always be different from the input, yet I had effectively forced them to be the same state.

#TS2: I first measured a delay of ~2 s, which didn't fit well at all with my calculations, so I reinspected my circuit and found that I didn't ground the IC. After I grounded the IC, my delay was ~12 s, which made sense as it was (very) close to my expected delay.

[96]:
```
Image(filename = img_path("L7A_delay and reset schematic.png"))
```

[96]:

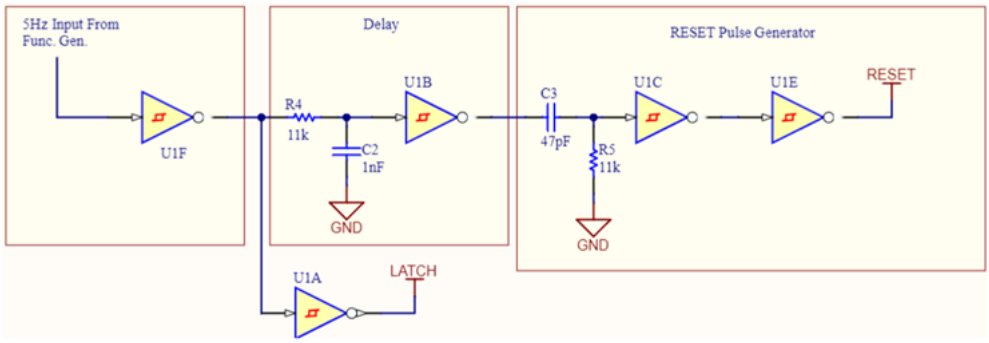

Figure 2: Function diagram of delay and reset pulse generating circuits

[97]:
```
Image(filename = img_path("L7A_inverter IC pinout.jpg"))
```

[97]:

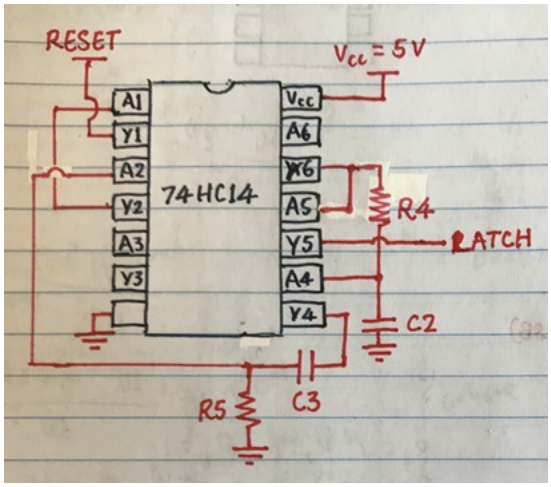

Figure 3: Pinout of inverter IC

[98]:
```
Image(filename = img_path("L7A_measuring delay.png"))
```

[98]:

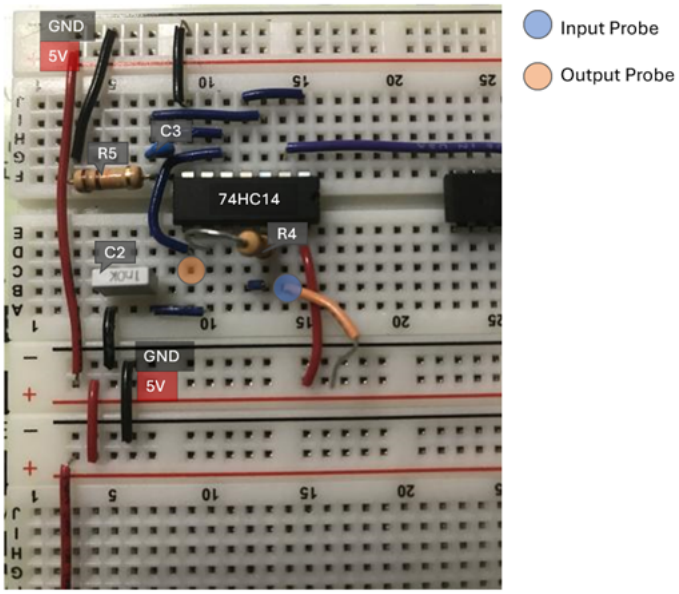

Figure 4: Setup of delay and reset circuit on breadboard for measuring delay generated by delay circuit

Results

I confirmed the delay circuit worked with a delay of ~11.65 s, which we see in Figure 9, which is very close to our expected delay.

Understandings

The delay circuit works by utilizing the Schmitt Trigger's inverting operations and the mini RC circuit made up of R4 and C2. The input signal must increase past a certain threshold, V_{T+}, and decrease past a different threshold, V_{T-} for the inverter to invert the signal to LO and HI, respectively. From the Schmitt-Trigger datasheet, we see in Figure 5 that there's a range of values for V_{T+}, between 2.6 and 3.5 V. Taking the middle of the range for a rough value for V_{T+}, we get 3.05 V. The datasheet tells us V_{T-} is between 1.3 and 2 V, which we take the middle of to get a rough value of 1.65 V.

Although the input signal changes abruptly, the capacitor slows down the growth/decay of the signal from the output of U1F such that there is a delay when the voltage at the input of U1B increases past or decreases below V_{T+} and V_{T-}. Looking closer at how the delay works, in Figure 7, we see what happens to the input signal into U1B when the input square wave goes HI, and it does take approximately 12 s for the signal to decay to the V_{T-} we're assuming. Now, we switch to look at U1B's output in Figure 8, and see that it indeed takes around 12 s to go HI, but the V_{T-} is actually around 1.75 V instead of the 1.65 V we initially assumed.

[99]:
```
Image(filename = img_path("L7A_schmitt trigger threshold voltages.png"))
```

[99]:

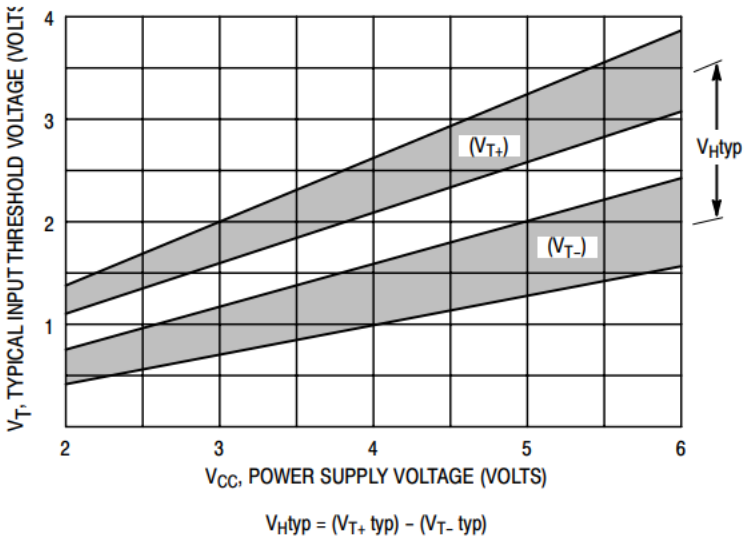

Figure 3. Typical Input Threshold, V_{T+}, V_{T-} versus Power Supply Voltage

Figure 5: Range of V_{T+} (negative switching voltage) and V_{T-} (positive switching voltage) values for Schmitt-Trigger Inverter depending on supply voltage

[100]:
```
Image(filename = img_path("L7A_zoomed out delay.png"))
```

[100]:

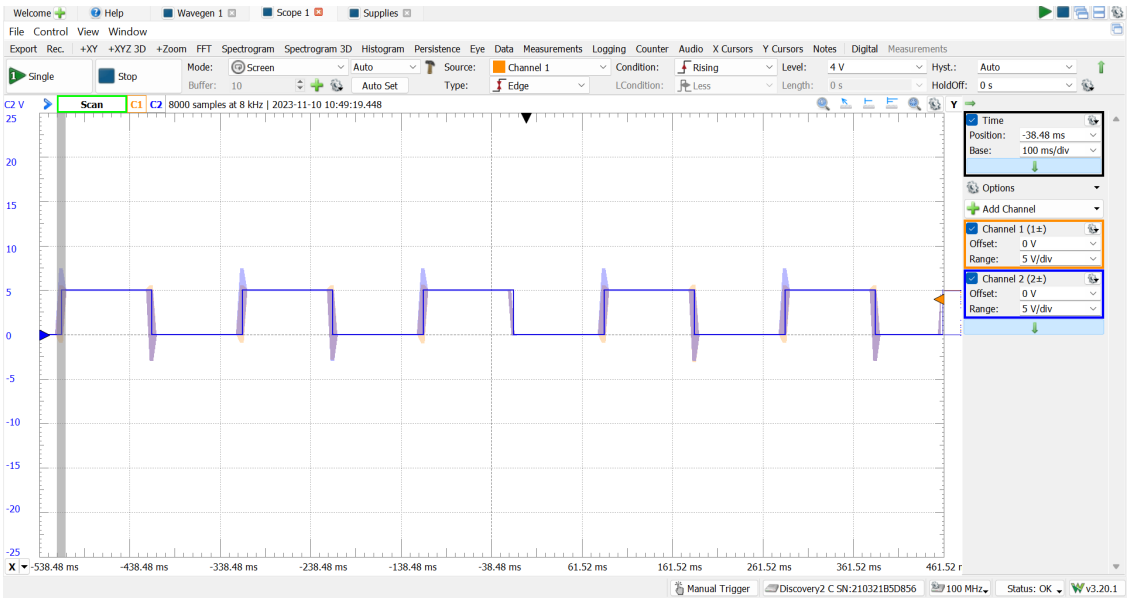

Figure 6: Zoomed out reading of delayed signal (output of U1B; CH1) compared with input signal (CH2)

[101]:
```
Image(filename = img_path("L7A_C2 discharging.png"))
```

[101]:

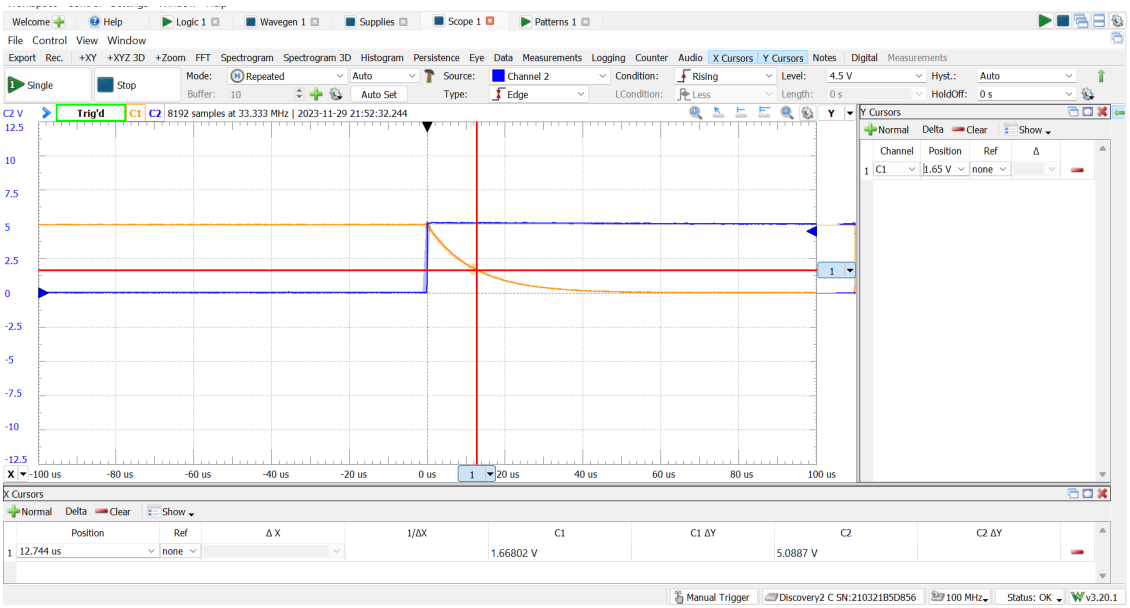

Figure 7: Input signal to IC U1F (CH2 blue) vs input signal to U1B/voltage over C2 (CH1 orange)

[102]:
```
Image(filename = img_path("L7A_C2 with delay.png"))
```

[102]:

