MECH 423 –2024W
Lab #3: PCB Assembly and Motor Control

Objectives

The goal of this lab is to build and test a printed circuit board (PCB) for driving and controlling DC, stepper, and servomotors from a microprocessor and computer. There are six parts of this lab: 1) PCB Assembly and Soldering, 2) DC Motor Control, 2) Stepper Motor Control, 3) Encoder Reader, 4) Closed loop control and 6) Two-axis control.

Logistics and Evaluation

This lab is done in teams of two. Each team will be provided a bare PCB and the required components. Each student will be provided a lab kit consisting of all of the required components. After completing each exercise, students demo their work to TAs during the Monday lab sessions according to the schedule below. A written lab report is due after completion of the final lab demo. Work on assembling the PCB can begin as soon as the components are available. Students should use soldering equipment available in KAIS 1210.

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.

Lab Exercises

1. PCB Assembly and Soldering

1. Assemble the PCB according to the instructions and sequence at the end of this document.
2. As you complete each section, check the following:
    a. Power supply: show that 3.3V and 5V can be measured correctly
    b. USB interface: show that USB-serial chip is recognized by a PC
    c. Microprocessor: show that the MSP430FR5739 successfully programs. NOTE: The green board cannot be programmed thru the USB interface, only thru the 6-pin JTAG connector. You need to flash firmware to the red MSP while it is connected to the green board via the JTAG connection. This connection transfers the flashed code to the green board. Make sure your green board is powered (via USB or power supply).
    d. Motor driver: output high to the DRV8841 PWM pin and show that the DC motor turns
3. Demo your PCB to a TA. They have a device to check connectivity of all your microprocessor pins.

Demo Requirements:

- All microprocessor pins are connected - demonstrated using LED peripheral test board.

Report Requirements:

- Clear photo of your board, front and back.
- Brief comments on problems or challenges you encountered and what you did to resolve them if applicable.

2. 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 to a TA.
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.

Demo Requirements:

- Velocity control of stage using C# interface slider with continuous range from **max CCW** to **max CW**. Note: Should cross-over zero velocity position.

Report Requirements:

- Clear screenshot of your C# interface in operation.
- A diagram of your packet structure design (start byte, cmd bytes, data bytes, etc).
- Electrical schematic diagram as described above.
- Answer: 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.
- Answer: What is the minimum PWM duty cycle for the motor to turn at no-load? Why does this limit exist? Discuss.
- Brief comments on problems or challenges you encountered and what you did to resolve them, if applicable.

3. 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 to a TA.
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.

Demo Requirements:

- Demonstrate 5 single-steps in CCW and 5 single-steps in CW.
- Velocity control of stage using C# interface slider with continuous range from **max CCW** to **max CW**. Note: Should cross-over zero velocity position.

Report Requirements:

- Clear screenshot of your C# interface in operation.
- A diagram of your packet structure design (start byte, cmd bytes, data bytes, etc).
- Show your stepper state lookup table for half-step control.
- Electrical schematic diagram as described above.
- Answer: What is the maximum speed of this stepper motor (in steps/s and rev/s)? Why does this limit exist? Discuss.
- Brief comments on problems or challenges you encountered and what you did to resolve them, if applicable.

4. 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 to a TA.
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.

Demo Requirements:

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

Report Requirements:

- Clear screenshot of your C# interface in operation.
- Electrical schematic diagram as described above.
- Plot of encoder response from manually turning the motor shaft both directions.
- Documentation of your experiment using a PWM signal to turn the motor and characterize rotation rate vs duty cycle. Describe the experiment and results, showing graphs of test data.
- Brief comments on problems or challenges you encountered and what you did to resolve them, if applicable.

5. 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.
5. Measure properties of your closed-loop control system such as rise time, overshoot, and settling time.
6. Demonstrate the working hardware and software to a TA.
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.

Demo Requirements:

- DC gantry stage moving to 5 different locations, taking input from the C# program. Hint: Locations can be **absolute** or **relative** to current location.
- 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.
- Showing the TA your code demonstrating the use of the 32-bit hardware multiplier.

Report Requirements:

- Plot DC motor position response to step inputs of 0-25%, 0-50%, 0-100% duty cycle.
- Rise times for each step input magnitude. Can be displayed on the plot above, or in a table, or equivalent.
- Estimated transfer function for DC motor plant. Include the final expression, coefficient values, and a few sentences of justification for why you chose this form and how the coefficients were calculated.
- Block diagram for closed-loop position control loop including plant, controller, and feedback transfer functions and gains.
- Report your selected value for a proportional controller gain Kp and briefly describe how you selected it.
- (OPTIONAL, BONUS) Implement a full PID controller in firmware and demo. Compare PID performance to just P, and report your controller gains for Ki and Kd and briefly describe the selection process.
- Measure and report properties of your closed-loop control system such as rise time, overshoot, and settling time. Show a plot of the closed-loop step response.
    - Does this match your initially estimated transfer function reasonably? If not, derive a closed-loop transfer function that represents your experimental results.
- Brief comments on problems or challenges you encountered and what you did to resolve them, if applicable.

6. 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.

Demo Requirements:

- 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.

Report Requirements:

- Clear screenshot of your C# interface in operation.
- Clear photo of your final gantry-commanded shape, as defined in the table in the lab manual. Quality of the shape will be evaluated.
- Brief description of how you synchronized the travel of the stepper and DC motors to achieve straight lines.
- (OPTIONAL, BONUS) Photos of other, more complex shapes of your own design, if you want to be fancy.
- Answer: 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.
- Brief comments on problems or challenges you encountered and what you did to resolve them, if applicable.

Report Conclusion Requirements: Conclusion is **optional**, but some bonus points will be given for reflection.

- What is a brief summary of your key lessons learned from Lab 3?
- What were the parts of this lab that were most valuable or interesting to you?
- What did you enjoy the most?
- What mechatronics topics do you wish you could have done more of, or want to learn about in future?
- Other thoughtful reflections.

Report Appendix Requirements:

- Attach the text of all C and C# code, organized by exercise number.

Grading Scheme and Schedule

Exercise	Due Date	Points	Grading Scheme
1. PCB assembly	Oct. 21	10	85% on completion and quality, 15% on report
2. DC motor control	Oct. 28	15	50% on completion and quality, 50% on report
3. Stepper motor control	Oct. 28	15	50% on completion and quality, 50% on report
4. Encoder reader	Nov. 4	20	50% on completion and quality, 50% on report
5. Closing the loop	Nov. 4	20	50% on completion and quality, 50% on report
6. 2-axis Control	Nov. 15	20	50% on completion and quality, 50% on report
Final report	Nov. 18		

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

Instructions and Advice on PCB Assembly

Work on the PCB is divided into 6 sections that the student will complete in order. **Do not skip to the next section unless you have successfully tested each previous section.** Students will be graded for functionality of the PCB, as well as overall workmanship (*e.g.* cleanliness and appropriate amount of solder). Losing components provided by the lab or excessive damage to the PCB requiring replacement may incur a grade penalty.

