MECH 423 –2020W
Lab #3: Motor Control

Objectives

The goal of this lab is to learn how to drive and control DC and stepper motors, as well as to assemble a 2-axis gantry system. There are five parts of this lab: 1) DC Motor Control, 2) Stepper Motor Control, 3) Encoder Reader, 4) Closed loop control and 5) Two-axis control.

Logistics and Evaluation

This lab is done individually. Each student will be provided a lab kit consisting of all of the required components. Students will be required to source basic hardware tools. After completing each exercise, students submit their work to Canvas, following the **lab submission guidelines**. A written lab report is due after completion of the final lab exercise, also submitted through Canvas.

Lab Report

A written lab report is required for Lab #3. The lab report should provide a brief description for each part of the lab along with pictures, circuit schematics, and diagrams of the apparatus and code structure. **Please don't write the entire lab report at the end, each section should be written as it is completed.** The lab report should be submitted through Canvas. Plagiarism is absolutely not acceptable and will result in a zero in addition to further academic penalties.

Lab Exercises

1. DC Motor Control

1. Follow the instructions at the end of this document to assemble the gantry system. **It is advised that students develop controls without the pulley system connected to avoid crashing.**
2. Write microprocessor code to generate PWM waveforms for the H-bridge motor drivers. Run the timer in continuous mode so that the PWM duty cycle could be set with full 16-bit precision.
3. Write a C# program to send commands to set the PWM value and motor direction from a PC. A multiple-byte packet structure will be necessary.
4. Write microprocessor code to accept commands from the C# program to set the PWM duty cycle and motor direction accordingly.
5. In C#, use a variable knob or slider as a velocity controller for the motor that ranges from the maximum speed in the CCW direction to the maximum speed in the CW direction.
6. **Connect the DC motor pulley system.**
7. Demonstrate the working hardware and software and submit to Canvas.
8. Describe your apparatus and code in your report and include a screenshot of your C# program. Attach your MCU and C# code as appendices.
9. Draw an electrical schematic diagram of the minimum components necessary to drive a DC motor including the MCU, motor driver, power supply, and accessories. The drawings should be easily understandable and accurate. You should label the components, component values, as well as pin number and pin names involved. I suggest using Microsoft Visio, but feel free to use any drawing programs that you are familiar with (e.g. Illustrator, Photoshop, Paint, Inkscape, Corel Draw, AutoCAD).
10. What is the minimum PWM duty cycle for the motor driver to generate a reasonable output waveform? Check using an oscilloscope. Why does this limit exist? Discuss in your report.

11. What is the minimum PWM duty cycle for the motor to turn at no-load? Why does this limit exist? Discuss in your report.

Video Deliverables:

1. C# user interface with a variable velocity control element.
2. Velocity control of stage with continuous range from **max CCW** to **max CW**. Note: Should cross-over zero velocity position.

2. Stepper Motor Control

1. **It is advised that students develop controls without the pulley system connected to avoid crashing.**
2. Write microprocessor code to generate the appropriate signals to control a stepper motor using half-stepping. Hint: make a look-up table and use a state variable to step through the table.
3. Write microprocessor code to command a single half-step control (both forwards and backwards) of the stepper motor commanded from a message from the UART.
4. Write microprocessor code to run the stepper motor at a constant speed with the speed commanded from a message from the UART
5. In C#, use a variable knob or slider as a velocity controller for the motor that ranges from the maximum speed in the CCW direction to the maximum speed in the CW direction and allows for single step control in both directions.
6. **Connect the stepper motor pulley system.**
7. Demonstrate the working hardware and software and submit to Canvas.
8. Describe your apparatus and code in your report and include a screenshot of your C# program. Attach your MCU and C# code as appendices.
9. Draw an electrical schematic diagram of the minimum components necessary to drive the stepper motor including the MCU, motor driver, power supply, and accessories.
10. What is the maximum speed of this stepper motor (in steps/s and rev/s)? Why does this limit exist? Discuss in your report.

Video Deliverables:

1. C# user interface with single-step commands for **CCW and CW** and a **variable velocity control** element.
2. Demonstrate 5 single-steps in CCW and 5 single-steps in CW.
3. Velocity control of stage with continuous range from **max CCW** to **max CW**. Note: Should cross-over zero velocity position.