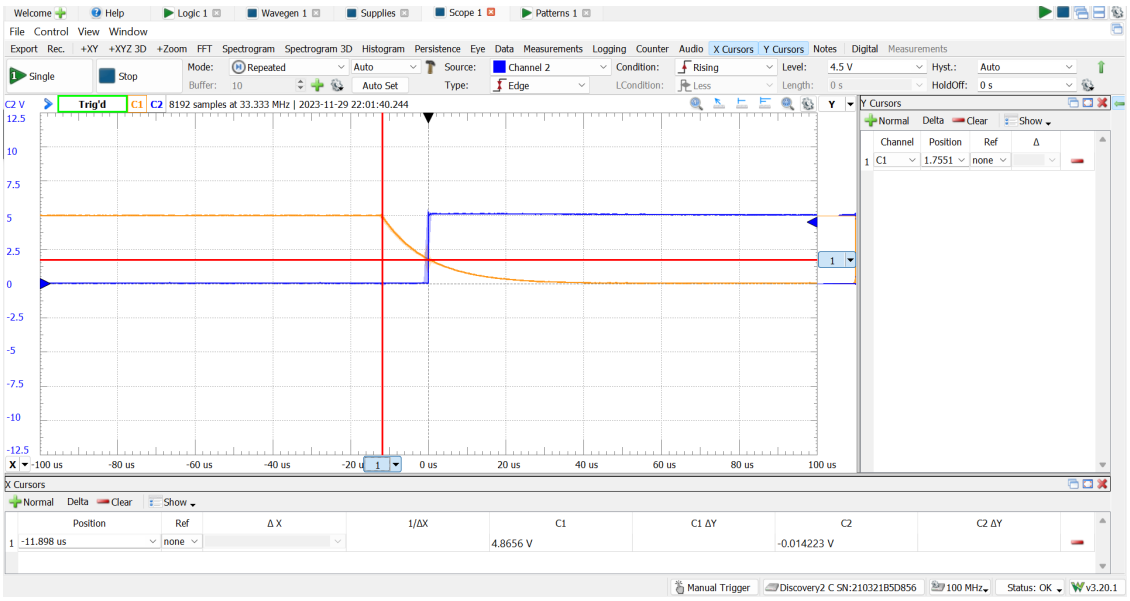

Figure 8: Input signal to inverter IC U1B (CH1 orange) vs output signal of U1B (CH2 blue)

[103]:
```
Image(filename = img_path("L7A_delay measurement.png"))
```

[103]:

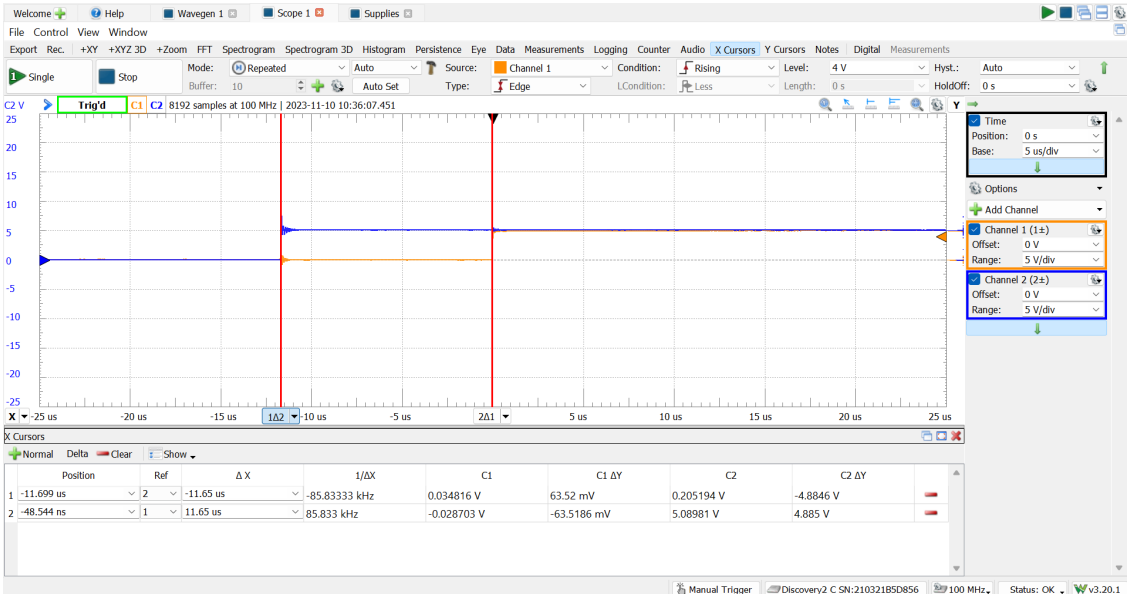

Figure 9: Measurement of delay from rising edge of output signal of delay circuit (CH1) and input signal (CH2)

2.1.2 RESET Pulse Circuit

Procedure

- I set up the RESET circuit on the breadboard around the IC as seen in Figure 10
- I kept the 5 Hz square wave oscillating from 0 to 5 V at the input of the IC
- First, I just scoped the counter reset signal with Scope 1+ and the input signal with Scope 1- to confirm the reset worked
- Then scoped the counter and latch reset signals with the AD2 Scope 1+ and Scope 1- inputs

Troubleshooting

#TS3: When I tried to measure the RESET signal output, I measured nothing even though I was inputting the 5 Hz square wave. I moved my output signal scope measurement to trace along the circuit, ensuring the signal was what I expected at every point. I found that past C3 in Figure 2, the signal flatlined at 0. Then I zoomed in significantly on the time scale and realized that the RESET signal pulsed very very briefly before going LO for the rest of the period. However, the delay from the rising edge of the input square wave to start of the pulse was around 2 s, when it should have been 12 s.

[104]:
```
Image(filename = img_path("L7A_measuring reset pulse.png"))
```

[104]:

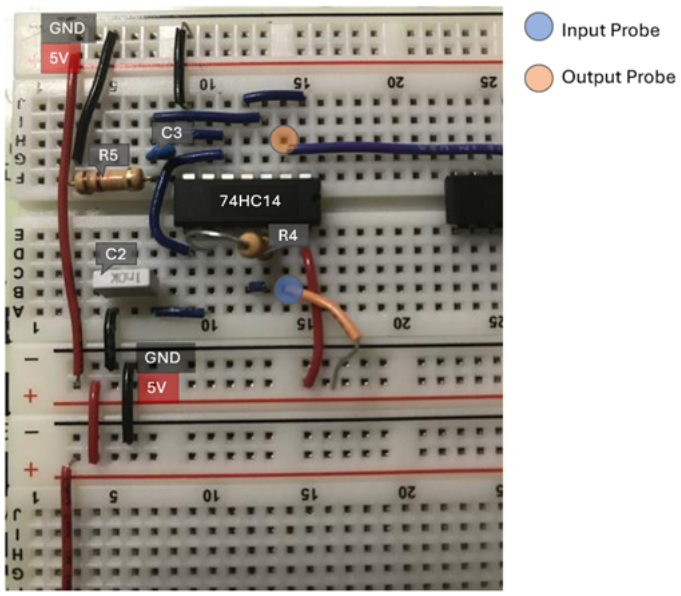

Figure 10: Setup of delay and reset circuit on breadboard for reading the counter reset pulse generated

Results

Just qualitatively, the reset pulse visible in Figure 11 and 12 sense as, well, it's a brief pulse. It has the same magnitude as the input signal 5 V, and is delayed by the delay circuit's ~12 s.

Figure 13 also confirms the counter and latch reset pulses occur together, which we want (with a delay for the counter reset), so there's no discrepancy between the two's values.

Understandings

The delay circuit results in the delay in the RESET pulse we see in Figure 11. This is further confirmed by the fact that the delay from the rising edge of the input signal to the rising edge of the RESET pulse is also around 12 s. In fact, it's approximately 11.72 s, which is very close to the 11.65 s delay measured directly from the delay circuit earlier.

Analyzing the behaviour of the circuit between U1B and U1C explains the shape of the reset pulse. When the input signal from the AD2 has been HI for a long time, the output of U1B is at 5 V, and the capacitor C3 will be an open circuit, so the input to U1C is LO (0 V), and its output is 5 V. Thus, RESET will be LO. Then, the input signal switches to LO, and at this instant, since there is still 5 V across C3, the input to U1C is -5 V, so the output stays the same (since the input is still below V_{T-}). However, the capacitor will discharge and eventually the input to U1C will go to GND (0 V, or LO). Then, the input signal switches to HI again. Now, briefly, the capacitor will act as a short-circuit, so the input to U1C will be above ~3 V, V_{T+}, and in that brief time before the capacitor charges and the input to U1C decays below V_{T+}, U1C will output LO, so RESET is HI, hence the brief pause in the HI state before the cycle restarts and RESET goes LO again.

[105]:
```
Image(filename = img_path("L7A_reset pulse.png"))
```

[105]:

Figure 11: Reset pulse generated by reset circuit (CH1) compared with input signal (CH2)

[106]:
```
Image(filename = img_path("L7A_reset pulse measurements.png"))
```

[106]:

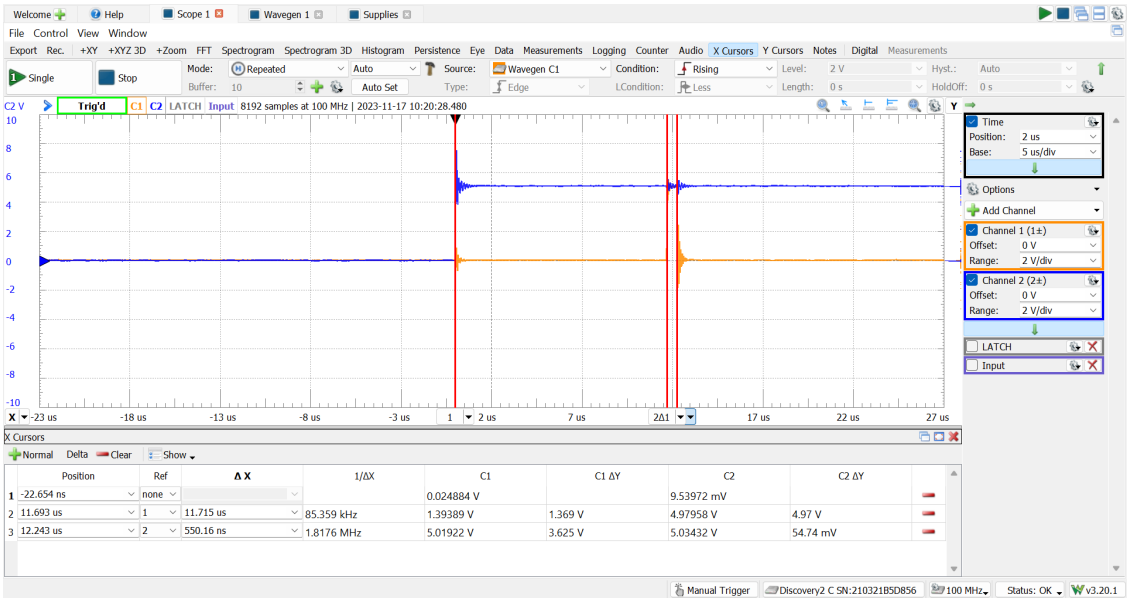

Figure 12: Measuring the delay between the rising edge of the reset pulse and the input signal and the width of the reset pulse

[107]:
```
Image(filename = img_path("L7A_reset and latch clock pulses.png"))
```

[107]:

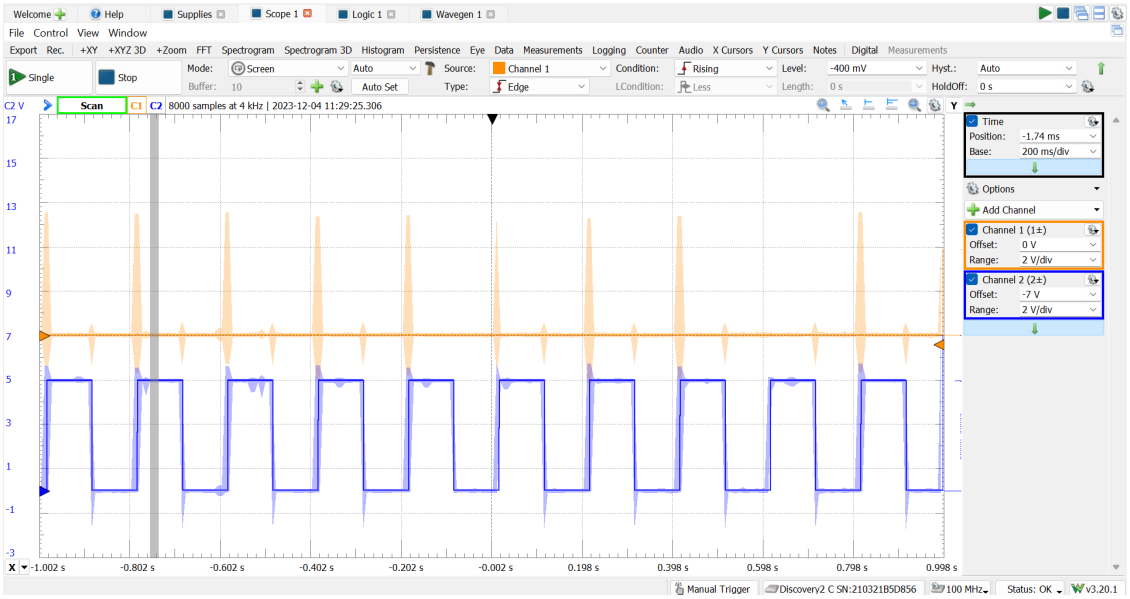

Figure 13: Counter reset signal (CH1 orange) vs. latch clock signal (CH2 blue)

2.2 Counter, D-latch, DAC

The counter keeps track of the current state of the motor, the latch stores this values, and the DAC converts this information to an analog voltage.