How to Make Good Solder Joints

A good solder joint begins with getting the solder to "wet" on the metal surfaces of the component or PCB. The solder will wet when the metal surfaces are clean and free of oxidation. This can be accomplished by controlling the amount of heating and appropriate use of flux. KAIS 1210 provides standard leaded solder, for which the correct temperature setting should be 600-650 °F. Excessive heating of the iron can oxidize the iron tip, which prevents heat transfer from the iron to the solder or metal pad. Tip oxidation can be seen when the tip goes from shiny to dull. To prevent oxidation of the iron tip, the iron tip needs to be cleaned by wiping it on a wet sponge before each use. Occasionally, so much oxide has been built up that wiping cannot clean the tip. In this case, use flux or tinning compound to clean the tip. Flux is a liquid cleaning agent that removes oxide from metal surfaces. Flux is activated when heated by the iron. Because of its liquid form, flux also serves to improve thermal conduction between the iron tip and the metal surfaces. Many types of solders will include a small percentage of flux. However, additional liquid flux is often required for soldering surface mount components. Tinning compound is a solid paste used to keep the iron tip clean.

Soldering two metal surfaces together involves aligning the following items in the same location: the iron tip, the solder, surface #1, and surface #2. Since we only have two hands and that surfaces #1 and #2 are not fixtured together, we must separate this process into a multi-step procedure:

1. Holding the iron and the solder to surface #1 allowing the solder to wet. This step is called "tinning" the surface, which provides the necessary solder to make the joint.
2. Repeat the process and tin surface #2.
3. Hold the two surfaces together and use the iron to re-heat the tinned solder and joining them together.

How to Solder Surface-Mount Components

Surface-mount is the primary form-factor for electronic components. In fact, many semiconductor products do not exist in the traditional through-hole packages. Some good instructional material is available here: http://www.sparkfun.com/tutorials/category/2 and here: http://www.youtube.com/watch?v=3NN7UGWYmBY. The following is a general procedure for soldering surface-mount components:

1. Apply flux to the pads
2. Tin one (and only one) pad for the component on the PCB
3. Solder the pin on component associated with the tinned pad
4. If necessary, re-melt the solder and adjust the placement of the component to allow other pins to align with the remaining pads
5. Apply flux again to the remaining pads
6. Solder the remaining pins to the remaining pads
7. If there is any solder bridging the pins, first try applying flux and reheating. If that fails, use a solder-braid to wick away some of the solder.

How to Solder Wires Together

1. Strip the insulation and install heat-shrink tubing if desired
2. Tin the wire using solder. Use flux if available since the metal conductor is usually oxidized
3. Repeat 1 and 2 for the other wire
4. Push the two tinned wire tips together, melting the solder to join them together
5. After cooling place the heat-shrink to insulate

Components

Components List

Components for the Lab #3 PCB are listed below. After receiving your kit, first count the components to measure sure you have at least the quantity shown in the table. To help you identify the components, component markings are shown in the table below. Note that some components are used to replace others to reduce the overall parts count.

Digi-Key Part Number	Reference	Description	Source	Qty
609-3462-ND	5X1 HEADER PROGRAMMING	BERGSTIK II .100" SR STRAIGHT	Kit	1
PCE4211CT-ND	MOTOR BYPASS CAP	CAP ALUM 100UF 50V 20% SMD	Kit	2
399-7339-1-ND	MCU CORE BYPASS	CAP CER 0.47UF 16V 5% X7R 0603	Kit	1
445-2331-1-ND	0.01UF CHARGE PUMP CRITICAL	CAP CER 10000PF 100V 5% NP0 1206	Kit	1
399-1202-1-ND	51PF FILTER ENCODER	CAP CER 51PF 50V 5% NP0 1206	Kit	2
S7018-ND	20PIN MCU HEADER	CONN HEADER 20POS .100 STR 30AU	Kit	2
CP-202A-ND	DC WALL JACK	CONN POWER JACK 2.1MM PCB	Kit	1
H2959CT-ND	USB CONNECTOR	CONN RECEPT MINI USB2.0 5POS	Kit	1
SSA34-E3/61TGICT-ND	PSU DIODE	DIODE SCHOTTKY 40V 3A DO214AC	Kit	2
F2112CT-ND	POLYFUSE USB	FUSE PTC RESET 6V .50A 1206	Kit	1
296-1063-1-ND	LATCH	IC D-TYPE POS TRG DUAL 14-SOIC	Kit	1
296-4300-1-ND	INVERTER	IC HEX SCHMITT-TRIG INV 14-SOIC	Kit	2
NCP1117ST33T3GOSCT-ND	3.3V PSU	IC REG LDO 3.3V 1A SOT223	Kit	1
NCP1117ST50T3GOSCT-ND	5.0V PSU	IC REG LDO 5V 1A SOT223	Kit	1
768-1135-1-ND	USB CHIP (FT230XS)	IC USB SERIAL BASIC UART 16SSOP	Kit	1
160-1169-1-ND	GREEN LED	LED GREEN CLEAR 1206 SMD	Kit	6
CSRN2512FKR200CT-ND	CURRENT SENSE RESISTORS	RES 0.2 OHM 2W 1% 2512	Kit	2
P1.0KECT-ND	1.0 KOHM (REPLACE B)	RES 1.0K OHM 1/4W 5% 1206 SMD	Kit	3
P2.70KFCT-ND	2.7 KOHM ENCODER FILTER/PULLUP 1%	RES 2.7K OHM 1/4W 1% 1206 SMD	Kit	4
P27.0FCT-ND	27 OHM USB TERMINAL 1%	RES 27 OHM 1/4W 1% 1206 SMD	Kit	2
P30KECT-ND	30 KOHM DRIVER PULLUPS	RES 30K OHM 1/4W 5% 1206 SMD	Kit	2
P3.0KECT-ND	3.0 KOHM XBEE RESISTORS	RES 3K OHM 1/4W 5% 1206 SMD	Kit	2
P47KECT-ND	47 KOHM PROGRAMMING PULLUP	RES 47K OHM 1/4W 5% 1206 SMD	Kit	1
535-9364-1-ND	CRYSTAL	RESONATOR CER 24.00MHZ W/CAP SMD	Kit	1
ED1609-ND	SCREW TERMINAL	TERMINAL BLOCK 5.08MM VERT 2POS	Kit	4
445-1377-1-ND	BYPASS GENERAL	CAP CER 0.1UF 100V 10% X7R 1206	Lab	18
587-2994-1-ND	PSU BYPASS (REPLACE A)	CAP CER 4.7UF 50V 10% X7R 1206	Lab	8
P150GCT-ND	CURRENT LIMITING RESISTORS	RES 150 OHM 1/10W 5% 0603 SMD	Lab	42
609-3248-ND	4X1 HEADER (OPT)	CONN HEADER 4POS .100 STR 30AU	Optional	3
3M9405-ND	XBEE SOCKET (OPT)	CONN SOCKET 10POS 2MM VERT T/H	Optional	1
5001K-ND	TEST POINT (OPT)	TEST POINT PC MINI .040"D BLACK	Optional	10
587-3248-1-ND	PSU BYPASS	CAP CER 10UF 50V 20% X5R 1206	Replace A	0
P2.4KECT-ND	2.4 KOHM DRIVER LED (24VDC MAX)	RES 2.4K OHM 1/4W 5% 1206 SMD	Replace B	0
P330ECT-ND	330 OHM 3.3V LED POWER	RES 330 OHM 1/4W 5% 1206 SMD	Replace B	0
P510ECT-ND	510 OHM 5V LED POWER	RES 510 OHM 1/4W 5% 1206 SMD	Replace B	0
296-29235-1-ND	MCU	IC MCU 16BIT 16KB FRAM 38TSSOP	Kit	1
296-28914-1-ND	MOTOR CONTROLLER	IC MOTOR DRIVER PAR 28HTSSOP	Kit	1