3. Encoder Reader

1. Assemble the encoder reader circuit using instructions provided at the end of this document.
2. Manually turn the shaft and check the appropriate signals are being transmitted to the timer A0 and A1 clock inputs using an oscilloscope.
3. Write microprocessor code to periodically capture the encoder counts and transmit it over UART. Use a timer interrupt to capture the encoder count at a regular interval.
4. Write a C# program to read in the encoder counts and calculate the rotational velocity in Hz and RPM. Display the instantaneous position and velocity in a textbox. Use the graphing object to plot the position and rotational velocity versus time.
5. Manually turn the motor shaft and show the shaft position and rotational velocity data are correct. Save example data for your report.
6. Incorporate elements from Exercise 1 to generate a PWM signal to turn the motor. Measure the rotate rate versus PWM duty cycle. Describe the experiment and results in your report.
11. Demonstrate the working hardware and software and submit to Canvas.

12. Describe your apparatus and code in your report and include a screenshot of your C# program. Attach your MCU and C# code as appendices.
13. Draw an electrical schematic diagram of the minimum components necessary to obtain and process signals from the encoder, including the MCU, latches, power supply, and accessories.

Video Deliverables:

1. C# user interface showing **real-time** plot of motor position and rotational velocity versus time.
2. Manually moving the motor shaft **CW** and then **CCW** to demonstrate change of direction in the **real-time** plots.
3. Control the motor using a variable velocity control element and show the changing plots in **real-time**.

4. Closing the Loop

1. Modify the C# and microprocessor code from Exercise 4 to apply a step input to the motor (e.g. PWM duty cycle changing from 0-25%, 0-50%, 0-100%, etc) while simultaneously acquiring the shaft position.
2. Measure the rise time of the motor and encoder for a few different magnitudes of the step input. Using this information, develop an estimated transfer function of the motor and encoder.
3. Use your estimated transfer function to develop a block-diagram model of a closed-loop position control system using the motor and encoder.
4. Implement a simple proportional control on the microprocessor using the 32-bit hardware multiplier with appropriate scaling. You may need to implement a saturation limit to the output of the control law to avoid exceeding physical limits of the controller and actuator. **(Hint: Think of zero-order hold discretization)**
5. Measure properties of your closed-loop control system such as rise time, overshoot, and settling time.
6. Demonstrate the working hardware and software and submit to Canvas.
7. Describe your work in your report. Include block diagrams and equations of your feedback control system. Also, include a copy of your MCU code as an appendix.

Video Deliverables:

1. DC gantry stage moving to 5 different locations, inputted from the C# program. Hint: Locations can be **absolute** or **relative** to current location.
2. Manually moving the stage from its set point and the stage returning. Hint: The control algorithm should try to compensate for the external force and return to its original position.

5. 2-axis Control

1. Using the software, you have developed in exercises 1-4, you will now put it all together to simultaneously control both axes.
2. Write microcontroller code to receive a package of bytes that control both the x and y axis. **(Hint: Think of G-code.)**
3. Write a C# program that takes a relative [X,Y] distance and a velocity as an input, and controls the gantry system to move there in a **straight line**. **(Hint: Both axes must arrive at the same time)**
4. Tape a piece of paper onto the stage and attach a pencil or pen to the top-axis so that the writing utensil is resting on the paper.
5. Demonstrate the working hardware and software by inputting the following and submit to Canvas:

Location	X (cm)	Y (cm)	Velocity (%)
1	5	0	100
2	0	5	50
3	-5	-5	20
4	7	2	60
5	-7	3	80
6	0	-5	10

6. Describe your approach, apparatus and code in your report and include a screenshot of your C# program and a picture of your piece of paper. Attach your MCU and C# code as appendices.
7. What are the advantages/disadvantages of using a stepper motor? What are the advantages/ disadvantages of using a DC motor? If you were designing a similar machine would you use stepper or DC motors and why? How would you implement a CW or CCW curve? What about a non-symmetrical curve? Discuss in your report.

Video Deliverables:

1. DC gantry stage moving **in order** to the 6 positions, noted above, while drawing on the paper with a pencil or marker. Note: **Taping** a pencil or marker in place should be sufficient.

Grading Scheme and Schedule

Exercise	Due Date	Points	Grading Scheme
1. DC motor control	Oct. 26	20	50% on completion and quality, 50% on report
2. Stepper motor control	Oct. 26	20	50% on completion and quality, 50% on report
3. Encoder reader	Nov. 2	20	50% on completion and quality, 50% on report
4. Closing the loop	Nov. 2	20	50% on completion and quality, 50% on report
5. 2-axis Control	Nov. 9	20	50% on completion and quality, 50% on report
Final report	Nov. 16		

Exercises must be submitted to Canvas by their due dates. Exercised submitted past the due date will incur a -5 point penalty for each one-week delay.