[108]:
```
Image(filename = img_path("L7A_counter latch dac.png"))
```

[108]:

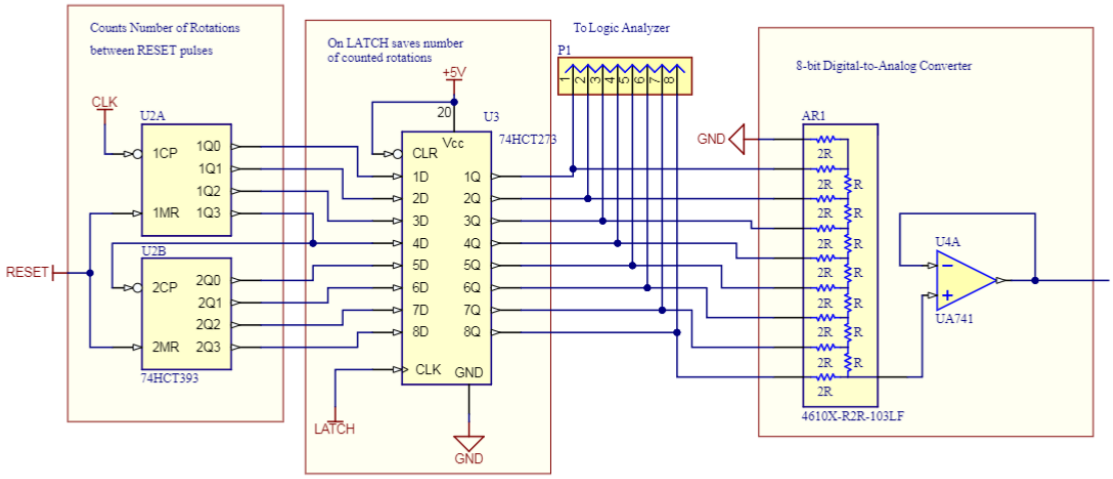

Figure 14: Function diagram of counter IC, latch IC, and DAC (consisting of resistor network and op-amp)

2.2.1 Counter

We link the two ÷ 16 counters to make a ÷ 256 counter, which counts how many times a hole on the motor disc passes through the housing in the period of $\frac{1}{f}$, where f is the frequency the reset and latch signals are sent at (we control f, but for this lab, it's 5 Hz). It increments every time its clock terminal receives (the falling edge of) a signal from the motor. It stores this value in binary form in its 8 bits.

Procedure

- I followed this datasheet for the dual 4-bit 74HCT393 counter IC: https://www.jameco.com/Jameco/Products/ProdDS/243205.pdf
- On WaveForm's Logic app, I set up DIO pins 0-7 in the BUS signal form, with 7 as the Most Signficant Bit (MSB) and DIO 0 as the Least Significant Bit (LSB) mapped as seen in Figure 17
- I connected these pins to the counter IC's bits like in Figure 16
- I used WaveForm's Patterns app to set up the clock and reset functions of the counter IC.
  - For the RESET pin, I connected it to DIO 8 and set it to be constantly low (so I won't get any unwanted resetting of the counter)

- For the CLK pin, I connected it to DIO 9 and set it as a 1 Hz clock, so I could watch the counter increase value at a decent pace.

- When determining the behaviour of the counter past its max value
  - I configured Logic to display the output from the counter in decimal form, so it was easier for me to understand the values
  - I adjusted the clock frequency to 10 Hz so it would reach the max value possible and over flow quicker

Troubleshooting

TS: My counter was acting weird, randomly jumping between decimal values. But, once I configured the value to display in decimal instead of binary form, I saw the digits of the binary value increasing from left to right, and not in a linear fashion, so I realized the endian of the diodes was reversed. I had set DIO 7 as the Least Signficant Bit and DIO 0 as the Most Signficant Bit when it should have been the other way around. I fixed this by reversing the endian of the diodes in the WaveForms Logic Bus window.

[109]:
```
Image(filename = img_path("L7A_counter IC pinout.jpg"))
```

[109]:

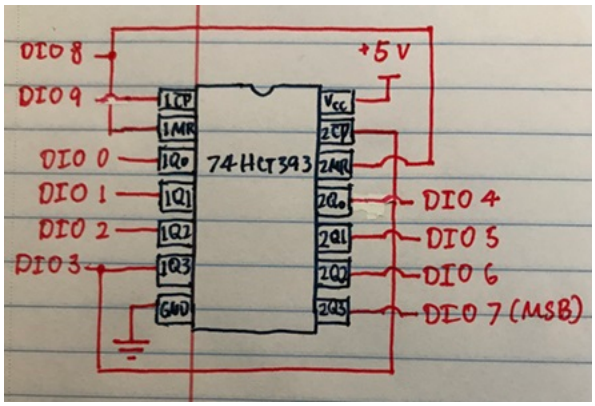

Figure 15: IC pinout diagram of counter for testing operation of counter

[110]:
```
Image(filename = img_path("L7A_counter pinout setup.png"))
```

[110]:

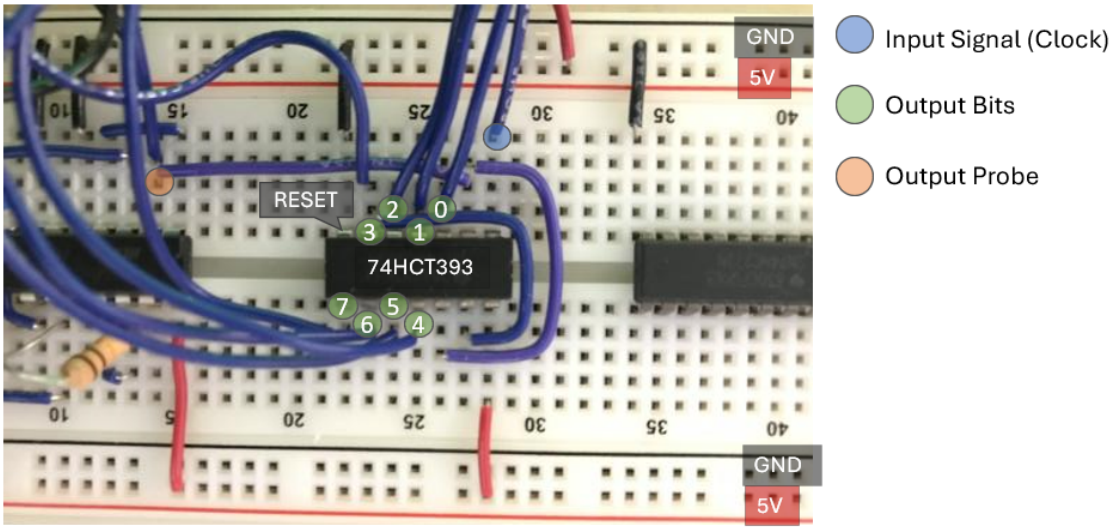

Figure 16: Setup of counter IC wired to test operation on breadboard

[111]:
```
Image(filename = img_path("L7A_counter logic setup.png"))
```

[111]:

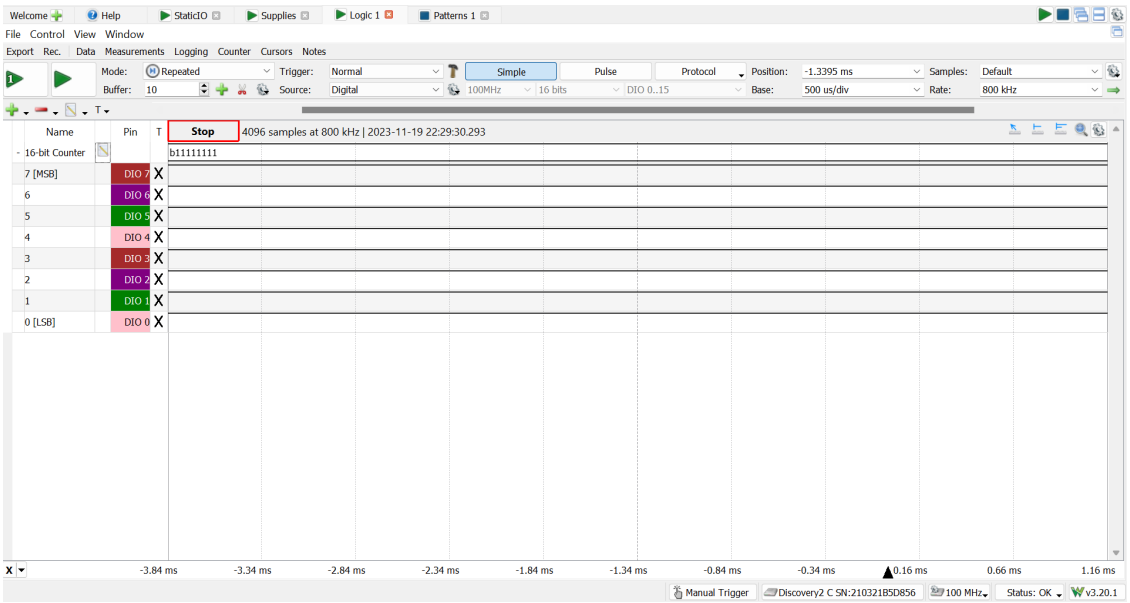

Figure 17: Setup of DIO output pins on Logic to read output (value) from counter

Results

The counter worked as expected, where each bit of increasing significant would switch states (from LO to HI or vice versa) at frequencies decreasing by a factor of 2.

At a binary value of 11111111 (seen in Figure 17), or 255 in decimal form, the counter overflowed and just reset back to 00000000 (0 in decimal form) and continued counting again.

Understanding:

Since we connected two 4-bit counters, when the individual counters can count to 2^4, 15, the now 8-bit counter can count to 2^8, 255 (since the counter starts at 0, not 1). Hence, at 11111111, or 255, the counter overflowed or went back to 0.

When integrated with the circuit, this counter is responsible for measuring the servo motors speed by counting the number of partial turns it makes - this will be explained in further detail in the Motor Sensor and Driver section.

[112]:
```
Image(filename = img_path("L7A_counter operation.png"))
```

[112]:

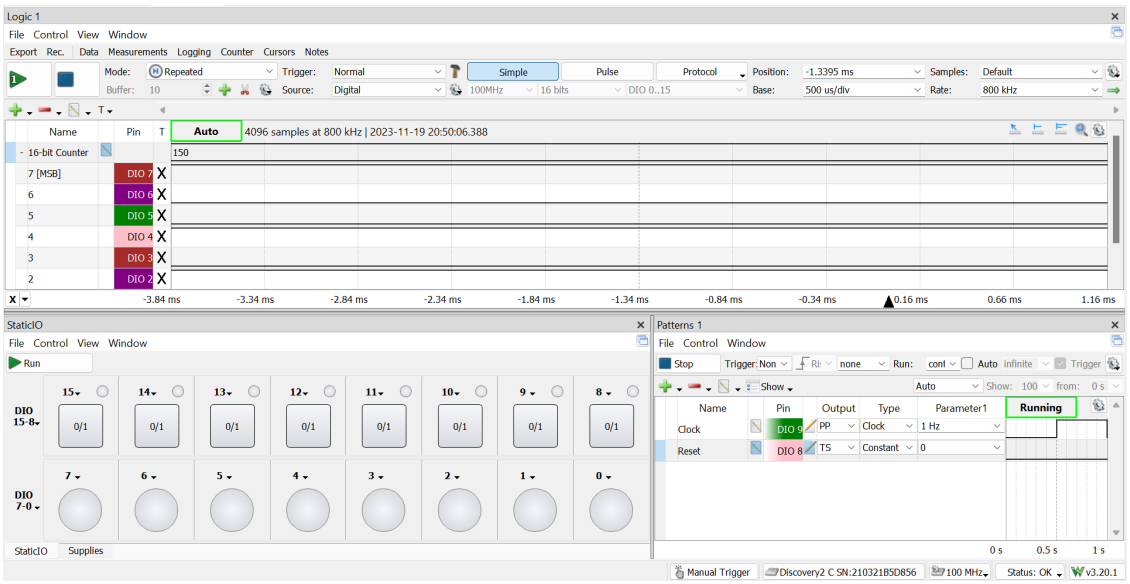

Figure 18: Working counter at decimal value of 150 with clock and reset inputs visible on bottom right

2.2.2 LATCH Circuit

The latch uses a flip-flop IC to store the value in the counter while the counter recounts up to the previous value it stored before it was cleared by the reset pulse.