Components Markings

Reference	Footprint	Markings	Qty
MCU CORE BYPASS	0603	None	1
0.01UF CHARGE PUMP CRITICAL	1206	None	1
50PF FILTER ENCODER	1206	None	2
PSU DIODE	DO-214AC	S34 49	2
POLYFUSE USB	1206	F	1
LATCH	SOIC-14	C74	1
INVERTER	SOIC-14	AC14	2
3.3V PSU	SOT-223	17L33	1
5.0V PSU	SOT-223	117-5	1
USB CHIP	SSOP-16	FT230XS	1
GREEN LED	1206	None	6
CURRENT SENSE RESISTORS	2512	R200	2
1.0 KOHM (REPLACE B)	1206	102	3
2.7 KOHM ENCODER FILTER/PULLUP 1%	1206	2701	4
27 OHM USB TERMINAL 1%	1206	27R0	2
30 KOHM DRIVER PULLUPS	1206	303	2
3.0 KOHM XBEE RESISTORS	1206	302	2
47 KOHM PROGRAMMING PULLUP	1206	473	1
CRYSTAL	3-SMD	24.0M	1

Parts Available in lab

The following parts will be stocked in KAIS 1210. Obtain these parts for your kit, but **do not take more than required**.

Digi-Key Part Number	Reference	Description	Qty
445-1377-1-ND	BYPASS GENERAL	CAP CER 0.1UF 100V 10% X7R 1206	18
587-2994-1-ND	PSU BYPASS (REPLACE A)	CAP CER 4.7UF 50V 10% X7R 1206	8
P150GCT-ND	CURRENT LIMITING RESISTORS	RES 150 OHM 1/10W 5% 0603 SMD	42

PCB Assembly Sequence

The Motor Control PCB to be built in this lab consists of the sections listed below. For each section, make sure you have all the necessary components before you begin. Next, solder the components in one section while ensuring there are no cold solder joints. Finally, test each section according to instructions before moving onto subsequent sections.

1. Underside resistors
2. Power supply
3. USB Interface
4. Microcontroller
5. Motor driver
6. Shaft encoder decoder

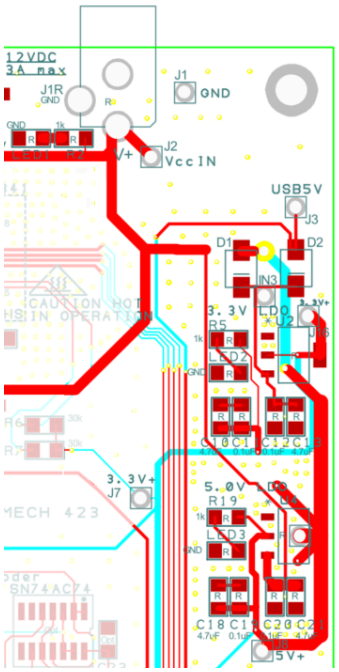
Read the schematic carefully to ensure that the component placed is the correct one. Failing to do so can cause damage. For several components (such as LEDs), direction matters. Create a list of components you need for each section before building it.

Full schematic and PCB drawings are available in a separate PDF file.

Specific Soldering Instructions

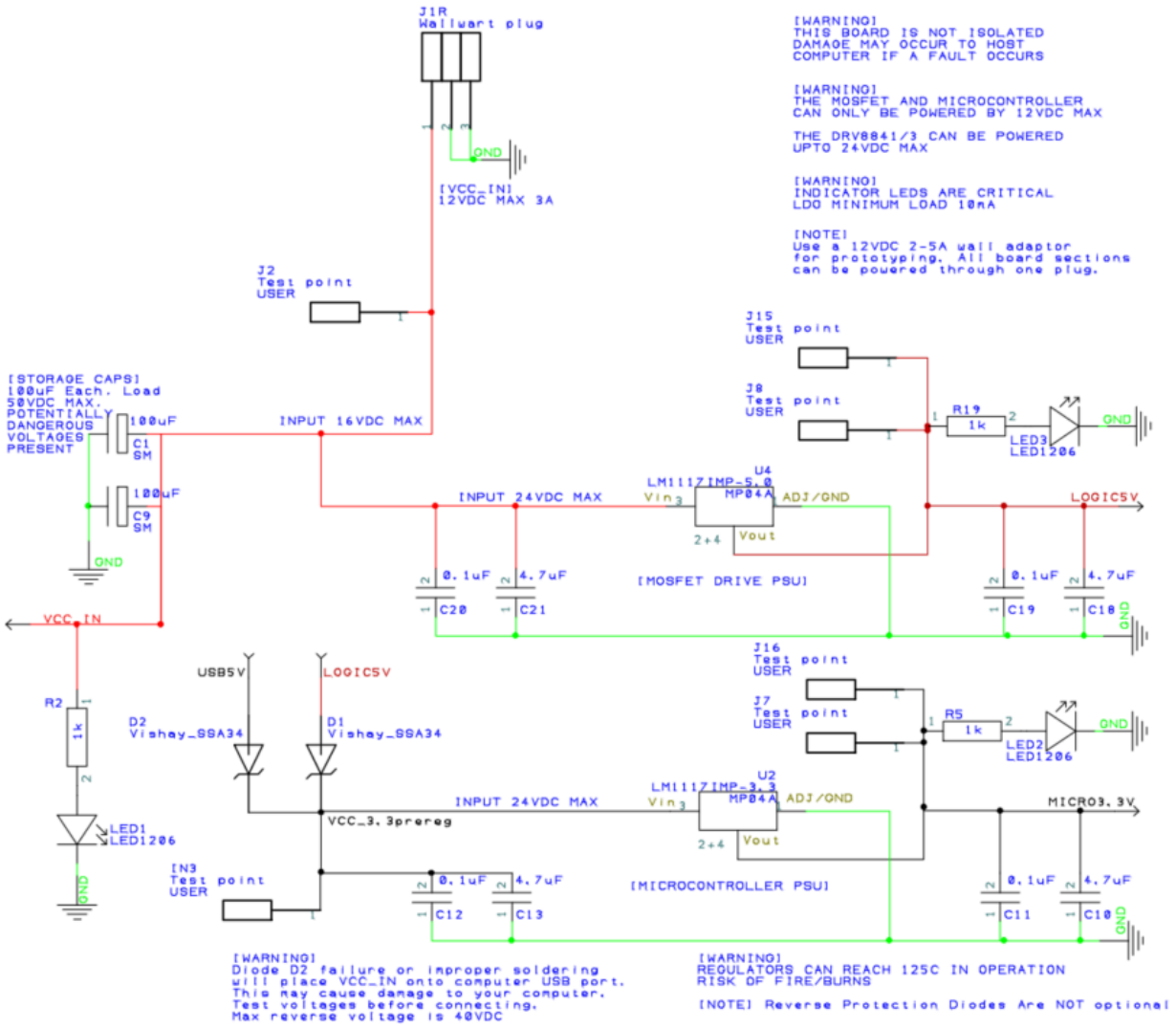
1. Underside Resistors

The only components on the bottom side of the PCB are 150 ohm resistors to limit the current going in and out of the microprocessor ports. Solder these components first as they provide a good practice.