Gantry Assembly

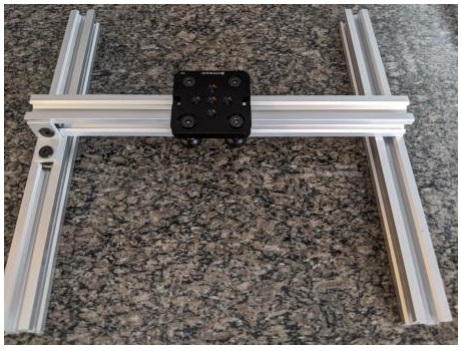
Safety

Please note there are a number of risks to be aware of while working with this device. Ensure the device is on a stable and level surface before operating. Keep fingers, hair, and loose clothing away from the device when it is operational. Keep your hands and any liquids away from the electronics while the circuit is live, only ever work on the circuit when it is **not powered** or **plugged into a power source**.

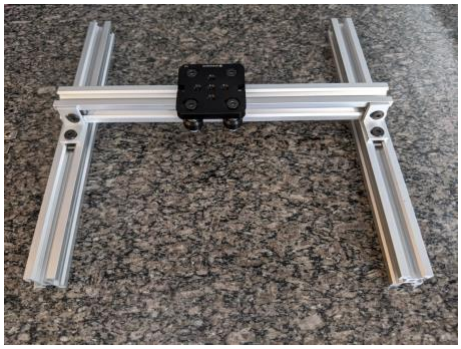
Step 1. Assemble the Frame

1.1 Gather the necessary components

1.2 Mount one side of the X-axis beam, put the slider on the extrusion



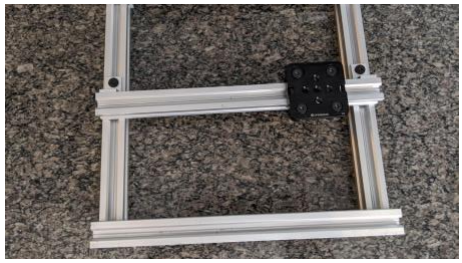
1.3 Mount the other side of the X-axis beam



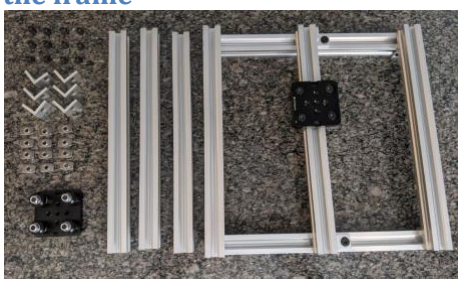
1.4 Flip the base, and place the other two extrusions as shown

1.5 Fix the four corners

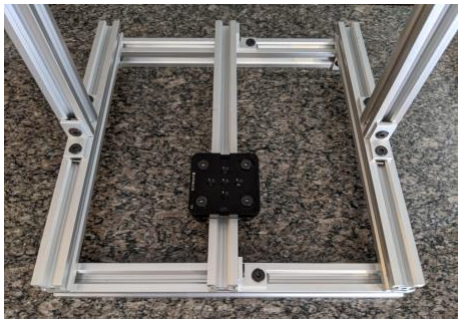
1.6 The base should now look like this



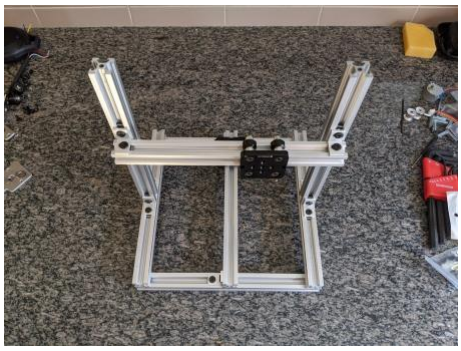
1.7 Gather the necessary components for the top of the frame