Procedure

- I followed the datasheet for the 74HCT273 octal flip-flop here: https://assets.nexperia.com/documents/data-sheet/74HC_HCT273.pdf
- This time I used th AD2's AWG instead of WaveForms' Patterns to generate the input pulses
  - I connected the Counter IC's clock terminal to the breakout board's WaveForm Gen 1

- I connected the input terminal of the latch and reset pulse generator to the breakout board's Waveform Gen 2

- I played with the frequencies of the clock input into the counter and the clock input to the latch to get different latch values. An example of these frequencies is in Figure 21

Troubleshooting

#TS I used the wrong pinout from an incorrect datasheet. :(Luckily, before I actually did any testing, the TA informed us about this so I switched the wires to the correct terminals (following Figure 19)

#TS I couldn't understand why my latch value was constantly at 0. With Professor Jones's help, I realized that I couldn't set the counter's clock input, and the reset and latch input to the same to the same frequency, since that meant the counter effectively reset itself at the same rate it counted. Thus, I tried inputting the clock at frequencies significantly higher than the 5 Hz of the reset and latch clock signals, which fixed this issue.

[113]:
```
Image(filename = img_path("L7A_latch IC pinout.jpg"))
```

[113]:

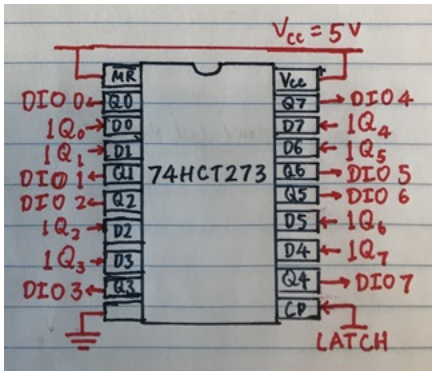

Figure 19: Pinout of latch with AD2 DIO pins and inputs from counter (labeled Q_0 through Q_7 according to counter's LSB to MSB)

[114]:
```
Image(filename = img_path("L7A_testing latch setup.png"))
```

[114]:

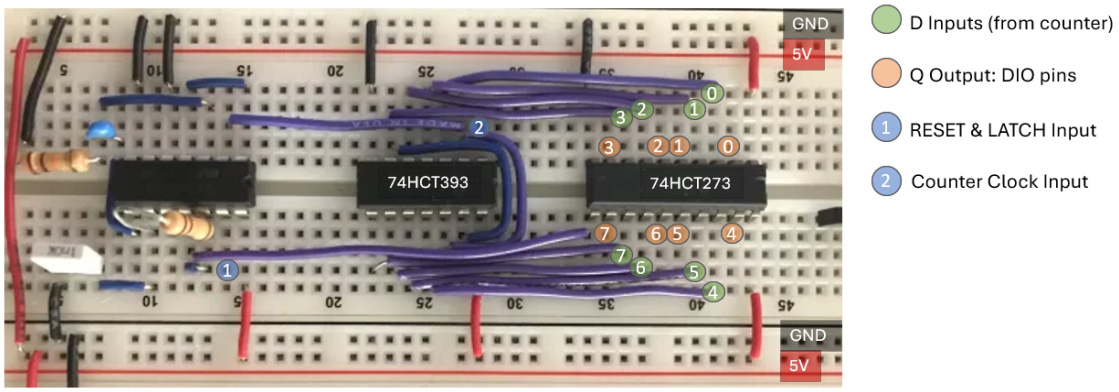

Figure 20: Circuit setup on breadboard of latch

[115]:
```
Image(filename = img_path("L7A_testing latch outputs.png"))
```

[115]:

Figure 21: Example of waves outputted from AD2 Wavegens to test latch with CH1 as latch clock and CH2 as counter clock

Results

Initially, I only saw the value the DIO 0 pin was reading change; only the signal from DIO 0 fluctuated up and down; the rest stayed in the LO state. After I fixed this in the TS described above, I found that varying the frequency of the signal I input into the clock's changed the value the latch read from the counter and outputted, but for one frequency, this value stayed constant.

In Table 1 in the DAC section, I explore the relationship between the counter's clock frequency and the latch's output value (the value it reads from the counter) further.

Understanding

Since I also set up the latch Q outputs with WaveForm Logic's bus feature in the decimal form with the same endian I configured the counter to count at, the latch would just output the decimal value the counter was counting to before it reset.

The relationship between the frequency of the RESET and LATCH signals and the counter clock means that the counter value and output of the latch reset at the same rate (the RESET and

LATCH signal frequency), and the counter output values count up in the time between the resets, which doesn't change, thus the latch always records one value for a counter clock frequency.

The proportionality I observed between the clock frequency and the latch value supports my assumption that it works properly, and this assumption is confirmed with the results in Table 1.

[116]:
```
Image(filename = img_path("L7A_latch q values with 330 Hz clock.png"))
```

[116]:

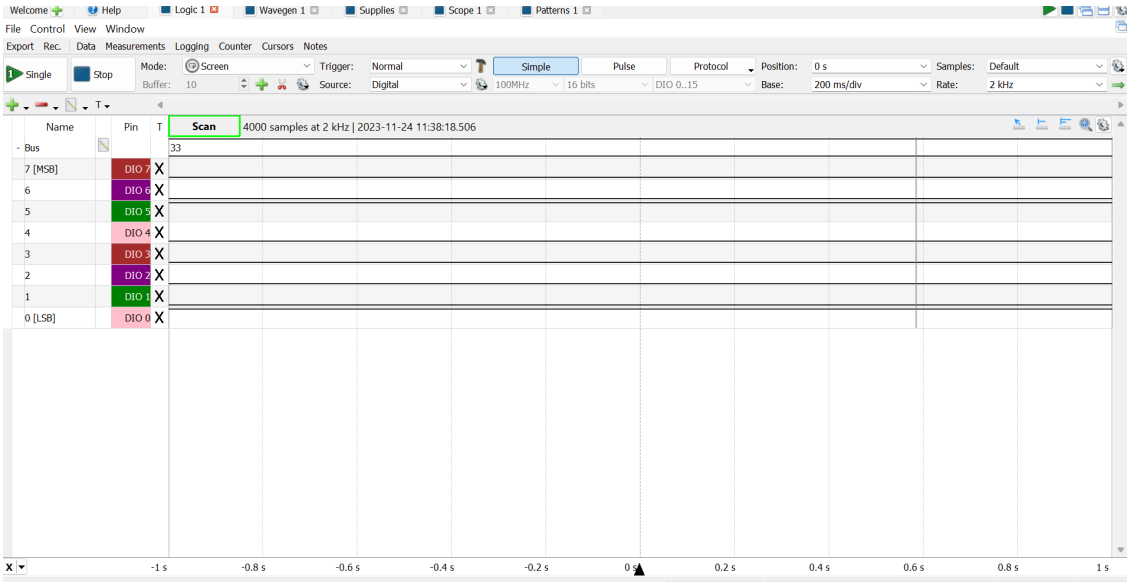

Figure 22: Output of latch with test counter and latch clock values in Figure 21

2.2.3 DAC

The Digital-to-Analog Converter, which consists of an R2R resistive ladder and an op-amp buffer. The resistive ladder transforms the binary value stored in the latch (since the latch stores the counter clock in 8-bit binary form)

Procedure

- Datasheet for resistor network DAC: https://www.mouser.ca/datasheet/2/54/r2r-1018811.pdf
- Datasheet for op-amp: https://www.ti.com/lit/ds/symlink/ua741.pdf
- We power the buffer op amp with V_{CC} = 5 V and V_{CC-} = -5 V so its output range goes from -5 go 5 V.
  - Note, in actuality, we expect the op amp to rail at around $\pm$ 3.7 V, which we observed in Lab 6.
- I connected the Q output bits from the latch to the resistive network in increasing order, as seen in Figure 22.5. I grounded one end of the resistive network and read the output voltage from the pin on the far side of the resistive network.

Troubleshooting

#TS: Initially, I had the DAC set up on one side of the gap on the breadboard and the wires connecting the bits from the latch to the DAC via the connectivity of the breadboard's columns, but I read nothing from the output terminal of the resistor network. Then I realized columns across the gap in the breadboard are not connected. Once I bridged the gap with wires, I read expected values from the DAC.

[117]:
```
Image(filename = img_path("L7A_dac setup.png"))
```

[117]:

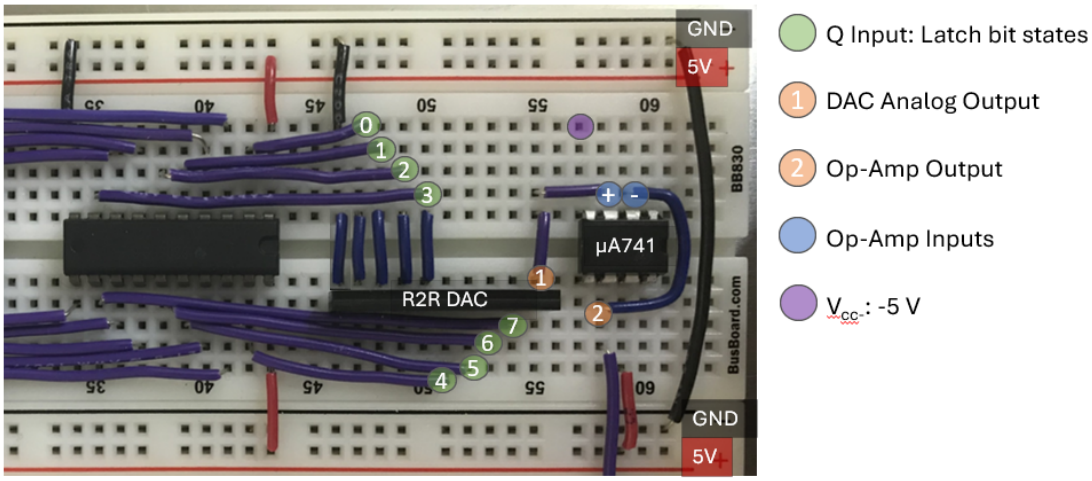

Figure 22.5: Setup of DAC on breadboard, consisting of a resistor ladder and an op amp

Results

I tested the output of the DAC with varying input frequencies to the counter's CLK terminal, and found that the DAC output was indeed proportional (somewhat) to the output value of the latch/counter and thus clock frequency.

Table 1. Counter and DAC output with varying counter clock frequencies

Clock Frequency	Counter Output	DAC Output
165 Hz	34	0.65 V
330 Hz	64	1.27 V
660 Hz	132	2.52 V
900 Hz	180	3.42 V
1000 Hz	192	3.72 V
1320 Hz	0	0.182 V

Understandings

Since the counter and latch reset at a frequency of 5 Hz, and I wanted to test the maximum value the DAC could read, I multiplied 5 by 255, the maximum value the counter could counter to, which gave me 1275 Hz, the frequency I set the counter CLK input to. Indeed, we see in Table 1 and Figure 23 that this gave me around a latch value of 246 and an analog DAC output of around 3.7 V. Since the op amp read 3.7 V, its upper voltage output bound, I knew that the latch succesfully outputted the maximum voltage. However, note that while I was worried the motor would spin fast enough to lead to the op amp railing and thus not provide any useful information on the speed of the motor, the TA reassured me this would never happen, and besides, the control loop of the op amp would ensure the speed of the motor couldn't exceed its corresponding DAC output voltage (which always correspondings to motor speed).

We observe in Table 1 that the DAC output is proportional to the counter/latch output, which is proportional to the clock frequency - up to a certain point, all details that point to the proper operation of the counter, latch, and DAC. The counter/latch output stores how many times the counter increments between the 5 Hz reset pulses, so naturally I multiply the clock frequency by this time frame, 0.2 s. Thus, we expect that the counter/latch value is roughly a fifth of the clock frequency. This expectation is supported by the values in Table 1. And of course, because of the very nature of DAC operations, which converts a binary value to an analog voltage, we expect the DAC output voltage to scale proportionally with the value stored in the latch. Again, this does occur in Table 1 - with the exception of latch values that lead to railing in the op amp, since at that point all latch value above roughly 195 will convert to the 3.7 V the op amp is capable of outputting.

Table 1 also shows us an interesting phenomenon; once we input a clock frequency past the frequency corresponding to the maximum value the counter outputs, we get a counter output of 0. Unsurprisingly, the counter output would be lower than that of lower frequencies, since the counter will overflow and start counting from 0 again, but it's interesting how the counter records exactly 0.

While the resistor network is what actually converts the digital signal from the latch to an analog voltage, the op-amp bolsters this voltage, since the output of the resistor network, after going through a series of resistors, may be too low to offer helpful information to the error signal amplifier op amp (i.e. might be difficult to make V_+ comparable to V_{in}).