2. Power Supply

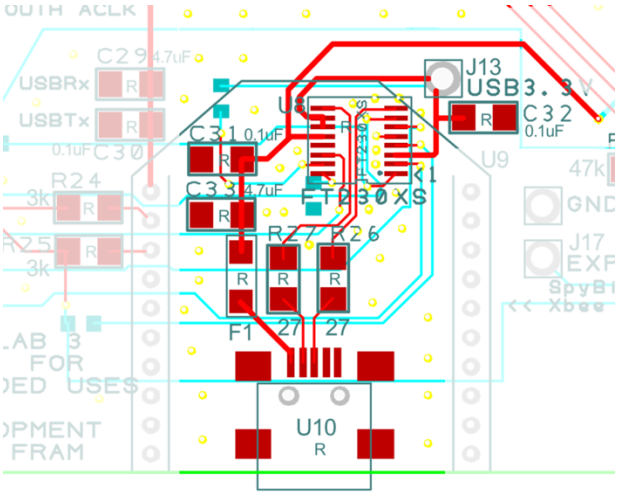
Assemble the power supply section using the components shown in the table. Refer to the PCB schematic and layout for more information. You need to solder the 3.3V and 5V linear voltage regulator and diodes for the DC input jack. This section can be self-tested if you can plug a 12VDC +center pin power brick into the board and have the 3.3V, 5V LED light, and PWR_DRV LED1 light.

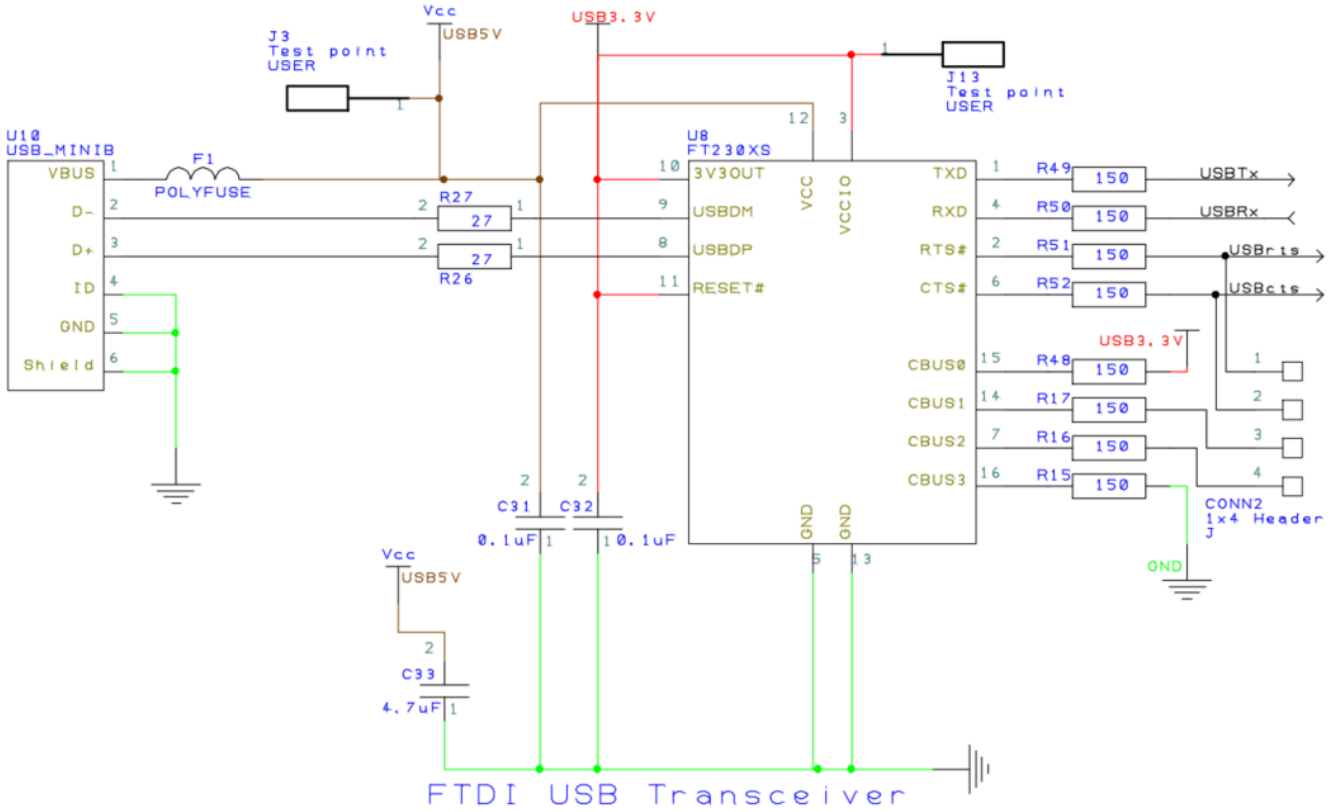


3. USB Interface

Assemble the USB interface using the following components. Check the reference material for information on the components. When you have completed this section, you should be able to detect the FTDI chip after plugging it into your computer. Warning: If your computer does not detect the chip, do not leave the board connected. **If the chip gets hot then remove the power immediately.**

