

MECH 421 – Mechatronic System Instrumentation

MECH 423 – Mechatronic Product Design

Lab #5: PCB Assembly and Stepper Motor Control

Objectives

The goal of this lab is to build and test a printed circuit board (PCB) for driving and controlling stepper, DC, and servomotors from a microprocessor and computer. There are three parts of this lab: 1) PCB Assembly and Soldering, 2) Stepper Motor Control, and 3) Two-axis Stage 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 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 #5. 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 unacceptable.

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. Write firmware to increase UART communication baud rate to 115.2 kbps. The higher baud rate will be necessary data acquisition in upcoming exercises. Demo the higher baud rate to the TA's to verify functionality. You may hardcode a message of your choice in firmware and transmit it over UART periodically.
 - e. 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 the LED peripheral test board.
- Serial communication using from board to laptop at 115.2 kbps. Demonstrate using Putty, MECH423 serial communicator, or any other serial program.

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. Stepper Motor Control

1. Follow the instructions at the end of this document to assemble the gantry system.
2. **To avoid crashing, the belt connecting motor to the pulley system should not be installed during development.**
3. 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.
 - a. NOTE: Output to the motor should be modulated down to 25% to prevent overheating. Please ask the TA for clarification if required.
4. Write microprocessor code to command the stepper motor to take a single half-step CW or CCW from a UART message.
5. Write microprocessor code to run the stepper motor at a constant speed with the speed commanded from a message from the UART. Note that the physical speed of the stepper motor should scale linearly with the velocity command. Measure the maximum no-load speed of the stepper motor.
6. In C#, set up a GUI for triggering single step control in both directions. Set up a 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.
7. **Connect the stepper motor pulley system. Measure the maximum loaded speed of the motor.**
8. Demonstrate the working hardware and software to a TA.
9. Describe your apparatus and code in your report and include a screenshot of your C# program. Attach your MCU and C# code as appendices.
10. Draw an electrical schematic diagram of the minimum components necessary to drive the stepper motor including the MCU, motor driver, power supply, and accessories.

Demo Requirements:

- Single step control in CW and CCW directions.
- Continuous velocity control of stage using C# interface slider from **max CCW** to **max CW**. Note: The slider 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.
- 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.

3. 2-Axis Control with Dual Stepper Motors

1. Wire a 2nd stepper motor to the external H-bridge motor driver. One stepper motor phase should be connected to AOUT, and one to BOUT. A diagram of the new recommended wiring scheme is available at the end of the document.
2. Write firmware code to drive the 2nd stepper motor using this external motor driver. Hint: write one function for each motor with an identical function call interface to abstract the details of the stepping implementation.
3. Write microcontroller code to receive a package of bytes that control both the x and y axis. (**Hint: Think of G-code.**)
4. 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)
5. 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.
6. Demonstrate the working hardware and software by inputting the following and demonstrating to a TA:

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

7. Come up with a new shape of your own design, such as a circle, a Bezier curve, a city skyline, etc. Use your creativity. Points will be awarded for complexity of the final design. Plot your creative shape. Compare the design and the final result.

Demo Requirements:

- Gantry stage moving 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.
- Gantry stage drawing your creative shape.

Report Requirements:

- Description of your apparatus and code algorithm. Attach your MCU and C# code as appendices.
- Screenshot of your C# interface in operation.
- A photo of your the shape drawn in step 6.
- A photo of your creative shape with the design overlaid. Explain what caused the deviations between design and stage position.
- What are the advantages and limitations of using stepper motor for 2-axis stage control? Which shapes are easy and which are hard to draw?
- Brief comments on problems or challenges you encountered and what you did to resolve them, if applicable.

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	Nov. 21	10	85% on completion and quality, 15% on report
2. Stepper motor control	Nov. 26	15	50% on completion and quality, 50% on report
3. 2-axis Control	Dec. 1	25	50% on completion and quality, 50% on report
Final report	Dec. 5		

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

Appendix A: 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, add 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% NPO 1206	Kit	1
399-1202-1-ND	51PF FILTER ENCODER	CAP CER 51PF 50V 5% NPO 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
P330EECT-ND	330 OHM 3.3V LED POWER	RES 330 OHM 1/4W 5% 1206 SMD	Replace B	0
P510EECT-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.