[118]:
```
Image(filename = img_path("L7A_12.76 kHz counter.png"))
```

[118]:

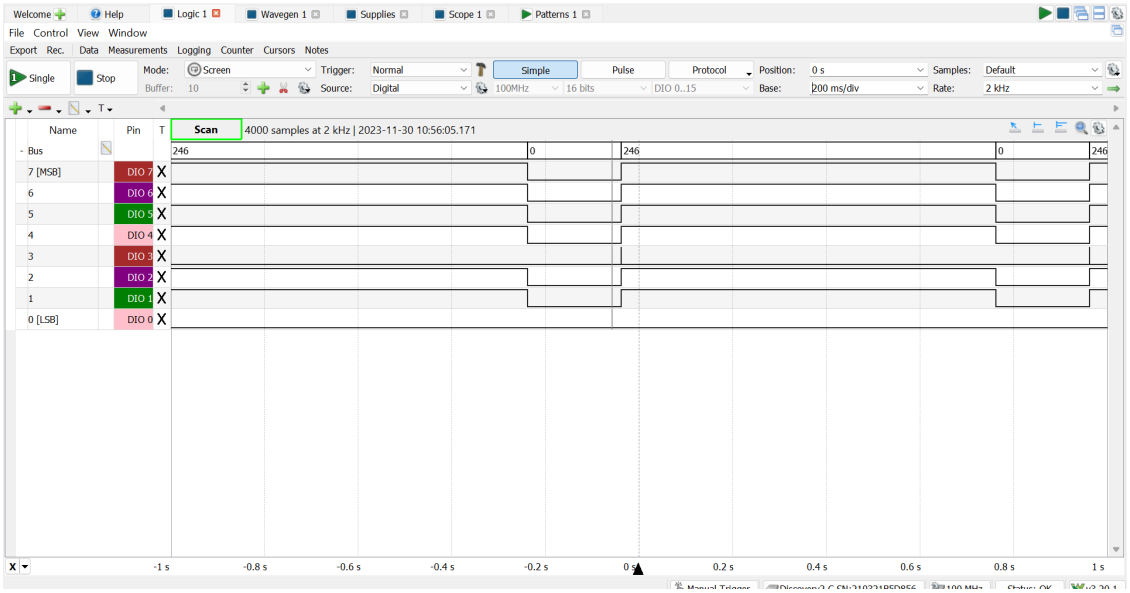

Figure 23: Maximum latch/counter value outputted with a counter clock frequency of 12.76 kHz and counter reset and latch clock frequency of 5 Hz

[119]:
```
Image(filename = img_path("L7A_max op amp output at 12.5 kHz.png"))
```

[119]:

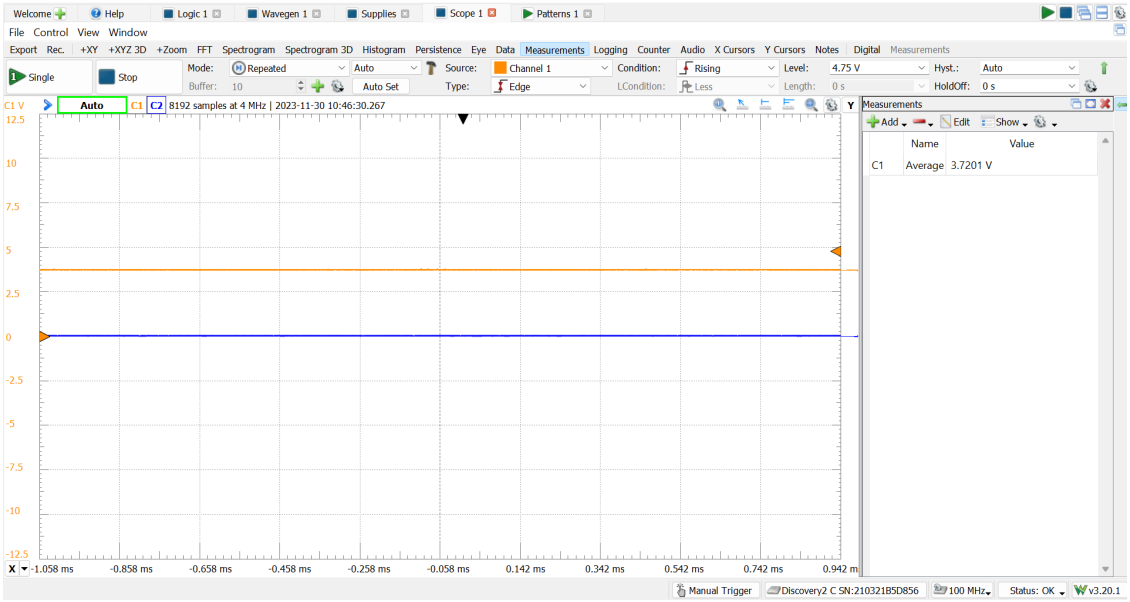

Figure 24: Op-amp in DAC observed to be railing at 3.7 V when latch stores a value of 246

2.3 Motor Sensor and Driver

[120]:
```
Image(filename = img_path("L7A_esa and motor schematic.png"))
```

[120]:

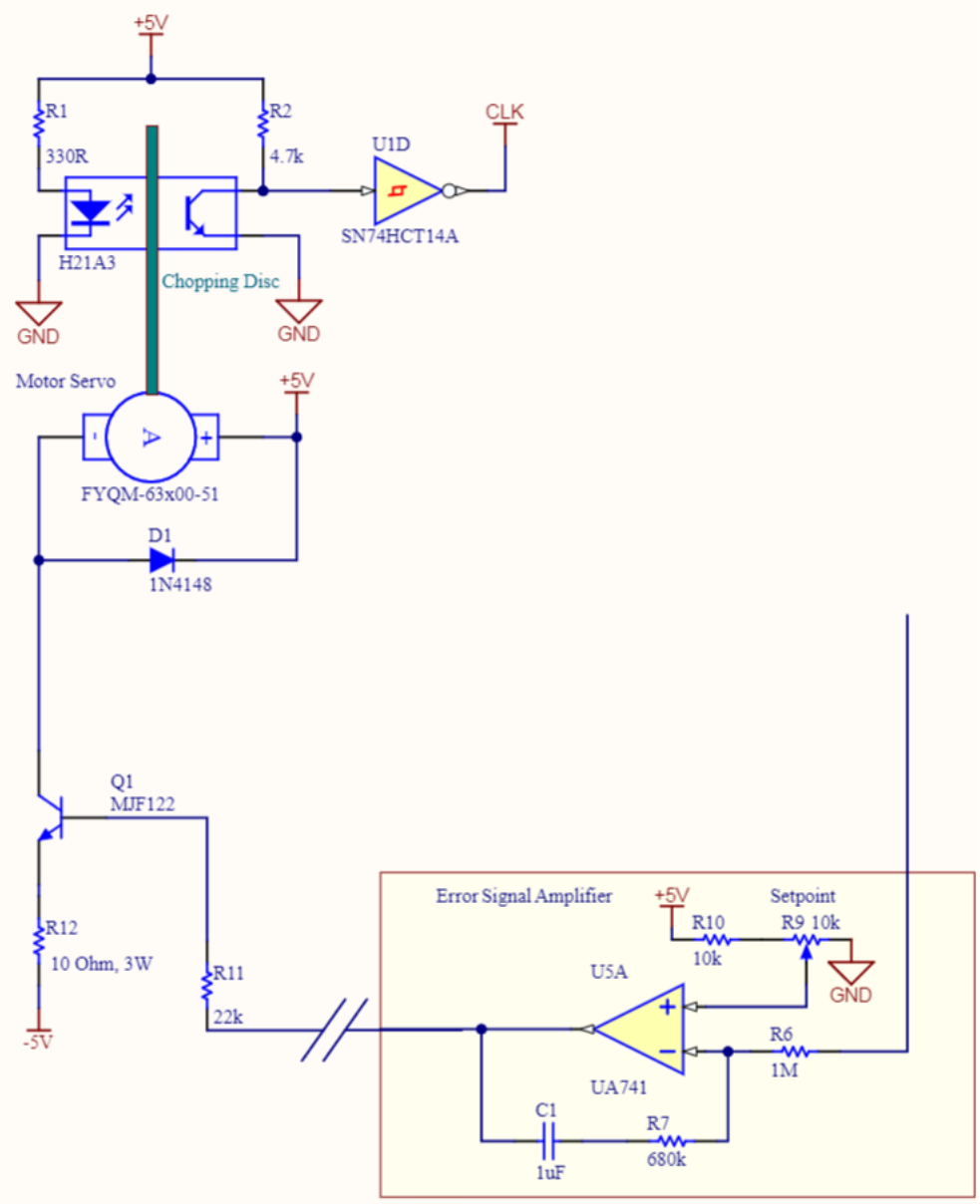

Figure 25: Functional circuit schematic of motor sensor and driver (error signal amplifier and motor itself)

2.3.1 Error Signal Amplifier

The error signal amplifier is integral (no pun intended) to enabling us to adjust the motor's speed. As an op-amp, it, well, amplifies the difference between the two input signals - both of which

contain information about the motor and the operation of the signal, and uses this information with a transistor to change the speed of the motor. This process will be described in more detail below.

Derivation of output voltage of error signal amplifier as function of input voltage

If we have V_{in} as the signal the DAC outputs, from Figure 25, we decide that the current flows from V_{in} to V_{out}, so we have current

$$I = \frac{V_{in} - V_+}{R_6}.$$

Via nodal analysis, we realize that

$$V_+ - IR_7 - V_C = V_{out}.$$

Since we have the current through a capacitor,

$$I = C\frac{dV_C}{dt},$$

we know

$$V_C = \frac{1}{C_1}\int I(t)\ dt.$$

Plugging these back into our nodal equation, we get

$$V_+ - \frac{V_{in} - V_+}{R_6}R_7 - \frac{1}{C_1}\int \frac{V_{in} - V_+}{R_6}dt = V_{out},$$

which we can rearrage to

$$V_{out} = \frac{R_6 + R_7}{R_6}V_+ - \frac{R_7}{R_6}V_{in} - \frac{1}{C_1R_6}\int V_{in} - V_+ dt.$$

Actually substituting in our resistor values, we get

$$V_{out} = \frac{1.68 \times 10^6}{10^6}V_+ - \frac{10^6}{6.8 \times 10^5}V_{in} - \frac{1}{0.68s}\int_0^t V_{in} - V_+ dt,$$

which evaluates to

$$V_{out} = 1.68V_+ - 1.47V_{in} - 1.47Hz\int_0^t V_{in} - V_+ dt.$$

Procedure

- I used the same datasheet for the op amp as the op amp in the DAC (see earlier section) since they are the same model
- I set up the error signal amplifier circuit like in Figure 26

- I used this datasheet for the BJT: https://www.onsemi.com/pdf/datasheet/mjf122-d.pdf
  - I plugged it into the breadboard following the pinout in Figure 28a
- Just to test the operation of the error signal amplifier, I used AD2's Waveform 1 generator to output a square wave voltage to the inverting terminal of the op amp and scoped the output
  - Since the op amp integrates the inputs, I knew to not have a constant input, else the op amp would rail, so I input a square wave that would go positive and negative (relative to the non-inverting input terminal) into the inverting input terminal.

Troubleshooting

#TS: I noticed that no matter how much I turned the potentiometer one way or another, non-inverting input voltage to the op amp did not change. I realized my potentiometer was broken, and switched it out for another potentiometer. This time, I saw the non-inverting input votlage to the op amp change as I turned the knob of the potentiometer.

#TS: The op-amp was railing, and after some discussions with my teammates, I realized that even though the inverting terminal voltage was going negative and positive relative to the constant non-inverting terminal, the square wave wasn't symmetric about the non-inverting temrinal voltage, V_+. It was close, but with an offset slightly below V_+, so the voltage was still overall more negative, and when you integrate that non-zero voltage over time, the op amp rails. Hence why I observed the op amp constantly at ~-3.7 V. I adjusted the square wave to have a offset of V_+ so the positive and negative portions wold actually cancel out, and this time, I watched the op amp output go to a constant voltage that wasn't $\pm$ 3.7 V. The op amp output didn't go to 0 V though, since there's still a $E(t)$ in the V_{out} term in the equation that takes different multiples of V_+ and V_{in} before taking their difference, so even if $V_+ = V_{in}$ in the long-term, the op amp output is still non-zero.