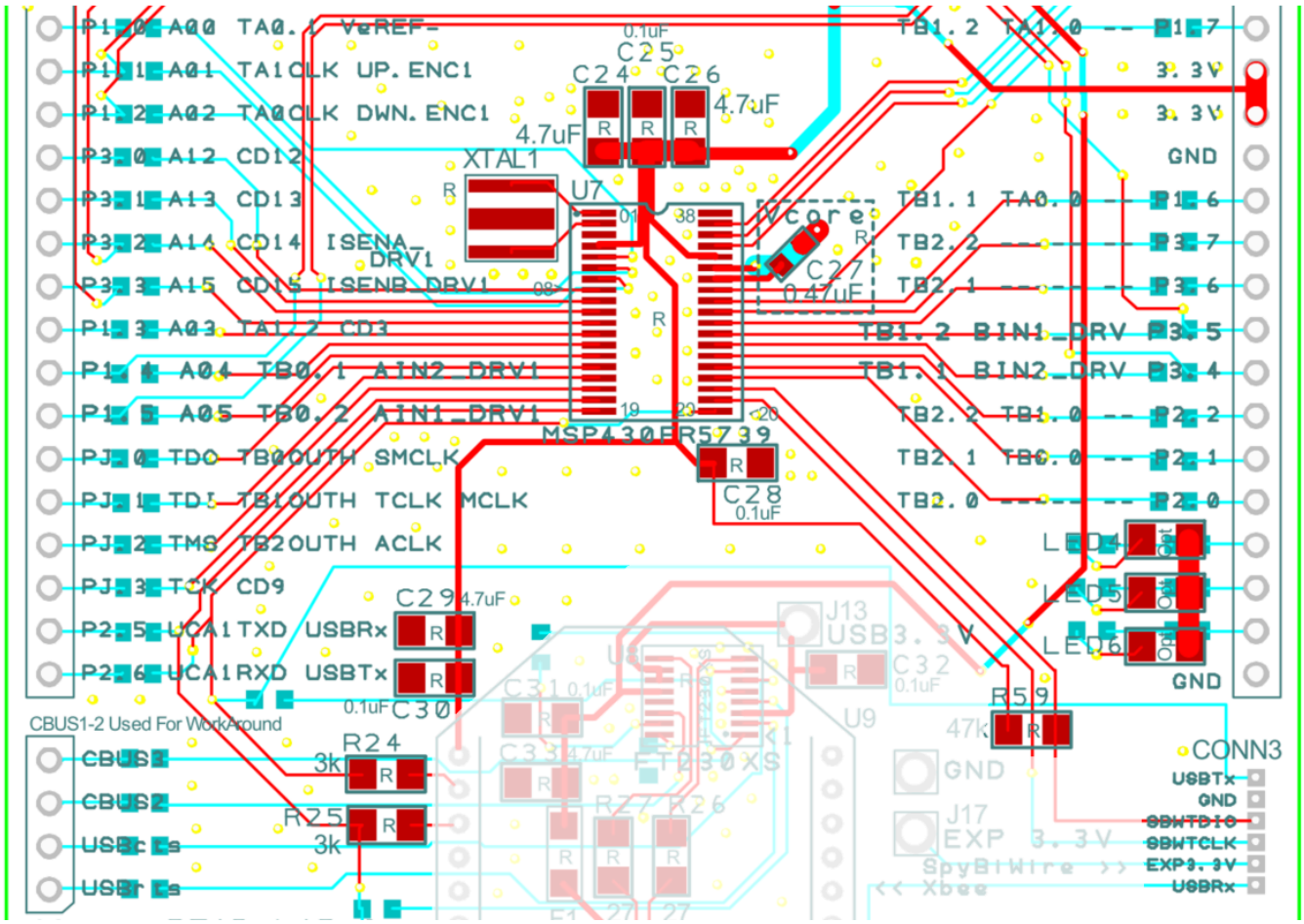
WARNING: Before connecting, ensure that you have not shorted the USB pins or the FTDI chip pins. If you are unsure, connect to an external USB hub to ensure your computer is protected.





4. Microcontroller

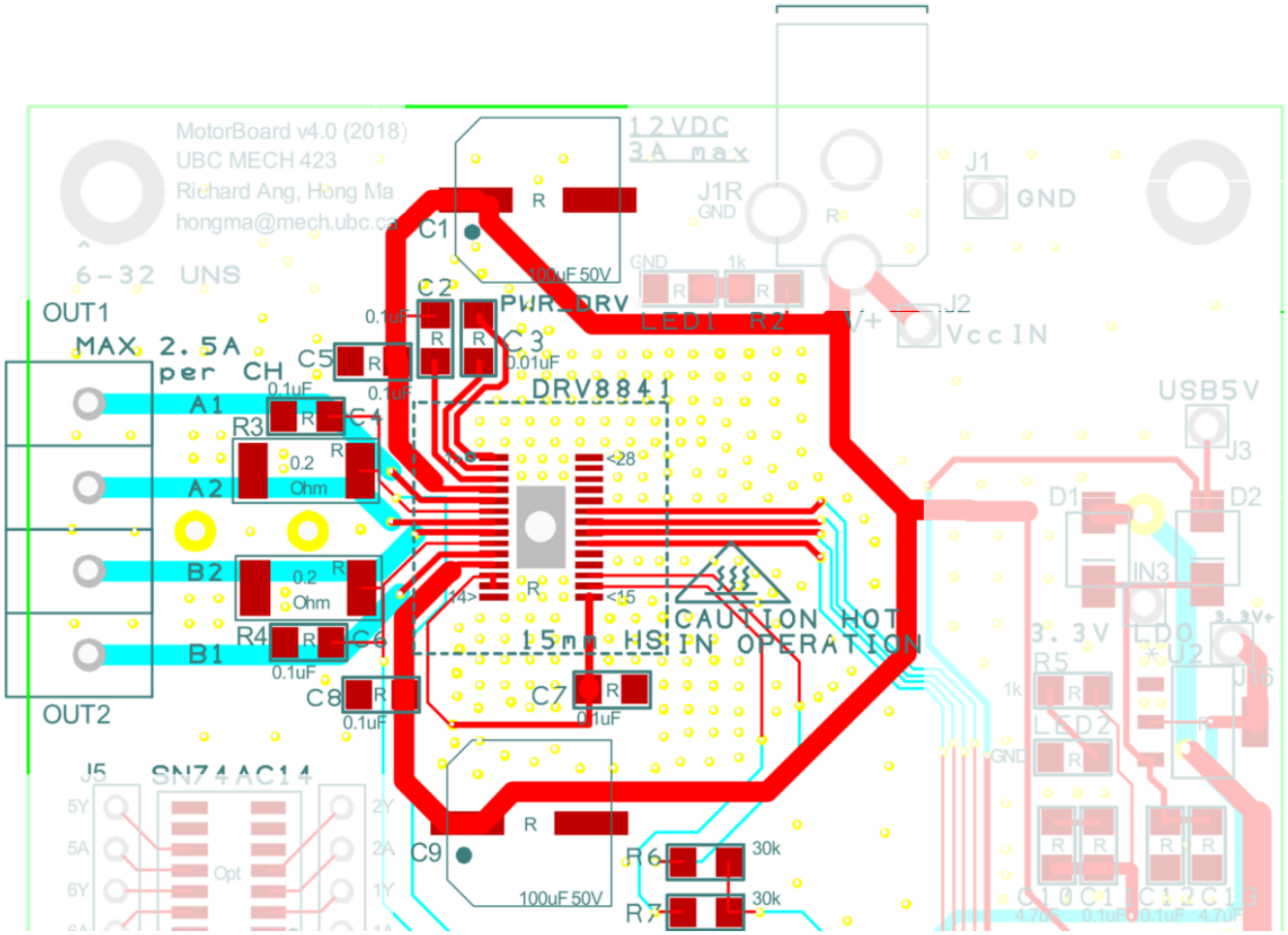
Assemble the microcontroller using the following components. Use the reference material for more information. You have completed this section once you can program and debug a program running on the MSP430FR5739 chip. **If the chip gets hot then remove the power immediately.**



Refer to design document for schematic.