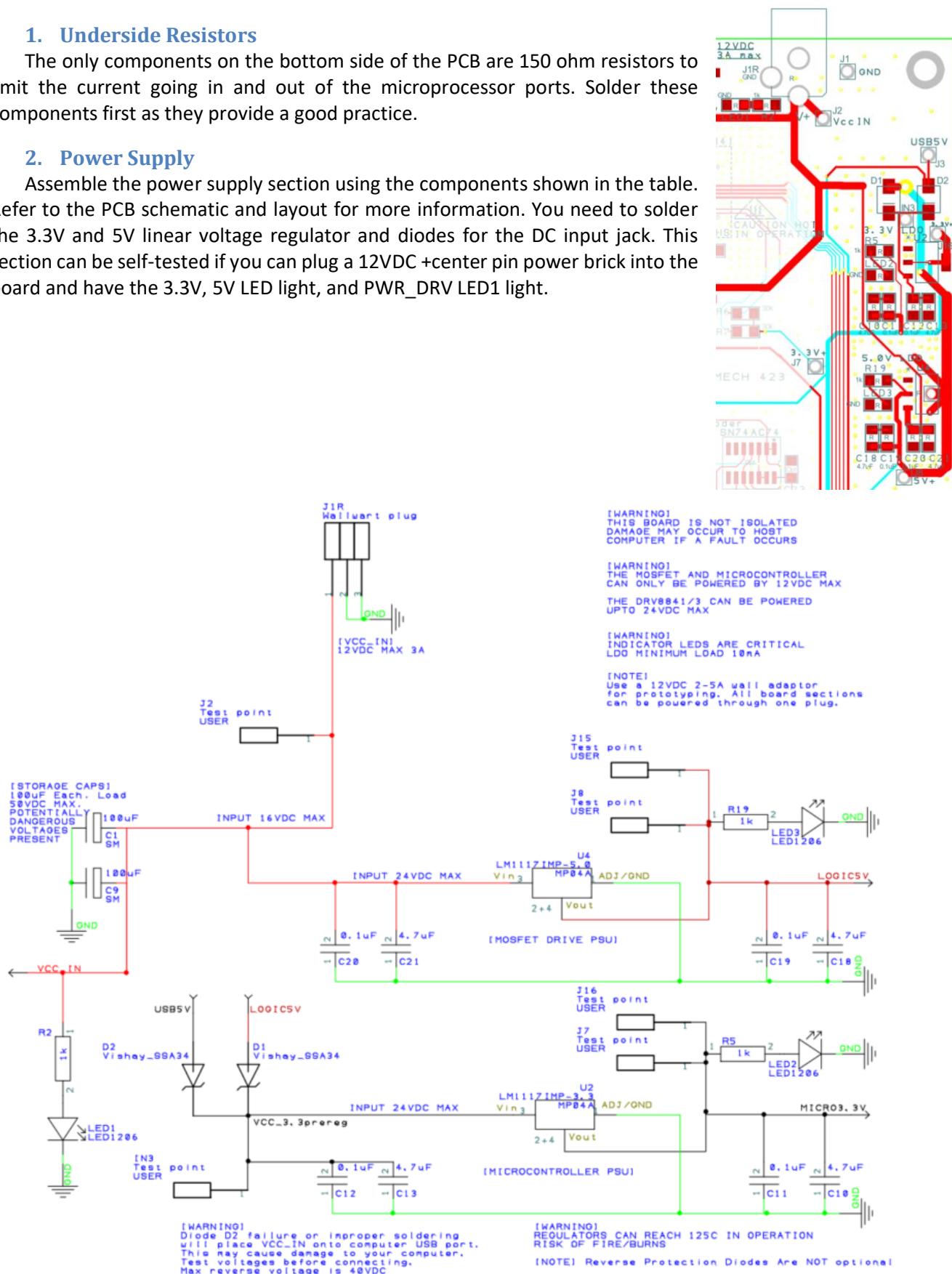
Appendix B: Specific Soldering Instructions

