Universal PoE Stepper Motor Driver for Argonne National Lab

Armando Rodea
Bryan Logan Monk
Putra Derindra Firmansyah

Final Report for ECE 445, Senior Design, Spring 2022
TA: Evan Widloski

May 2022
Group 28

Abstract

The universal power over ethernet (PoE) stepper driver is a small, accurate and low cost stepper motor driver that tackles problems with current drivers at Argonne National Lab. Combining PoE circuitry with a microcontroller and high accuracy stepper motor drivers, a user will be able to connect our device to a PoE network switch and use a single cable for both powering and communicating with a stepper motor. This report will delve into the design of the project, verify it, outline the costs and discuss future work to be done on the project.

Contents

1. Introduction

1.1 Problem and Solution Overview

At Argonne National Laboratory the Advanced Photon Source houses a synchrotron beamline where scientists conduct various x-ray diffraction experiments. Numerous stepper motors are used for precise alignment and automation during experiments. Currently, the drivers that are used are bulky and expensive. Each driver needs a dedicated slot, and in many sections of the beamline, these slots are already filled, making it difficult to add more drivers. Hence, a compact, scalable driver would be ideal, but current market solutions require a power supply for each driver in addition to wiring for serial communications. This is not an optimal solution as the number of motors increases.

This project aims to make use of already installed power over ethernet (PoE) network switches at Argonne to both communicate with a stepper driver and power it. Such a solution would allow for neat cabling and saving of space. In our solution, an ethernet cable connects to our device from a PoE network switch and a circuit will separate the power and data transmitted. Furthermore, there will be a motor control system that interfaces with the microcontroller unit (MCU). This will then generate current pulses to control the motor. In order to power the MCU and the motor control system, there is a power subsystem that steps down 48 V from the network switch into 12 V for the motors and 3.3 V for all the integrated circuits (ICs) on the board. This module will be universal to most stepper motors with different current requirements while being relatively low cost.

1.2 High-level Requirements

In order to create a universal PoE stepper motor driver, it requires features such as requesting the necessary power from the network switch, separating ethernet data and power, stepping down high voltage to a usable level for the motors and ICs, and lastly circuitry and logic to control stepper motors accurately. The list below contains the three high-level requirements needed in order to make such a device.

1. The driver must be compatible with any bipolar stepper motor with a requirement of 3 A per phase or lower. This requirement is necessary since there are a wide range of stepper motors used at Argonne with different sizes and current requirements. Being able to power motors with a rating of up to 3 A per phase covers the vast majority of stepper motors used.
2. The driver must be able to control and step down the 47.5-48.5 V PoE++ power delivery to 11.5-12.5 V and 3.0-3.6 V to create up to 6 A of usable power for the motors. 3.3 V is the required voltage for the IC chips on the board. This is necessary because the proper PoE negotiation must happen so that power is supplied to the device, and the stepper motors require a 12 V supply with up to 3 A per phase for each motor.
3. The driver must have reliable transmission of data over ethernet and interpret the commands into STEP/DIR pulses to drive the motor; reliable meaning that it does not overheat and so does not reach over 125° C as to not heat the TMC2660 stepper driver chip. As Argonne uses stepper drivers for important alignment and automation purposes, it is important to ensure that it may continuously function. The TMC2660 chip has a maximum operating temperature rating of 125° C. [1]