[121]:
```
Image(filename = img_path("L7A_Error signal amplifier setup.png"))
```

[121]:

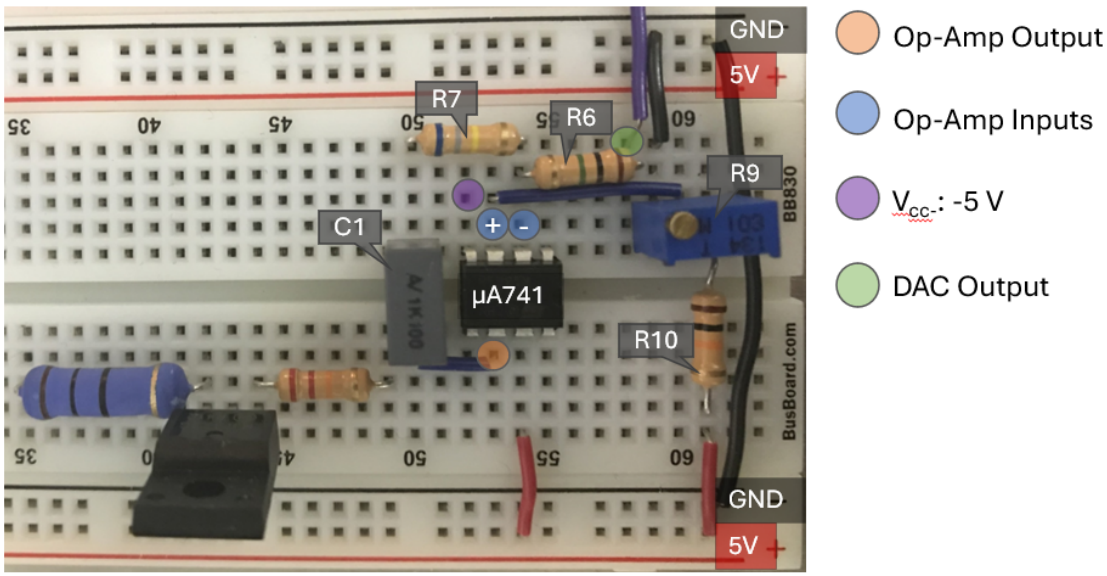

Figure 26: Circuit setup of error signal amplifier on breadboard

[122]:
```
Image(filename = img_path("L7A_motor driver bjt setup.png"))
```

[122]:

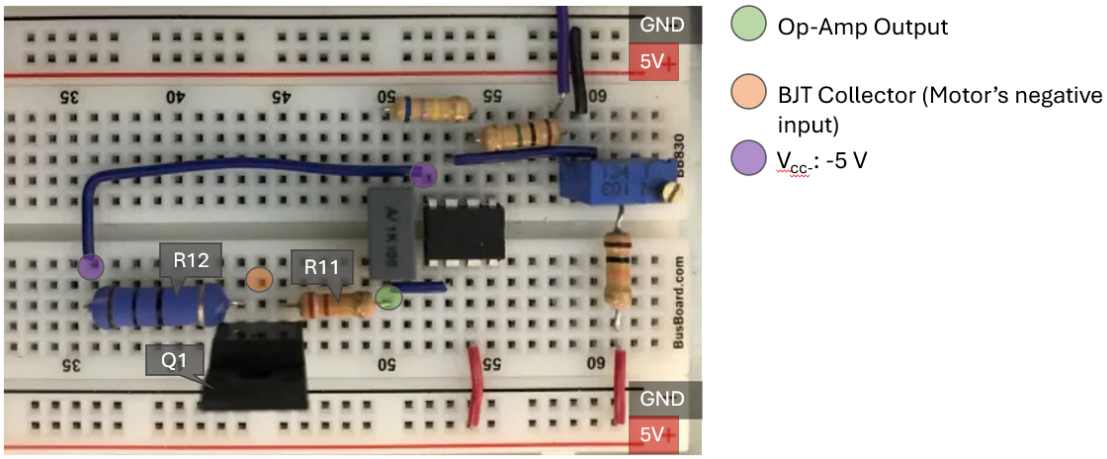

Figure 27: Circuit setup of BJT that connects breadboard circuit to motor

[123]:
```
Image(filename = img_path("L7A_bjt datasheet.png"))
```

[123]:

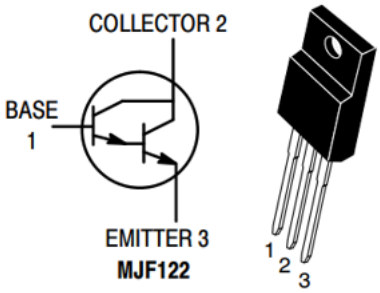

Figure 28a: Pinout of MJF122 BJT used in motor driver

[124]:
```
Image(filename = img_path("L7A_bjt schematic.png"))
```

[124]:

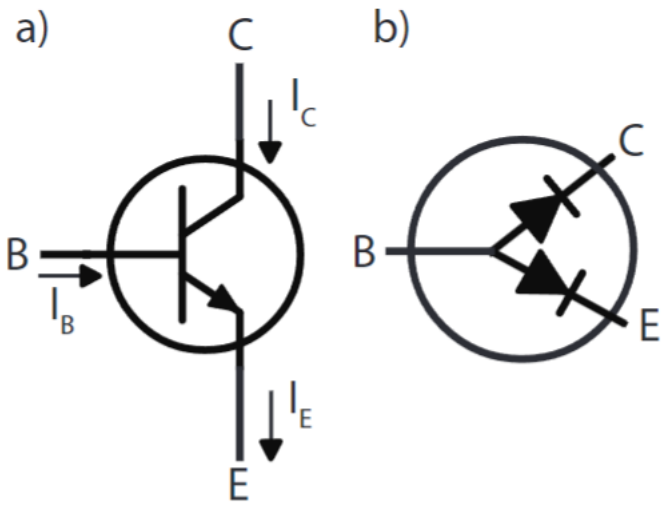

Figure 28b: a) Block diagram of BJT. b) Simplified diode model equivalent of BJT. Note that you cannot make a BJT by connecting two diodes back to back.

Results

The op amp behaved as expected, integrating the inputs to its two terminals. I'll discuss this further below.

Understandings

Above, we derived the output of the error signal amplifier as a function of the inputs to the op amp like this:

$$V_{out} = \frac{R_6 + R_7}{R_6} V_+ - \frac{R_7}{R_6} V_{in} - \frac{1}{C_1 R_6} \int V_{in} - V_+ dt.$$

We can simplify this function to a more understandable function of time, but let's first define an error function

$$E(t) = \alpha V_d(t) - \beta V_c(t),$$

where $V_d(t)$ represents the desired voltage, also known as V_+, and $V_c(t)$ represents the current voltage outputted by the DAC in the earlier section, also known as V_{in}, and and are some constants. We can understand that this error function, $E(t)$, is a measure of how far apart these two values are.

Looking at V_{out}, we note that $E(t)$ is incorporated in V_{out} as the $\frac{R_6 + R_7}{R_6} V_+ - \frac{R_7}{R_6} V_{in}$ term., but there's a $-\frac{1}{C_1 R_6} \int V_{in} - V_+ dt.$ term left. Interestingly, we see that the argument of the integral also fits the $E(t)$ format, so we can ultimately simplify V_{out} of the op amp and really, the entire

error signal amplifer circuit to

$$V_{out} = E_1(t) + y \int E_2(t)dt,$$

with y a value that accounts for the coefficients in the integral term.

So what the error signal amplifer circuit does, is, well, use the error between the desired voltage and the voltage it reads from the DAC - which stores information about the current state (speed) of the motor to do something. Now you might ask, do what?

The error signal amplifier is effectively a form of a PI, proportional integral, controller. It uses the current error of the system - the difference between the current and desired state, multiplied by some constant of proportionality (hence the 'proportional') *and* an multiple of the integral of the error of the system built up over time, to lower this error. In this case, the desired voltage, V_+ represents our desired motor speed, and V_{in} from the DAC is the current speed of the motor. Thus, this part of the circuit is really what controls the speed of the motor and enables us to do so by changing the desired voltage, V_+, with the adjustable potentiometer.

Note that the error signal amplifier doesn't behave exactly like a PI controller, since

$$E(t) \neq c(V_d(t) - V_c(t)),$$

where c is the constant of proportionality; rather, the resistors are configured in a way that means $V_d(t)$ and $V_c(t)$ are separately multiplied by constants.

The output signal of the actual error signal amplifier goes through R11, the 22 kΩ resistor, and via Q1, directly affects the motor input voltages. As a BJT, Q1 lets us control a larger current, going from its collector to emitter, via a smaller current going to its base. In Figure 28b (a), we see the the smaller current, I_B, which comes from the op-amp, can control the larger current through the BJT, I_C, which is what powers the motor. In Table 2, we see that the voltage at the BJT collector (which connects to the motor) becomes progressively more negative as the speed of the motor increases, which makes sense, since the greater the voltage difference between the positive and negative supplies of the motor, the more current it has and the faster it spins.

2.3.2 Motor Control with BJT - Manual Motor Test

The motor is what the whole circuit is controlling.

The DC motor spins through a phototransistor housing, and uses the phototransistor to count the number of holes in the motor disc that pass through the housing. Each time these happens, a signal is sent to an inverter connected to the counter clock that increments the counter (see counter section). This (with some calculations) keeps track of the speed of the motor. And well, the motor is what the whole circuit is controlling.

In this part, we explore how the motor uses its phototransistor and the holes on the rotating disc to track the number of revolutions the motor disc undergoes.

Procedure

- Datasheet for phototransistor used in motor: https://www.farnell.com/datasheets/13244.pdf
- I set up the motor sensor circuit on the breadboard next to the motor as seen in Figure 30, following the schematic in Figure 25.
  - Note that I did not plug in the voltage supplies
- I connected the output of the motor sensor circuit's transistor to my AD2's Scope 1+ terminal, and just read the signals on WaveForms
- I spun the motor disc at varying speeds with my hand too check if the signals outputted were what I expected

Results

We can see the results of testing the motor and phototransistor manually in Figure 29. When I rotate one of the holes on the motor disk through the phototransistor housing, I observed the output of the phototransistor circuit (note: not actual phototransistor) go LO briefly, and when I kept the hole in the phototransistor housing, I watched the output stay LO.

Understandings

The motor counts how many revolutions it has turned by using a phototransistor to keep track of how many times one of the holes in the motor disk passes through the phototransistor housing, switching its state from off to on, or LO to HI. Although the number of these pulses doesn't correspond exactly to the number of full motor revolutions, they are related by a constant of proportionality. We can understand the total number of full rotations, R, as

$$R = \frac{C}{\text{no. of holes on disk}},$$

where C is the total number of times a hole passes through the phototransistor housing, and the number of holes on the disk is 10 (I counted). Because we reset the counter at a frequency f, what we actually see on the counter as the speed of the motor (# of revs per second) is

$$\text{rps} = \frac{Cf}{10}.$$

Converting that to Rpm, we have

$$\text{rpm} = \frac{Cf}{10} \times 60 s/min.$$

Right now, since our counter reset and latch clock frequency is 5 Hz, we get that the rpm of our motor can be calculated with

$$\text{rpm} = 6Cf/min.$$

We can see that the phototransistor is connected to a transistor connecting 5 V to gnd, so when it goes HI, the transistor closes and connects the 5 V and R2 to gnd, so the input signal to the U1D inverter goes LO, and the inverter outputs HI. When the hole fully passes the phototransistor, the

phototransistor outputs LO again, so the transistor opens and input signal to the U1D inverter is 5 V again, so the U1D inverter outputs goes from HI to LO. This signal is connected to the counter's CLK terminal, and since the counter is HI-to-LO edge-triggered, the counter increments once when the hole fully passes the housing.

Figure 29 shows how the voltage output of the motor goes low when the hole passes into the housing since the phototransistor going HI connects the output to GND. Otherwise, the output is connected to 5 V (there's a resistor, R2, but no current, so it's still 5 V)

[125]:
```
Image(filename = img_path("L7A_manual motor test.png"))
```

[125]:

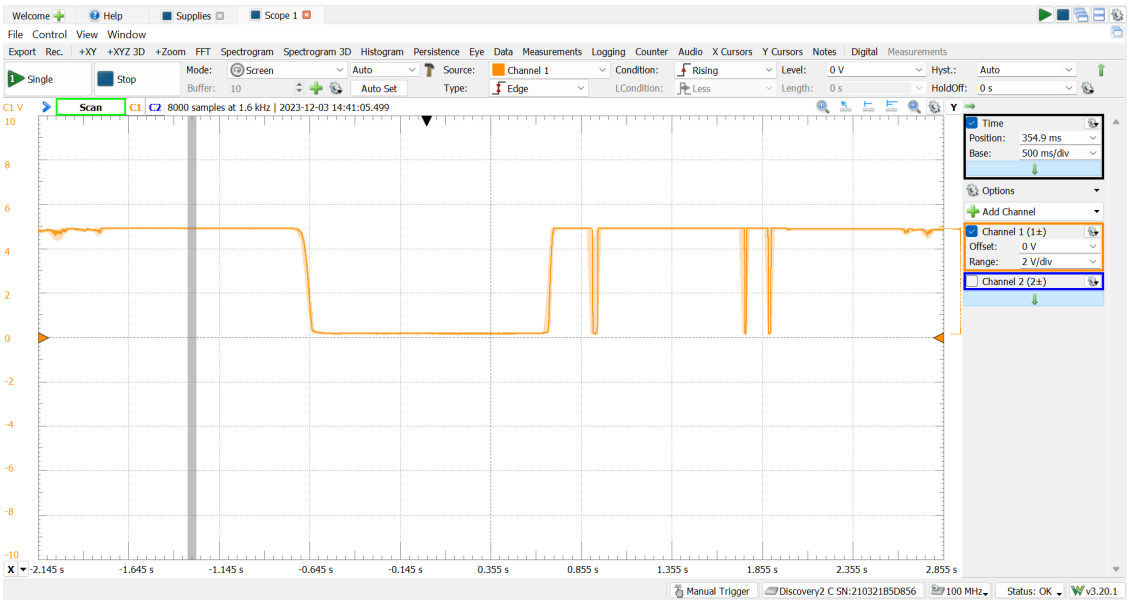

Figure 29: Motor phototransistor output when motor disc is manually spun

2.3.3 Motor Control with BJT - Running Motor

Here, we actually run the motor using a supplied voltage.