1.8 Mount the two support arms, front and back



1.9 Mount the Y-axis beam with the slider on the extrusion.



Step 2. Assemble the X-axis

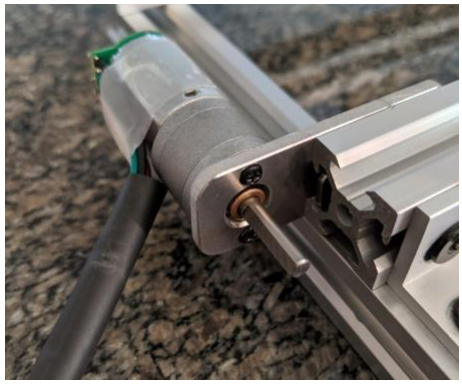
2.1 Gather necessary components

2.2 Mount the free pulley

2.3 Mount the DC motor mount

2.4 Gather DC motor parts

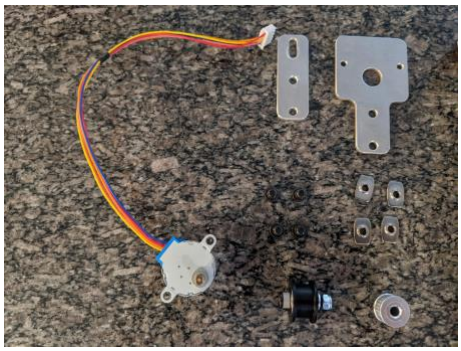
2.5 Mount the DC motor



2.6 Attach the 4mm bore gear

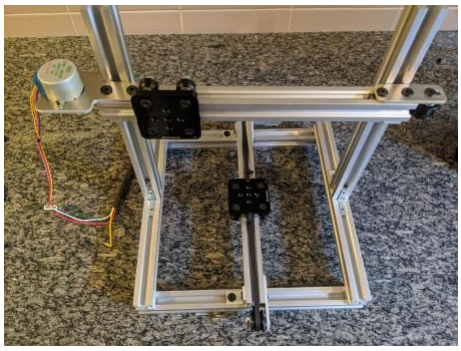
Step 3. Assemble the Y-axis

3.1 Gather necessary components



3.2 Mount the components similar to the X-axis.

Note: Stepper motor mounted with nuts and bolts



Step 4. Assembling the belt drive

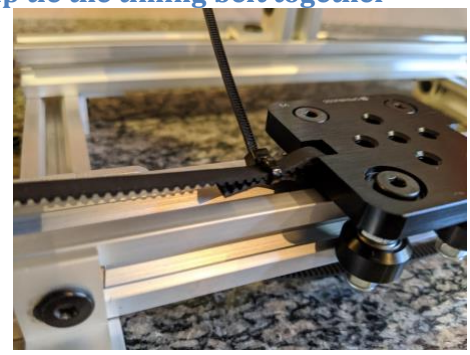
NOTE: Do not attach the belt until after you are

4.1 Feed belt through V-slot extrusion

4.2 Loop through slider

Note: Timing belt should lock with itself

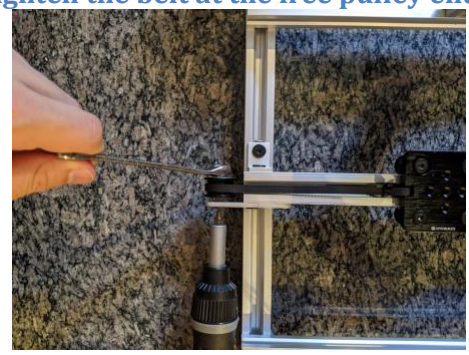
4.3 Zip tie the timing belt together



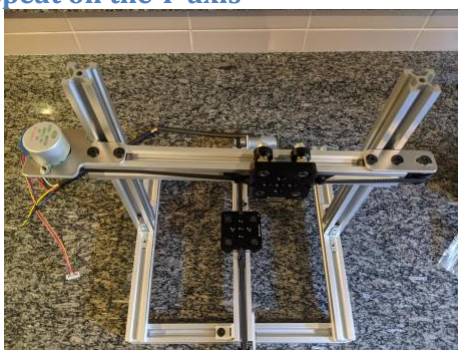
4.4 Zip tie the other end

Note: Timing belt should be looped around both the gear and free pulley.