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

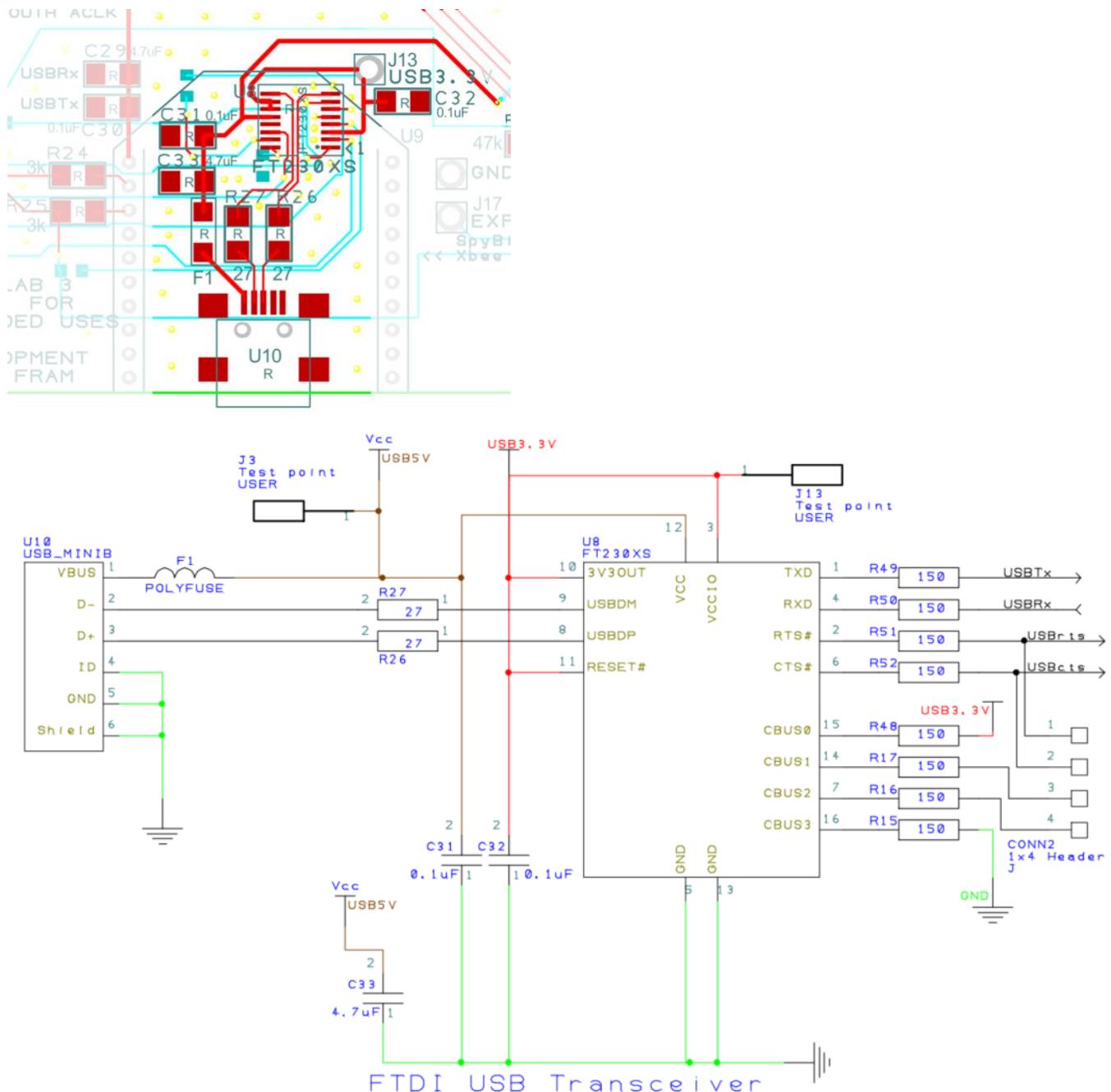
Assemble the power supply section using the components shown in the table. Refer to the PCB schematic and layout for more information. You need to solder the 3.3V and 5V linear voltage regulator and diodes for the DC input jack. This section can be self-tested if you can plug a 12VDC +center pin power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