Procedure

- I removed the grounding wire from the unused inverter terminal on the inverter IC to set up the clock input to the counter
- I connected 5 V and -5 V wires from the AD2 power supplies to the same columns the red and black wires of the motor were plugged into, and a 10 Ω 3 W resistor in series between the two (seen in Figure 30)
  - I connected the output of the phototransistor transistor to the input of the remaining inverter left on the Schmitt-trigger inverter IC (seen in Figure 31) for the counter clock

- To run the motor without reset but still a latch reading, I inputted a 5 Hz signal via the Waveform 1 output on the Ad2 directly into the latch clock.
- Then, to enable the reset functionality, I switched the 5 Hz signal from the latch clock into the inverter IC (as seen in the RESET and DELAY circuit in Figure 2)

Troubleshooting

TS: My motor frequently just stopped running, but almost always, it was because a wire came loose, since I often had to extend wires from the main breadboard vertically up to the breadboard adjacent to the motor, where they didn't stay in well.

[126]:
```
Image(filename = img_path("L7A_motor running raw setup.png"))
```

[126]:

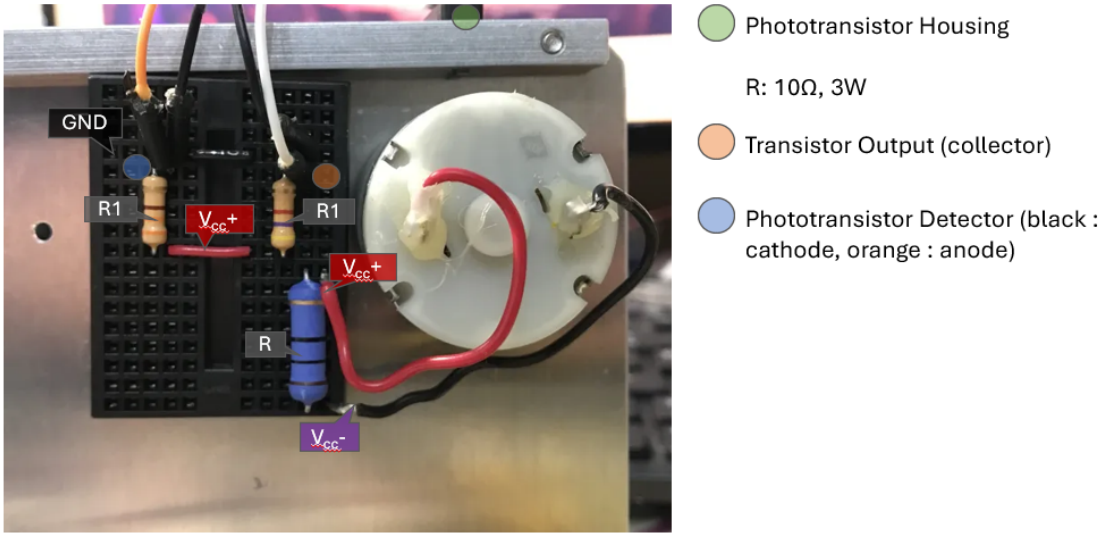

Figure 30: Motor sensor setup - note resistor on right is labeled wrong, and should be R2 instead of R1

[127]:
```
Image(filename = img_path("L7A_updated inverter pinout.jpg"))
```

[127]:

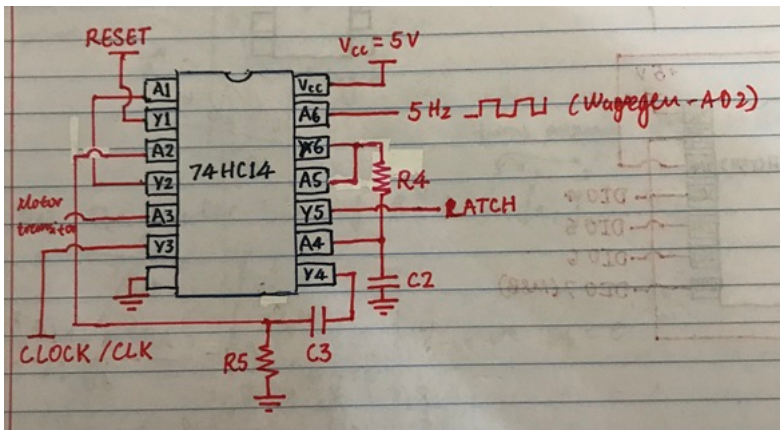

Figure 31: Updated inverter IC pinout (with counter clock signals from motor)

Results

When the motor ran without a reset signal, we see in Figure 32 that the counter was just either maxed out at 255 or randomly jumping between other values I couldn't get a screenshot of. Figure 33 shows us that with the reset enabled, the counter records a slightly lower value of 144 - which I didn't get a photo of, but take my word for it.

Understandings

When the motor ran with 5 V and -5 V into the positive and negative inputs without a reset, the latch was just flipping between random values, most stably at 255. But when the motor is powered from 5 V and - 5V, it spins so fast that in the 0.2s interval between the latch clocks, less than 255 holes rotate past the phototransistor, but a significant enough number do that the latch frequently reads the max value from the counter. When I enabled the reset, the counter value immediately dropped to 85, then climbed back up to 144, which I attribute to either just the motor's response to a sudden change in the system or one of the wires coming loosing then reconnecting.

The resistor is specifically rated for 3 W because it dissipates a lot of power. We have

$$P = \frac{V^2}{R},$$

and since $V_{cc+} = 5V$ and $V_{cc-} = -5V$, we get

$$P = \frac{100V^2}{10\Omega},$$

which evaluates to

$$P = 10W.$$

Since the resistor is only rated for 3 W, I assume a significant amount of power is used in the motor itself. However, this evidently isn't enough, since when I was running the motor for close to a minute, I smelled smoke, and when I touched the resistor to confirm my theory that too much current was going through it, it was so hot, I burned my finger.

[128]:
```
Image(filename = img_path("L7A_motor running raw.png"))
```

[128]:

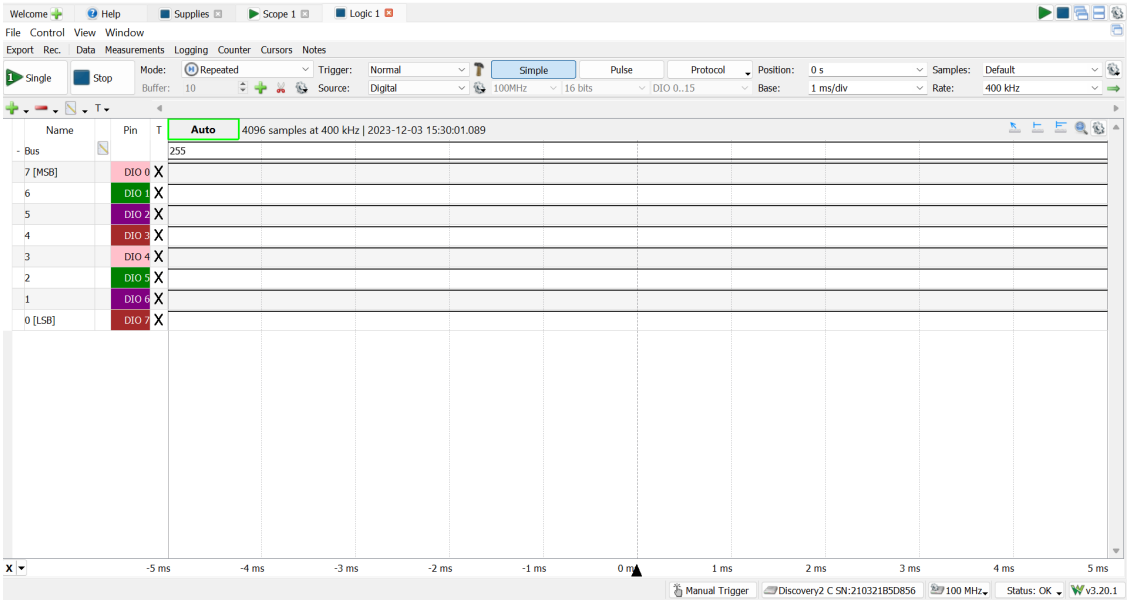

Figure 32: Latch output of motor running without counter reset signal

[129]:
```
Image(filename = img_path("L7A_motor running with reset.png"))
```

[129]:

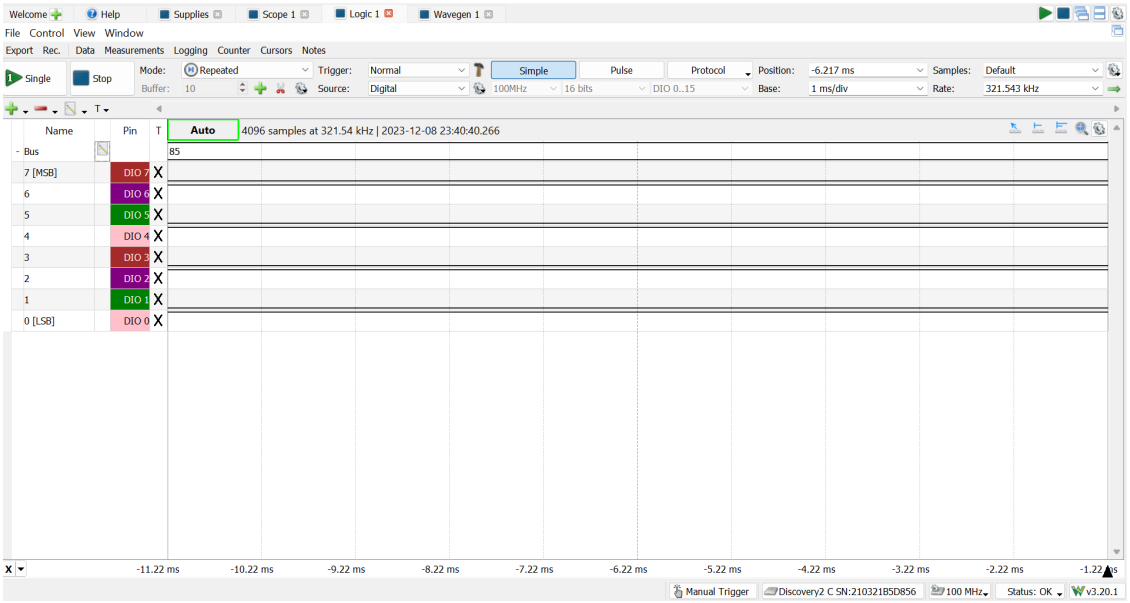

Figure 33: Latch output of motor running with counter reset signal enabled (it later went to 144 and stayed there, but I didn't get a screenshot of that)

2.3.4 Motor Control with BJT - Actual Motor

This is what it has all led up to… where all the components of the circuit we've worked on come together.

Procedure

[130]:
```
Image(filename = img_path("L7A_actual motor setup.png"))
```

[130]:

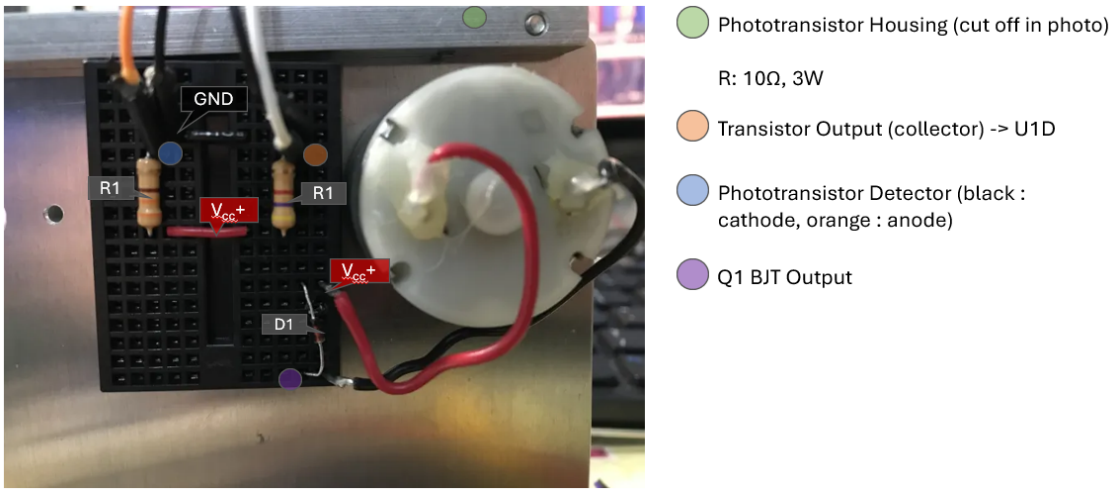

Figure 34: Full motor sensor setup, with speed control enabled via error signal amplifier and BJT Q1, note resistor on right is labeled wrong, and should be R2 instead of R1

Results

Once I turned the potentiometer all the way up, to an effective resistance of 10 kΩ, I achieved the maximum speed of the motor, which registered as 125 on the counter/latch. Using the conversion equation from counter output to rpm we derived in the above section, we find this translates to 3750 rpm, and roughly 63 full turns of the motor per second, which is very fast, but feasible.