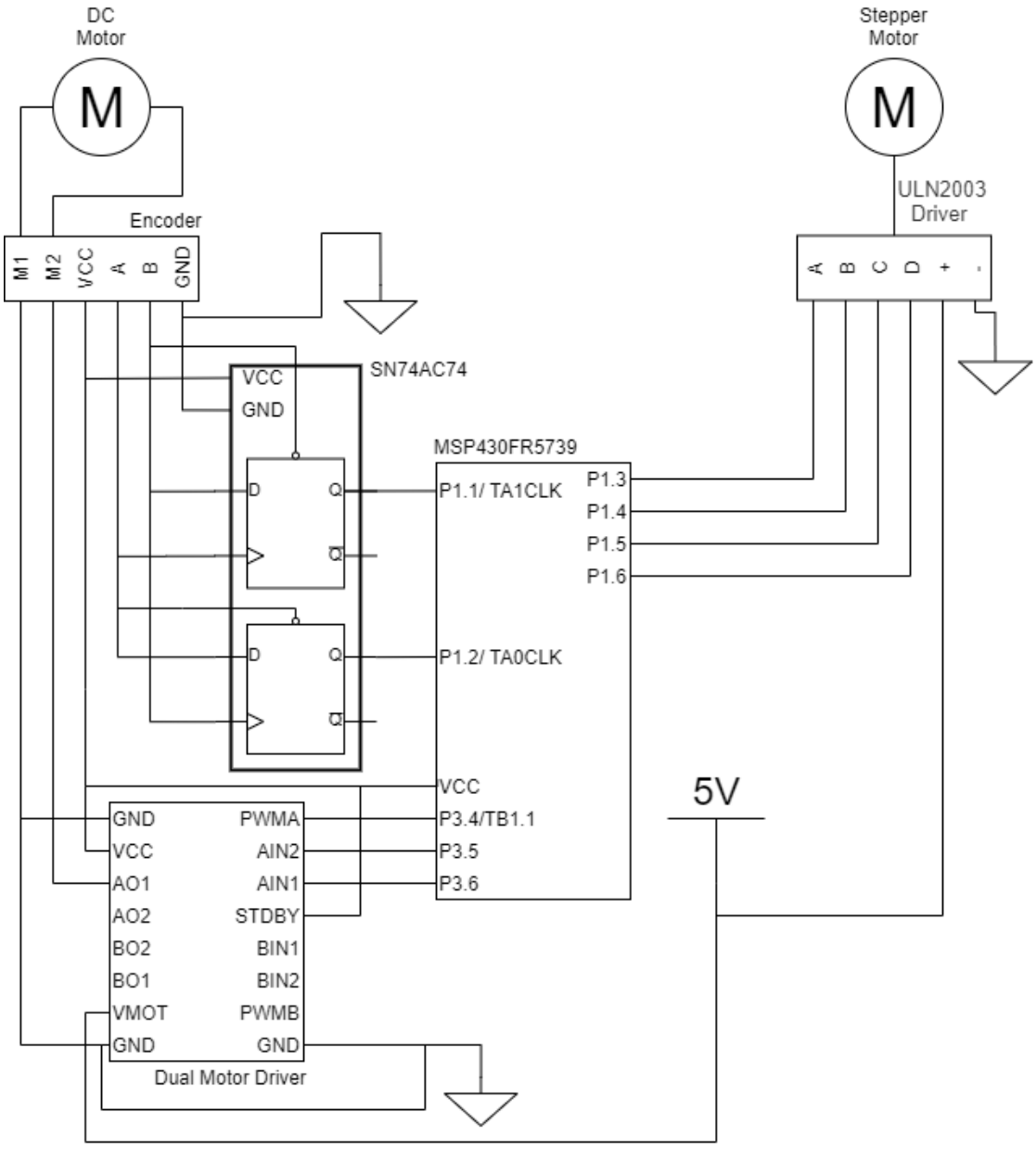
4.5 Tighten the belt at the free pulley end