5. Motor Driver

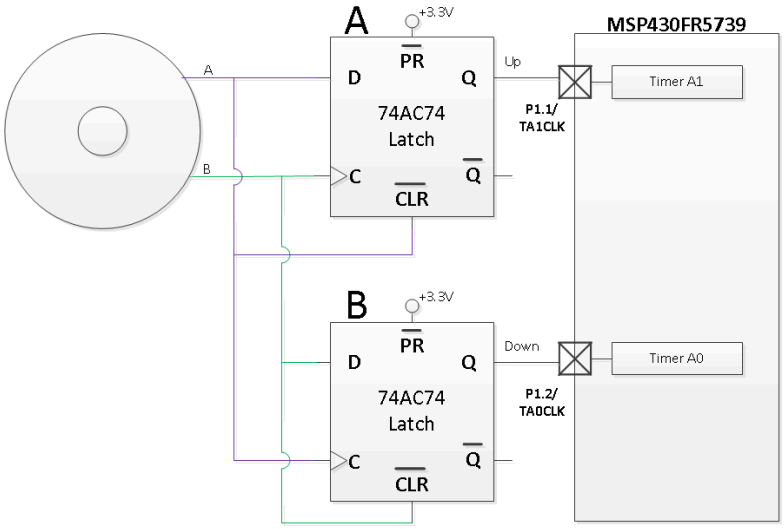
The motor controller can be tested by providing a PWM signal to the drive inputs and connecting a motor. After soldering the motor driver chip flip the board over and solder the hole on the back of the chip. The C3 capacitor is special and very important to the proper operation of the chip.

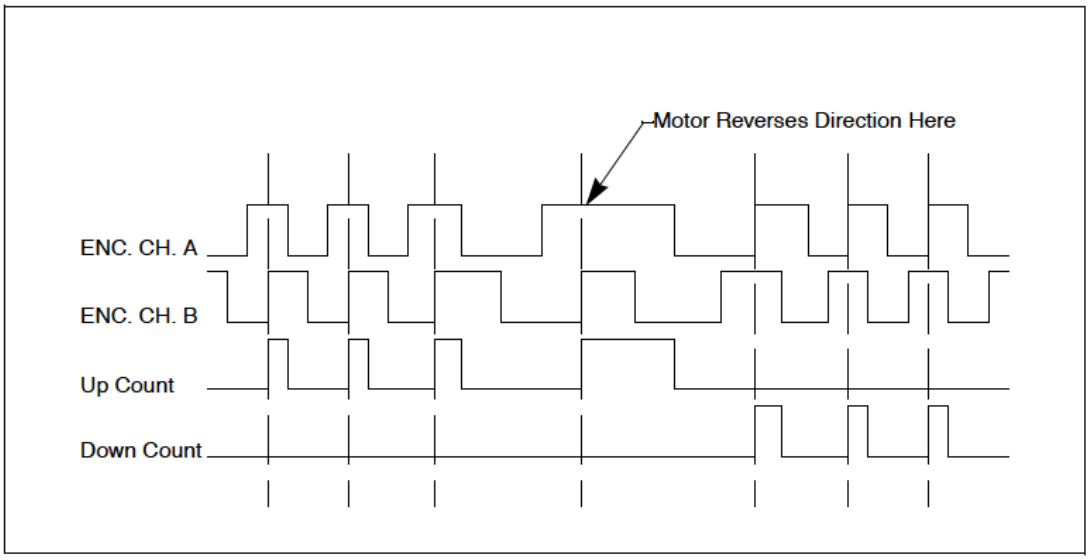


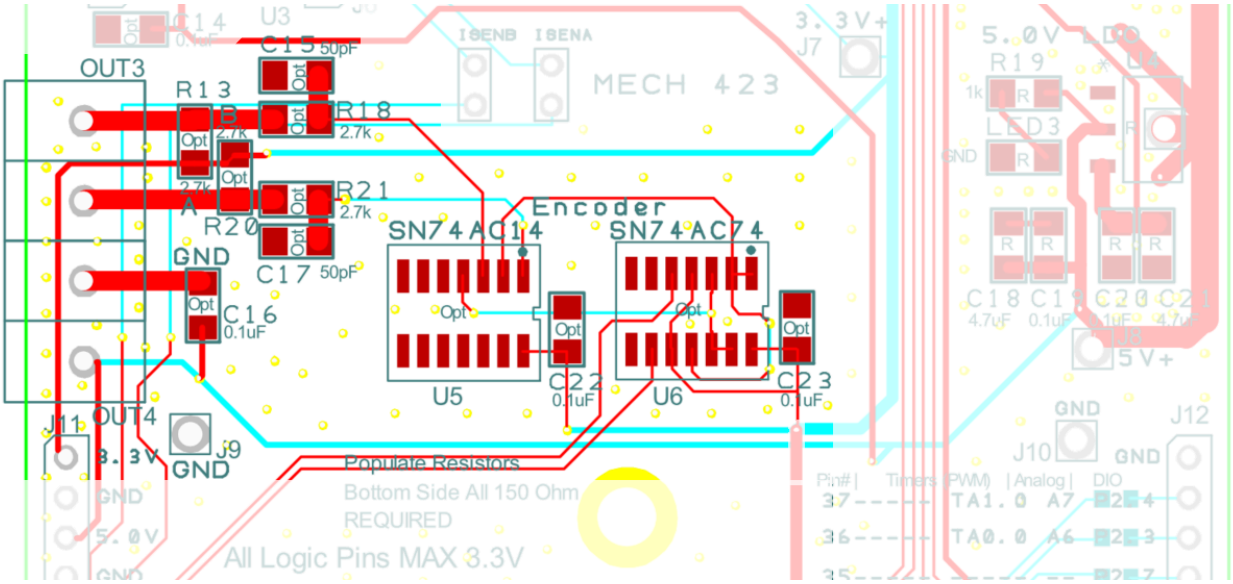
Refer to design document for schematic.

6. 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. The detailed schematic, available in the design document, contains an additional Schmidt trigger inverter to reduce spurious counts caused by electrical noise.







Refer to design document for the detailed schematic.

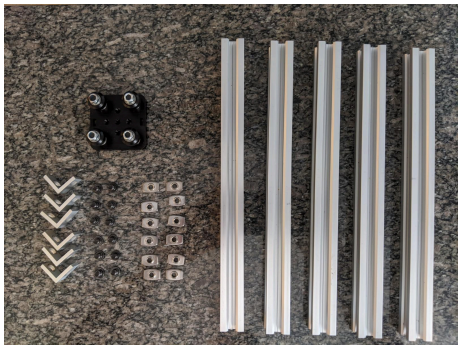
Gantry Assembly

Safety

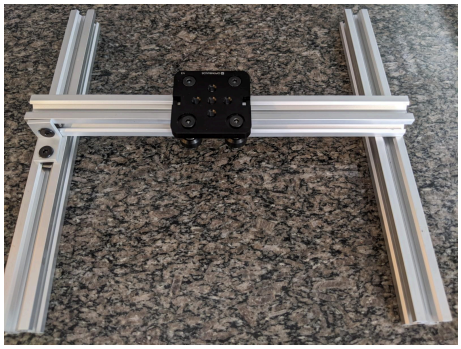
Please note there are a number of risks to be aware of while working with this device. Ensure the device is on a stable and level surface before operating. Keep fingers, hair, and loose clothing away from the device when it is operational. Keep your hands and any liquids away from the electronics while 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