Table 2. Relationship between V_+, latch/counter output, RPM of motor, and BJT's collector output

V_+ (control)	Latch/Counter Logic Output	RPM (revs/min)	BJT Output
0.42 V	20	600	3.2 V
0.79 V	40	1200	0.79 V
1.04 V	53	1590	0.8 V
1.17 V	60	1800	0.3 V
1.41 V	73	2190	-0.6 V
1.63 V	85	2550	-1.4 V
1.97 V	104	3120	-2.63 V
2.09 V	110	3300	-3.1 V
2.25 V	119	3570	-3.7 V
2.5 V	125	3750	-4.07 V

Figures 35 and 36 show the voltages scoped from the BJT output and V_+ input for various counter/latch outputs found in Table 2. Interestingly, both voltages destablize as they increase.

Understandings

Table 2 confirms our understanding that V_+, the voltage into the non-inverting termainal of the error signal amplifier op amp, controls the motor speed, as the table tells us the two are proportional by a factor of roughly ~5. And it does this by changing the BJT output. The BJT output is the voltage that goes into the motor's negative voltage supply, and the higher it is, the less of a voltage difference there is between V_{CC-} and V_{CC+}, 5 V, so the motor runs slow. Vice versa, the motor runs faster when the BJT output is lower, since there's a greater voltage difference and current.

Since the speed of the motor is controlled by the V_+ input to the error signal amplifer op amp, it is ultimately limited by the range of possible V_+ values. Since we get V_+ as the output of a voltage divider between a 10 kΩ resistor, R10, and a 10 kΩ potentiometer, R9, from a 5 V source and gnd (as visible in Figure 25), then the maximum V_+ occurs when R9 maxes out at 10 kΩ, so R9 = R10, and the minimum V_+ occurs when R9 is at a minimum resistance of 0 Ω. By the voltage divider formula of

$$V_{out} = \frac{R9}{R9 + R10} V_{in},$$

we realize that the maximum V_+ is 2.5 V and the minimum is 0 V. Thus,

$$0V \leq V_+ \leq 2.5V.$$

We also have a flyback diode between the positive and negative terminals of the motor to prevent damage to the BJT. When we have inductive loads like a motor, it dislikes sudden current changes, so if the BJT suddenly lowers the current rapidly, to oppose this, the voltage of a motor might spike to compensate for the lowered voltage. This spike may be violent enough to damage the BJT.

Earlier, we found the equation

$$\text{rpm} = 6Cf/min$$

to translate the counter output we read the the rpm of the motor. Well, if we want the counter to actually output the rpm, we have C = rpm, so we get

$$6f = 1.$$

This tells us we must input a frequency of $\frac{1}{6}Hz$ to the reset and latch generator circuit. However, we observe that since our largest value we read from the latch was 125, our maximum rpm was ~3750, which exceeds the largest number an 8-bit counter can store, which is 255. Thus, at minimum, we need an 12-bit counter for this (since $2048 = 2^11 < 2750 < 2^12 = 4096$).

[131]:
```
Image(filename = img_path("L7A voltage input and bjt 20.png"))
```

[131]:

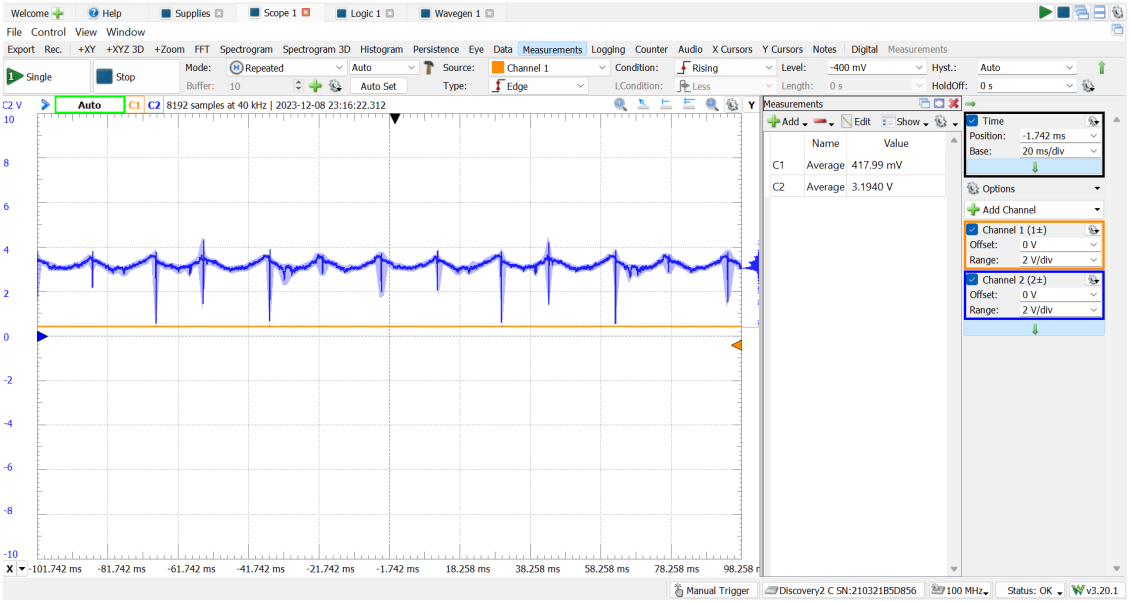

Figure 35: Q1 voltage output (motor negative voltage supply; CH2 blue) and V_+ (CH1 orange) when latch outputs 20

[132]:
```
Image(filename = img_path("L7A_DAC and bjt output 40.png"))
```

[132]:

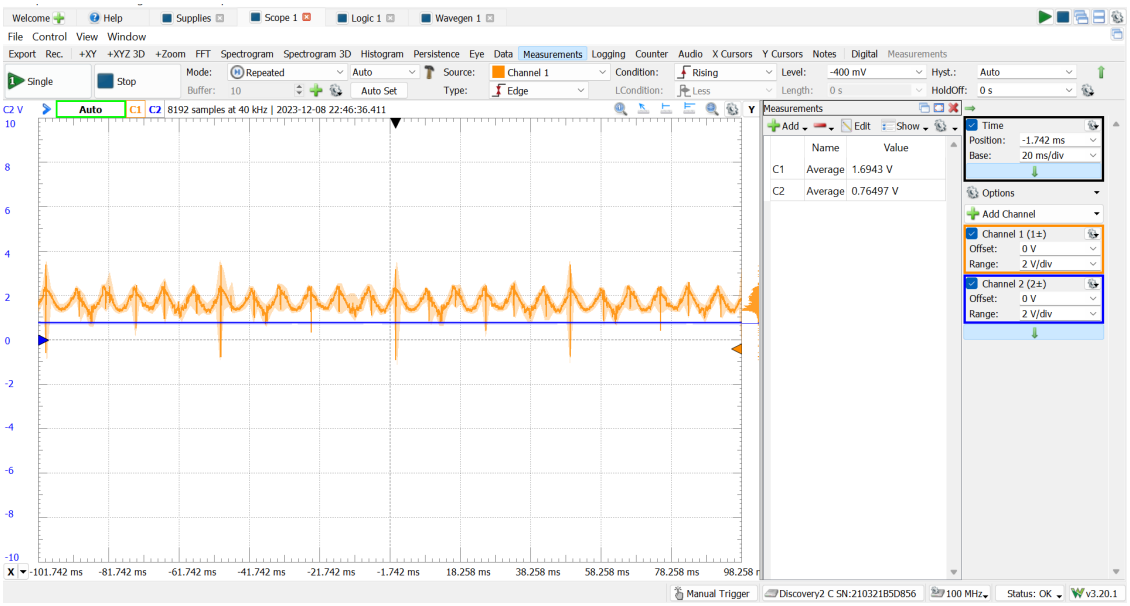

Figure 36: Q1 voltage output (motor negative voltage supply; CH2 blue) and V_+ (CH1 orange) when latch outputs 40

3 Conclusion

In this project, we combined our knowledge from all the prevoius labs in this course to operate a working servo motor.

The scale of the circuit meant I actually had to consider things like circuit organization. So, before I actually made the circuit, I developed a colour scheme for wires that'd help me understand the purpose of everything, and made sure to avoid things like overlapping wires for ease of troubleshooting and moving ICs and other components around.

3.0.1 RESET and LATCH circuit

Here, we used a Schmitt-trigger inverter to operate the counter and latch ICs and synchronize the singals to the counter and latch so the latch would read values from the counter before the counter values reset. To do so, we first input a 5 Hz square wave into the first inverter U1F, which outputs to the rest of the reset and latch circuit.

We created this delay using our understanding of an RC circuit, and took advantage of a capacitor, C2, and resistor, R4, to slow down the changing of the signals into an inverter, U1B. We created the reset signal for the counter with an additional RC circuit (the pair of C3 and R5 in Figure 2), which changed the square waves inputted from the Ad2 Wavegen to a short pulse-like signal.

But all this necessitated the use of a Schmitt-trigger inverter IC, which decreases noise by squaring up input signals, and the way it only inverts input voltages past a certain threshold that's different for HI and LO inversions is what enables the RC circuits to create the delay and reset signals.

This part of the circuit was mainly responsible for automating the operation of the rest of the circuit, as with a single 5 Hz square wave from the AD2 waveform generator, we can operate the counter reset and latch clocks.

As well, the inverter is necessary for translating the output of the motor phototransistor to the counter clock to count the holes that pass through the phototransistor housing.

3.0.2 Counter, Latch, and DAC

The counter, latch, and DAC combined to tell us the current speed of the motor.

With the counter, we combined the separate $\div 16$ counters to create a $\div 256$ counter that could count up to 255 with its 8 bits. The counter counts how many times the holes in the disc pass through the phototransistor in the time interval before it resets (0.2 s, since we use a 5 Hz reset).

We use the latch to store the values in the counter in a similar 8-bit fashion, and we send the data from its bits to a dac, which translate the digital data to an analog voltage.

The DAC consists of a resistor ladder, which converts the data from the 8-bits to an analog voltage. However, the output voltage may be low, so we then use a op-amp to buff this voltage, before sending this information about the motor's current speed to the error signal amplifier.

3.0.3 Motor Sensor and Driver

Error signal amplifer

Here, we used the negative feedback of an inverting op-amp to create a control loop we then used to control the speed of a motor. To create a PI (proportional integral) controller, we used our knowledge of the golden rules of an op amp, and how integrating op amps work.

The error signal amplifier introduced us to the concept and application of a PI controller, which uses negative feedback to control a signal. Similarly to PID (proportional integral differentiator) controllers, a PI controller uses its knowledge of the difference between the current and desired state of a system, and the integral of this error over time, to adjust the current state of the system until it matches with the desired state. Here, the states represent the speed of the motor, and we can control the desired speed of the motor by adjusting the voltage into the non-inverting terminal of the op amp.

PID controllers, which also accounts for the derivative of the error between the desired and current state, are a form of feedback control loop widely used in the industry, in things such as thermostats and other process control applications.

Motor

The motor is what the circuit actually runs. It uses DC positive and negative voltage supplies to run, and it's these supplies we adjust to change the motor speed.

I was unfamiliar with how the phototransistor worked, but reading the datasheet helped me understand how I could use it and the other characteristics of our circuit, such as the counter reset and latch clock frequency, or the number of holes on the motor disc, to calculate the motor's rpm.

Running the motor was also my first exposure to how important it is to be safe even with these low-power appliances we use in the lab. If I had left the motor running with 5 V and - 5V supplies and went away, I wouldn't have smelled the smoke and realized it was overheating, and shut off the supplies. Considering the 10 Ω resistor I plugged between the supplies of the motor to limit the current was hot enough to burn me after running for just under a minute, who knows what could have happened had I let the motor keep on running. As well, we were warned not to just plug in 5 V and -5 V as the supplies on the motor without a resistor, since we should understand theoretically that with the negligible resitance of a motor, the current would be far to high to be safe.

We ended with a maximum rpm of 3750.

[133]:
```
# @title
%%capture
!apt-get install texlive-xetex texlive-fonts-recommended␣
↪texlive-latex-recommended texlive-plain-generic pandoc
```

[134]:
```
# @title
# Capture to prevent lots of output... Remove this if troubleshooting!
%%capture

pdf_file_name = file_name.split('.')[0]+'.pdf'  # Same as file_name with .ipynb␣
↪changed to .pdf
!rm $file_name
!rm $pdf_file_name
```

```
import os

full_path = os.path.join(path, file_name)
!cp  "$full_path" ./

!jupyter nbconvert "$file_name" --to pdf
```