3. USB Interface

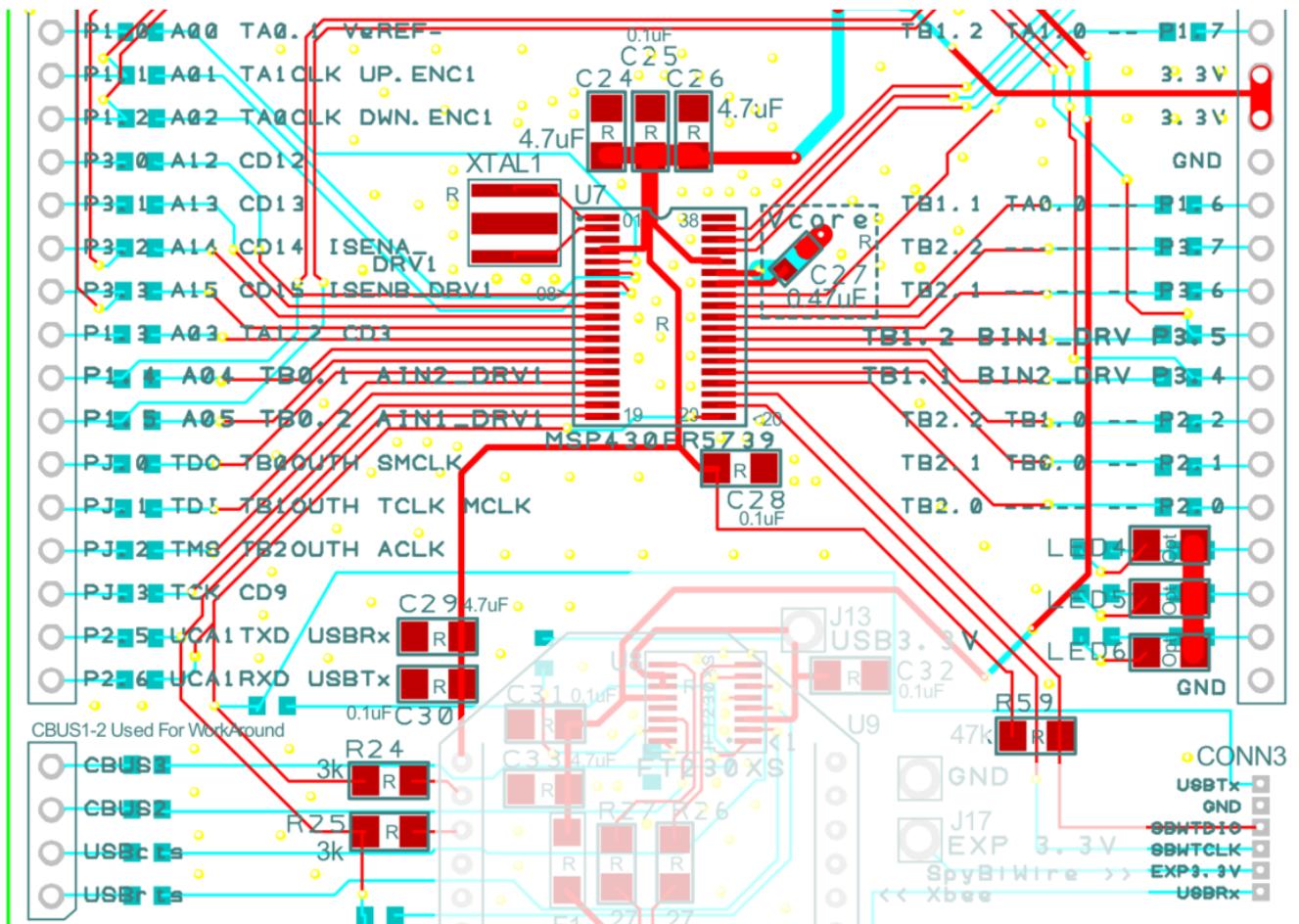
Assemble the USB interface using the following components. Check the reference material for information on the components. When you have completed this section, you should be able to detect the FTDI chip after plugging it into your computer. Warning: If your computer does not detect the chip, do not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