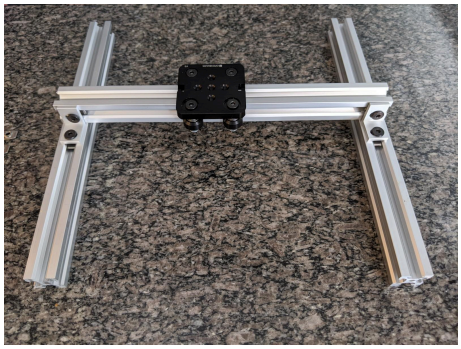
1.1 Gather the necessary components



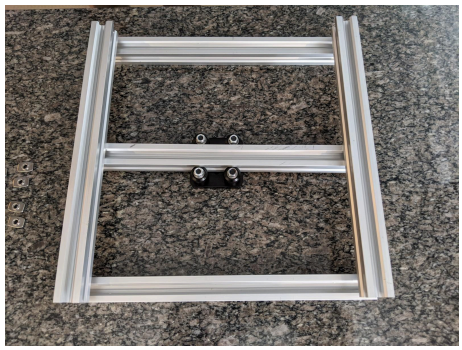
1.2 Mount one side of the X-axis beam, put the slider on the extrusion



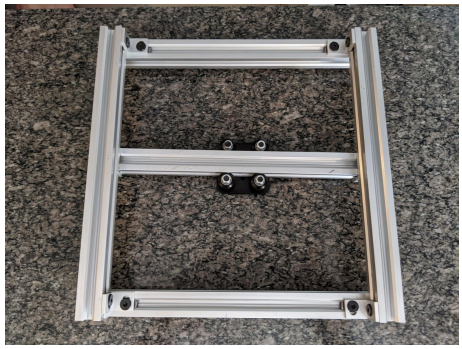
1.3 Mount the other side of the X-axis beam



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

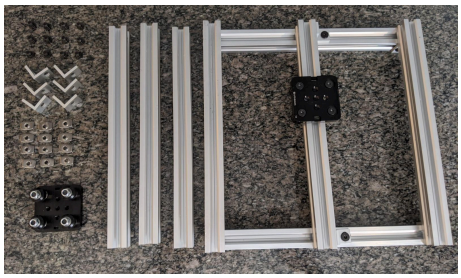


1.5 Fix the four corners

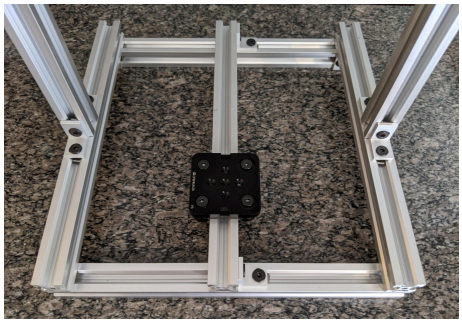


1.6 The base should now look like this

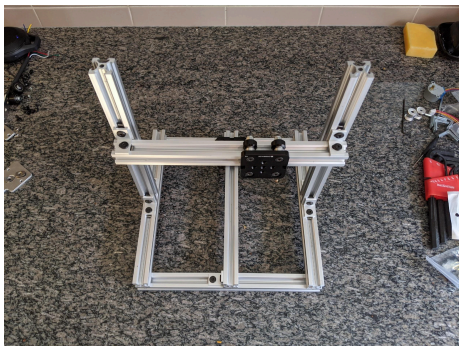
1.7 Gather the necessary components for the top of the frame



1.8 Mount the two support arms, front and back



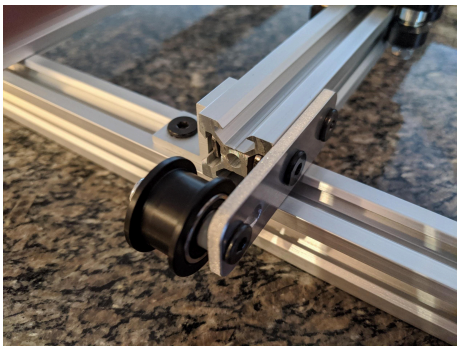
1.9 Mount the Y-axis beam with the slider on the extrusion.



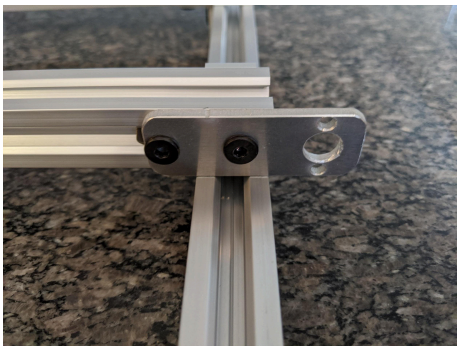
Step 2. Assemble the X-axis

2.1 Gather necessary components

2.2 Mount the free pulley

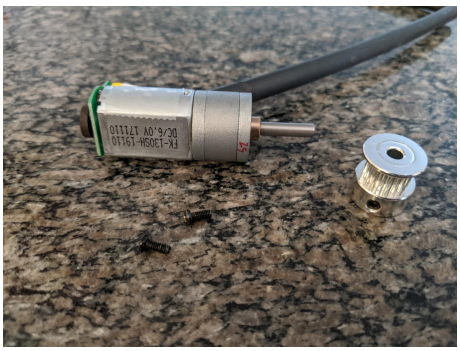


2.3 Mount the DC motor mount

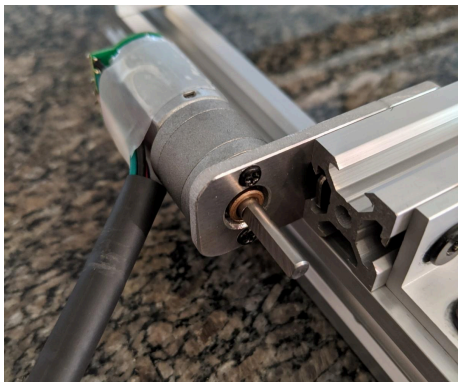


2.4 Gather DC motor parts

Note: DC motor in photo has been upgraded.



2.5 Mount the DC motor

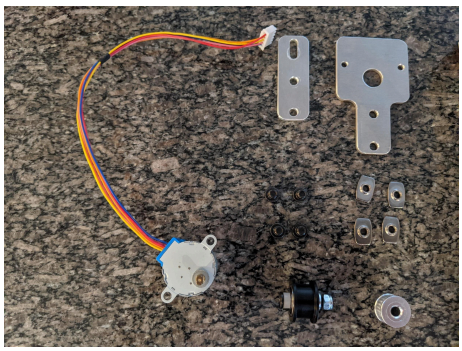


2.6 Attach the 4mm bore gear

Step 3. Assemble the Y-axis

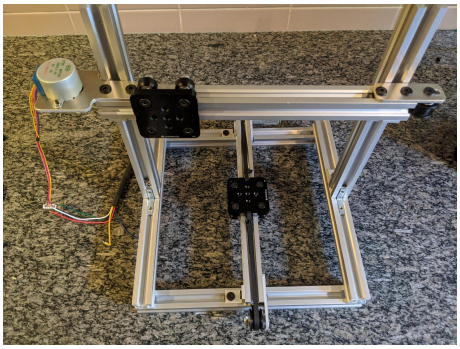
3.1 Gather necessary components

Note: Stepper motor in photo has been upgraded.