4.6 Repeat on the Y-axis

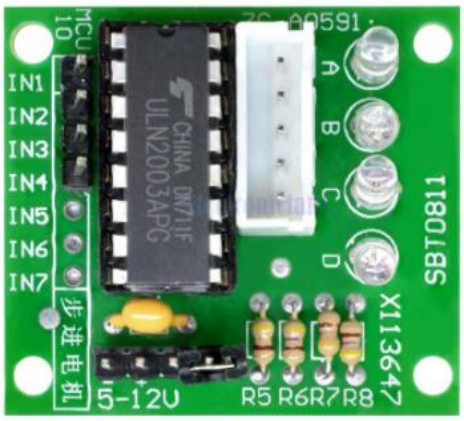


System Schematic



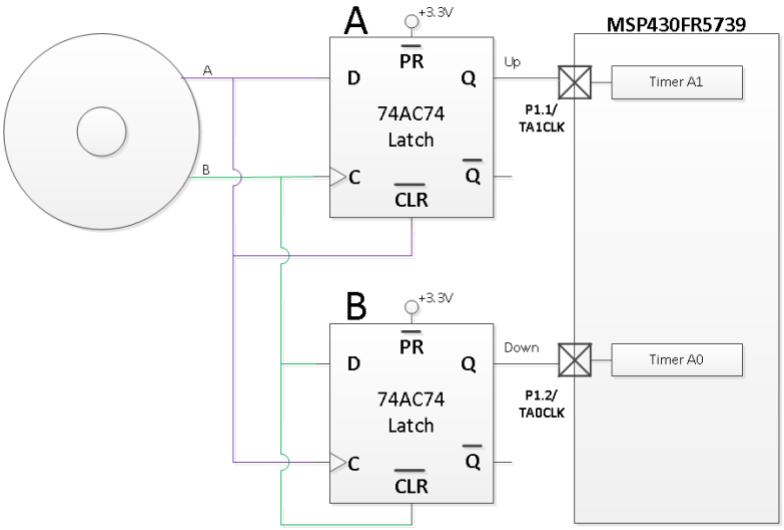
Notes:

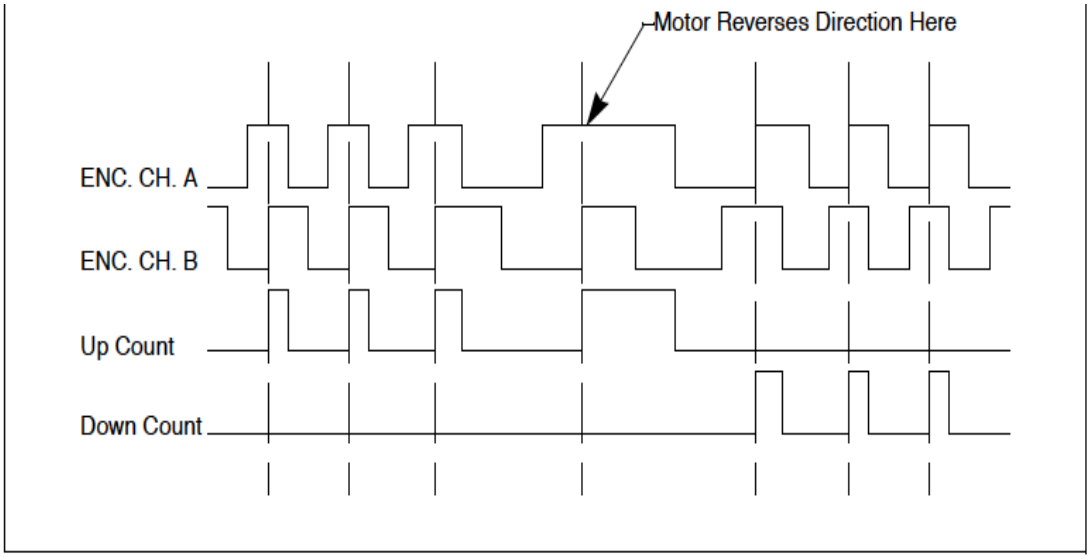
- The logic power (Vcc) is provided by the MSP430EXP board. The motors are powered from the 5-volt power supply. It is important that these are **not** mixed up.
- Circuits should be constructed on the supplied breadboard.
- Grounds are **common** and should all be **connected**. This is the most common problem students have every year.
- The stepper motor is driven as a unipolar stepper using the ULN2003 driver mounted on the following module:



Shaft Encoder Interface:

The decoder below will allow you to turn an A/B input from a motor encoder into up/down count pulses for the timer A clock. A simplified schematic is shown below along with the expected waveforms. (**Note: You only require one SN74AC74N chip as there are two latches in the microchip.)**





Components:

DC Motor: https://www.pololu.com/product/3712

Dual Motor Driver: https://www.pololu.com/product/713

Stepper: https://download.mikroe.com/documents/datasheets/step-motor-5v-28byj48-datasheet.pdf

Stepper Motor Driver: https://www.st.com/resource/en/datasheet/uln2001.pdf

Encoder: https://www.pololu.com/product/3081

Dual D-Type Flip Flop: https://www.ti.com/lit/ds/symlink/sn54ac74-sp.pdf?ts=1602870064687&ref_url=https%253A%252F%252Fwww.google.com%252F