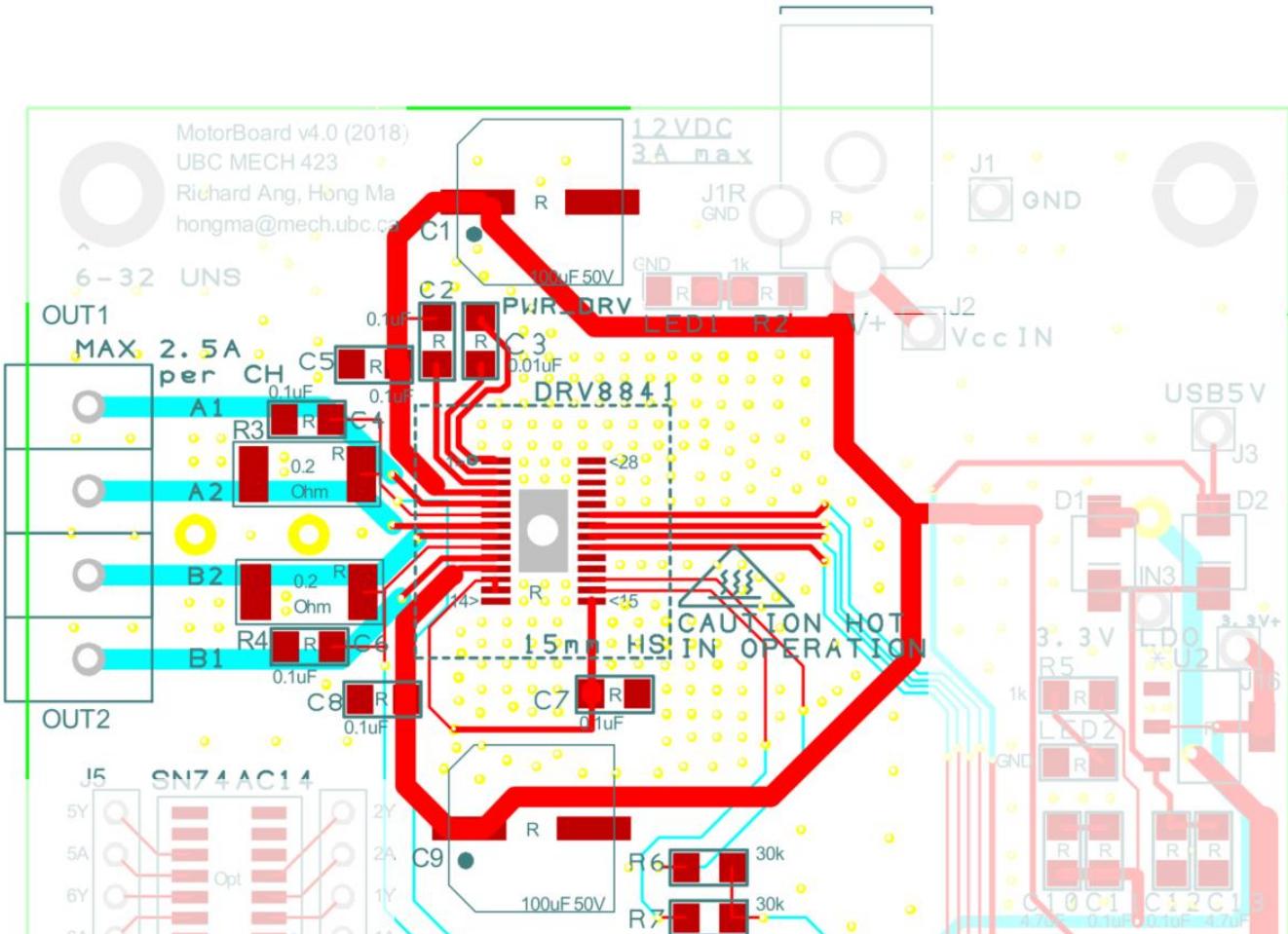
Assemble the microcontroller using the following components. Use the reference material for more information. You have completed this section once you can program and debug a program running on the MSP430FR5739 chip. **If the chip gets hot then remove the power immediately.**



Refer to design document for schematic.