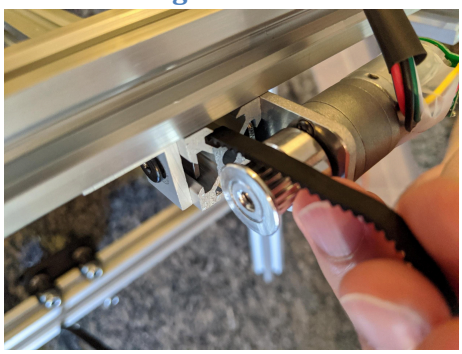
3.2 Mount the components similar to the X-axis.

Note: Stepper motor mounted with nuts and bolts.



Step 4. Assembling the belt drive

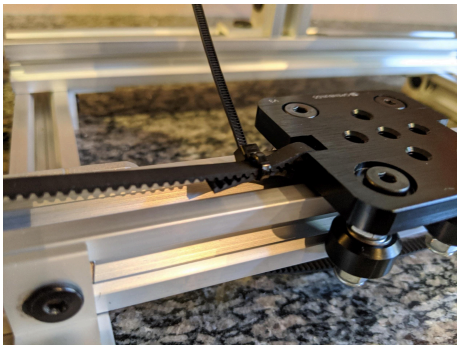
4.1 Feed belt through V-slot extrusion



4.2 Loop through slider

Note: Timing belt should lock with itself.

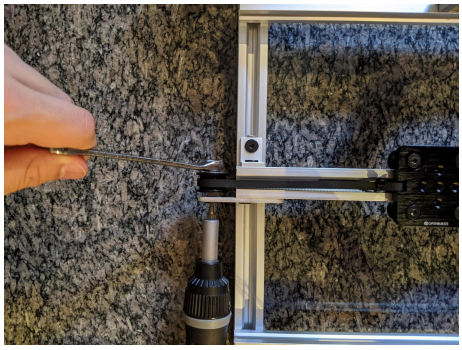
4.3 Zip tie the timing belt together



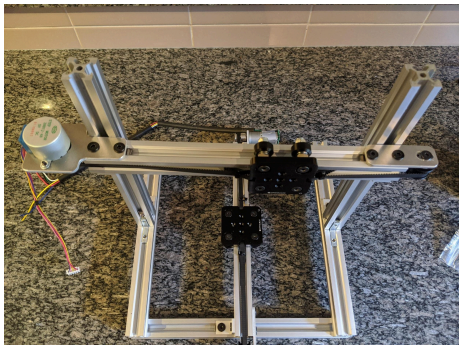
4.4 Zip tie the other end

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

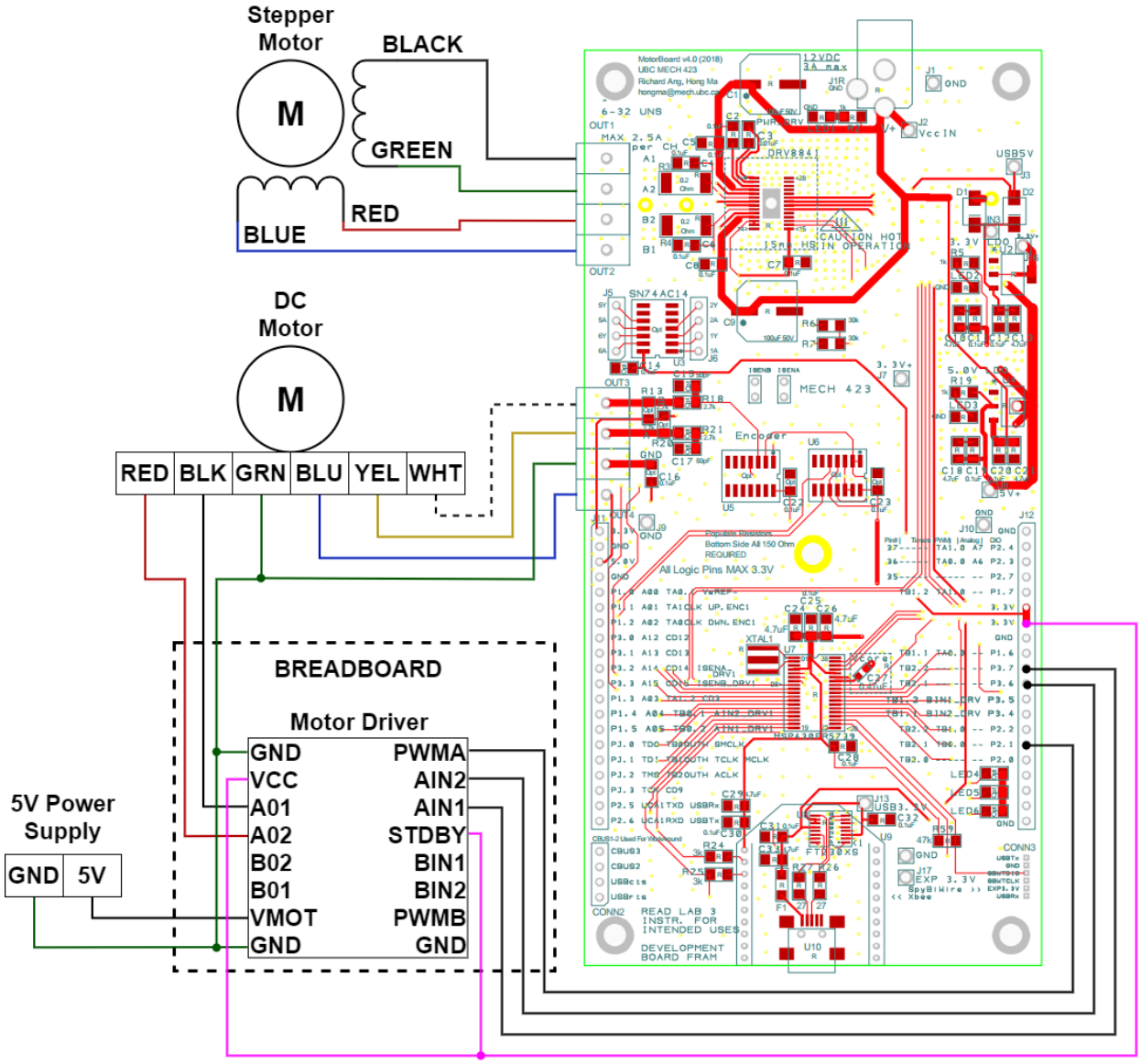
4.5 Tighten the belt at the free pulley end



4.6 Repeat on the Y-axis



System Schematic



Notes:

- The logic power (Vcc) is provided by the MSP430EXP board. The DC motor is 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.

Components:

DC Motor: https://www.pololu.com/product/4823

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

Stepper: https://www.pololu.com/product/1208