5. Motor Driver

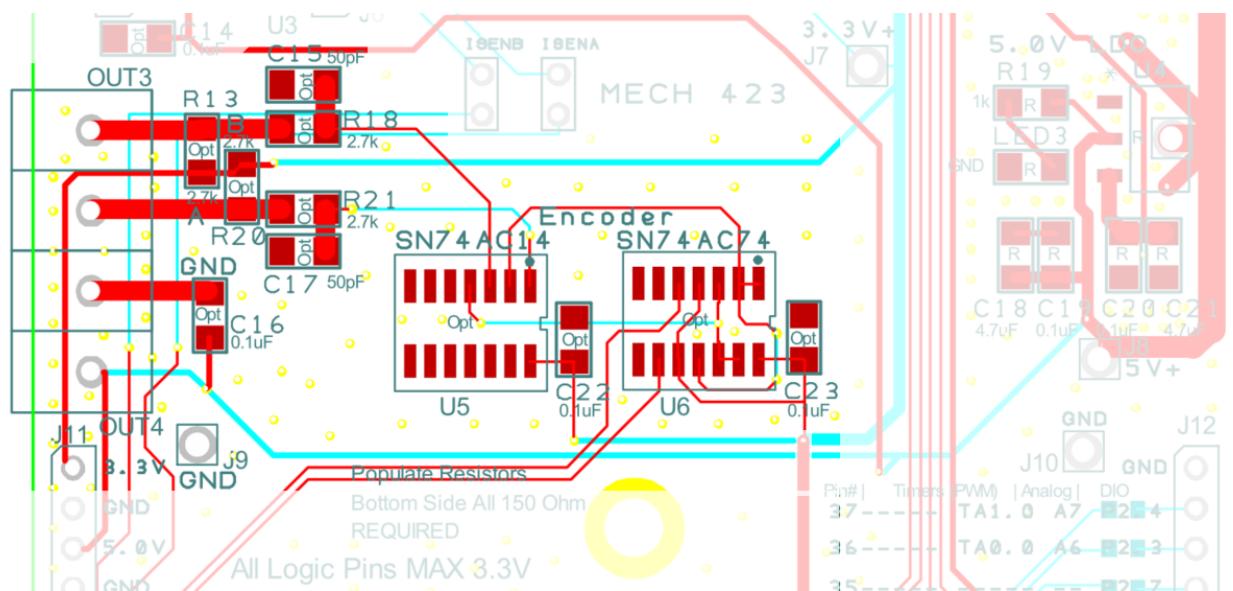
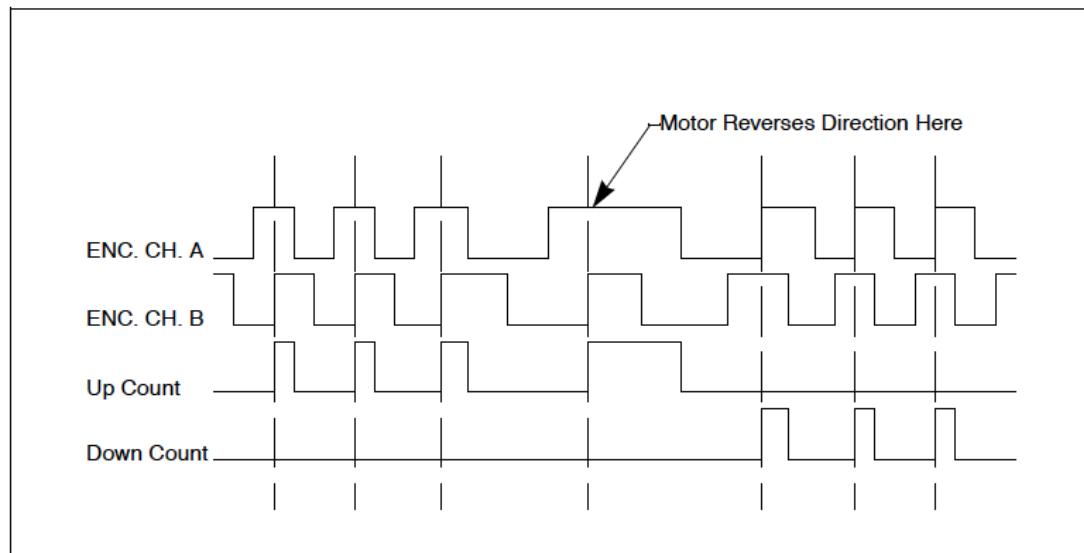
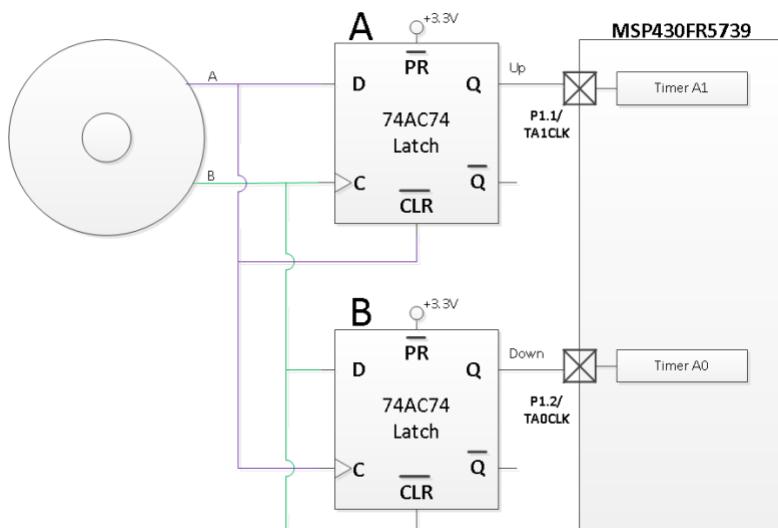
The motor controller can be tested by providing a PWM signal to the drive inputs and connecting a motor. After soldering the motor driver chip flip the board over and solder the hole on the back of the chip. The C3 capacitor is special and very important to the proper operation of the chip.



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.

Appendix C: 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



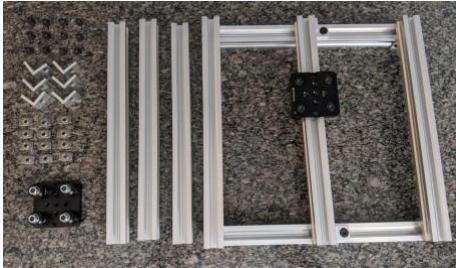
1.5 Fix the four corners



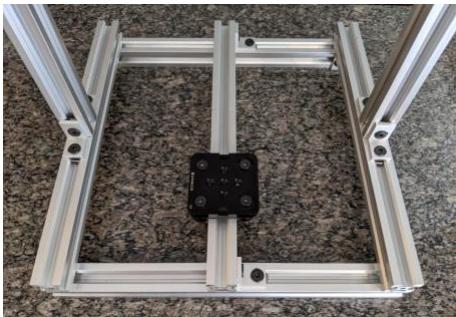
1.6 The base should now look like this



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



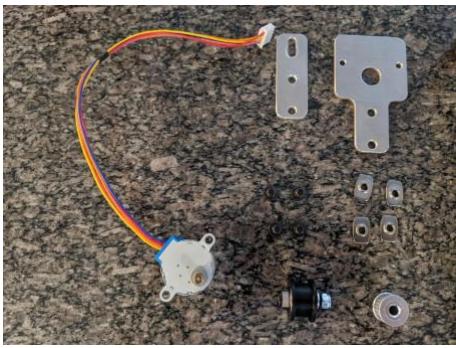
2.6 Attach the 4mm bore gear



Step 3. Assemble the Y-axis

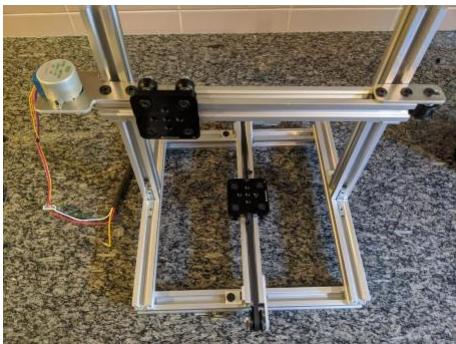
3.1 Gather necessary components

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



Appendix D: Hints

Exercise 1 Supporting Notes

- When upping your baud rate on the green board: You need to get the new version of the MECH 423 serial communicator. In the older version, the baud rate variable is typed int16, which is insufficient for the higher baud rates. You can also use alternative like Putty, or code a custom C# form.

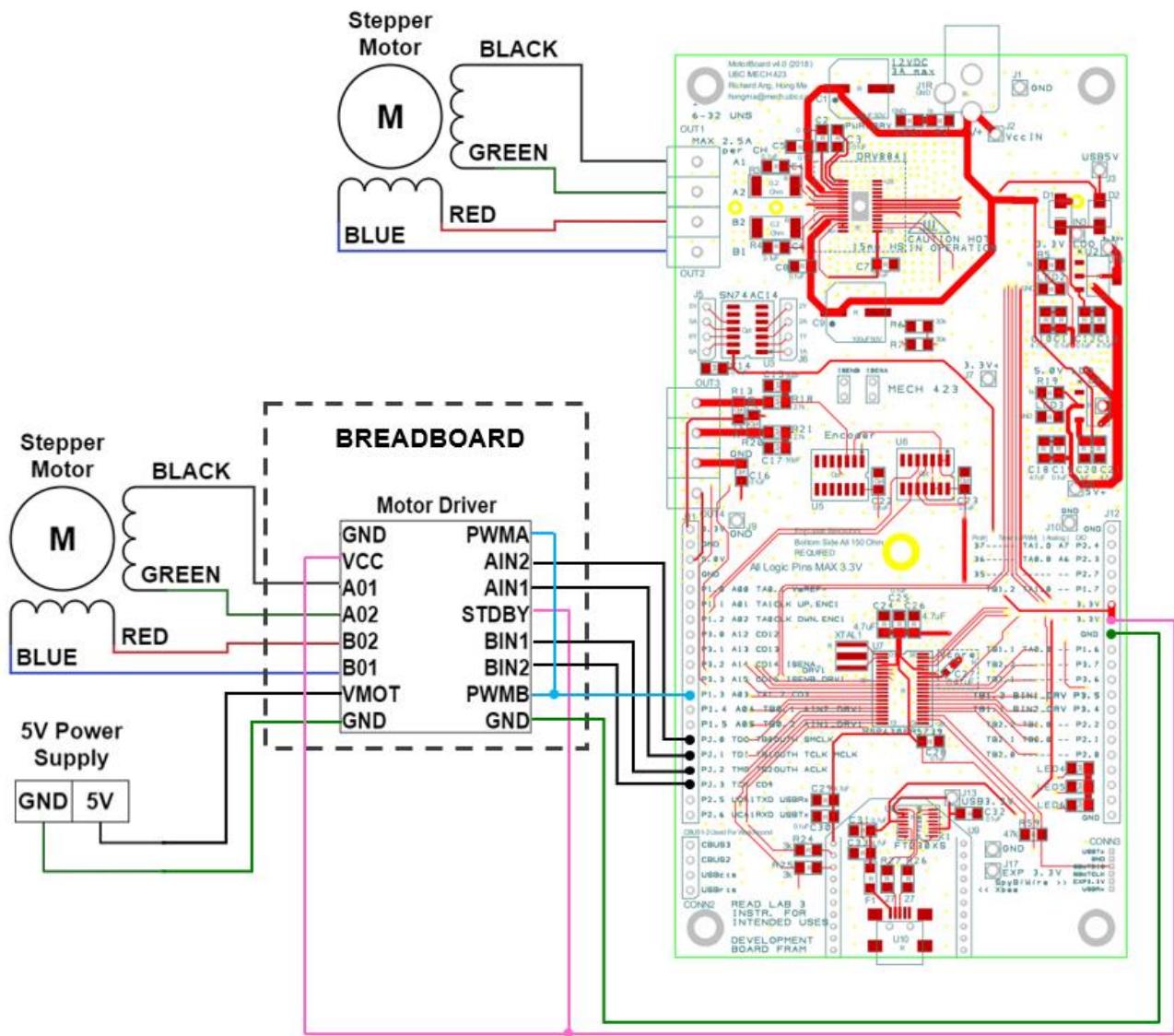
Exercise 3 Supporting Notes

- When driving the stepper using the DRV8841 chip, you will need to sequentially energize and de-energize the 2 phases to make the motor rotate. The commands to each of the 4 stepper wires will correspond to pins P1.4, P1.5, P3.4, and P3.5.
- IMPORTANT:** Do NOT set the outputs of P1.4 / 1.5 / 3.4 / 3.5 to **digital high** (3.3V out) to energize a phase. This will cause the driver chip to overheat and burn out.
- Instead, set each of P1.4 / 1.5 / 3.4 / 3.5 to be timer outputs for timers TB0.1, TB0.2, TB1.1, TB1.2.
- To energize a phase, output a PWM with **25% duty cycle**. To de-energize, you can set 0% duty cycle to the timer output, or you can set the output to digital low.
- Notice that as velocity **increases**, the CCRO value of the timer that counts steps must **decrease**. Hint: Calculating the CCRO value as a function of the velocity command is a nonlinear function.
- It is recommended to write a mathematical expression for the stepper velocity (rate of step change) as a function of the CCRO value to determine how to set the CC00 value as a function of the velocity command.

Exercise 6 Supporting Notes

- The 2 stepper motor drivers have different interfaces that require different firmware to control, though each can be made to produce the same outputs. It is recommended to write one function for each motor (ie, STEP_X() and STEP_Y()) that each implement the different control strategies, but have an identical call interface, to make higher-level code easier.
- To prevent this stepper and driver from overheating, the outputs to the phases should use a PWM signal for energizing. A good value for the PWM duty cycle is 60%, but if your motor feels hot, feel free to lower this value.
- The same PWM signal can be shared between input channels A and B on the external motor driver.
- The PWM signal does not need to be shut off when energizing / de-energizing phases, it may be allowed to run continuously. The AIN1 / AIN2 / BIN1 / BIN2 control the outputs.
- When setting pins PJ.0 thru PJ.3 to define the steps of the stepper on the external driver, pay close attention to the 'H-SW Control Function' table at the top of pg.4 in the motor driver datasheet. When switching the voltage to the stepper coils, the 'low' state of the waveform should be using the 'short brake' combination of inputs, not the 'high impedance' combination. Doing this incorrectly produces a mostly-working result but may damage the driver over time.
- When synchronizing steppers, you will do calculations involving division. **BE VERY CAREFUL OF DATA TYPES AND INTEGER DIVISION.** You may cast values to float as necessary to preserve precision while dividing.

Appendix E: System Schematics



Notes:

- 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:

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

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