1.3 Design Overview

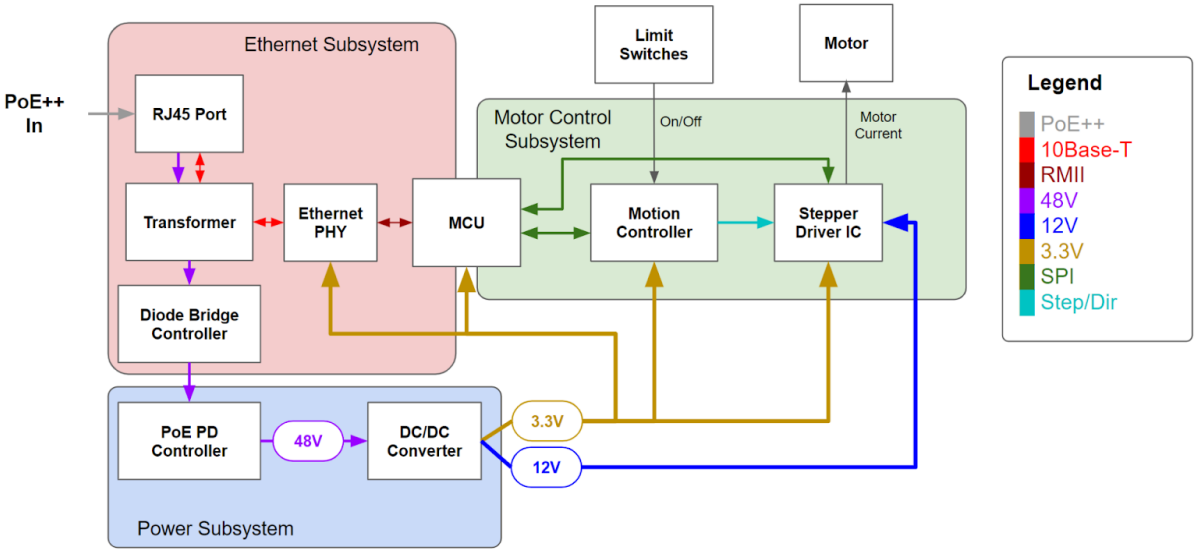


Fig. 1: Block diagram of universal PoE stepper driver.

The universal PoE stepper driver consists of three subsystems as illustrated in Fig. 1. The ethernet subsystem accepts an ethernet cable from a PoE network switch through an RJ45 port. The RJ45 port interfaces with the transformer through AC and DC signals for data and power, respectively. The transformer sends 48 V to the diode bridge controller for it to be rectified to account for polarity in different pins going into the power delivery (PD) controller in the power subsystem. The transformer also communicates with the ethernet PHY chip to send and receive data which then goes into the MCU.

The power subsystem is responsible for powering all integrated circuits (ICs) on the board as well as powering the motor. To do that, the power subsystem must accept 48 V and convert it into 3.3 V which powers all the ICs, and 12 V which is used to power the motor.

The motor control subsystem contains the logic and circuitry to drive the motor and stop the motor using limit switches. The motion controller first receives SPI commands from the MCU for various commands. It then produces the corresponding step and direction pulses which go into the stepper driver IC. The stepper driver IC then produces a current to move the motor accordingly. Additionally, the motion controller receives on or off signals from the limit switches which stops the stepper motor if a boundary is reached.

2 Design

2.1 Ethernet Subsystem

2.1.1 Ethernet Subsystem Design Procedure

For the ethernet subsystem, we heavily referenced the Analog Devices DC2911A development board for our PoE circuitry [3]. This features both the LT4293 PD controller and the LT4321 diode bridge controller. It can provide both the PoE+ and PoE++ standards for which there are network switches installed at Argonne. We were also already in possession of this board from Argonne, which is why we used this as a starting point for our design.

For our ethernet PHY, we used the Texas Instruments DP83848. Because of the parts shortage, all standard 10/100 Base-T ethernet PHYs were out of stock. For this reason, we needed to desolder a PHY from some external source, and the Waveshare DP83848 development board was the best candidate since this chip had seemingly widespread support and it was relatively cheap. Additionally, examples were using this chip with our exact MCU, the STM32F407. However, an alternative to this would be the Microchip LAN8742. This is commonly integrated on STM32 development boards with ethernet connectivity, and we would have started with one of these development platforms if we went this route.

2.1.2 Ethernet Subsystem Design Details

This subsystem is responsible for receiving both power and data from the PoE network switch, also called the PSE. The ethernet data signals are AC, with the Rx pair between pins 1 and 2 while the Tx pins are between 3 and 6. The power is 48 VDC, which can be on different pins depending on which PoE standard is being used. In our case, we are using two standards: PoE+ (25.5 W) and PoE++ (51 W). PoE+ operates in mode A, while PoE++ operates in mode B. Mode A has positive DC voltage on the Rx data lines (pins 1 and 2), and GND on the Tx lines (pins 3 and 6). In mode B, the normally unused pins are used for power, with positive voltage on pins 4 and 5 and GND on pins 7 and 8.

After the power and data come in through the RJ45 port, a transformer allows the AC ethernet signals to pass through while the DC power does not. The power is taken from the transformer center taps, which are the POE_CT signals seen on Fig. 2.

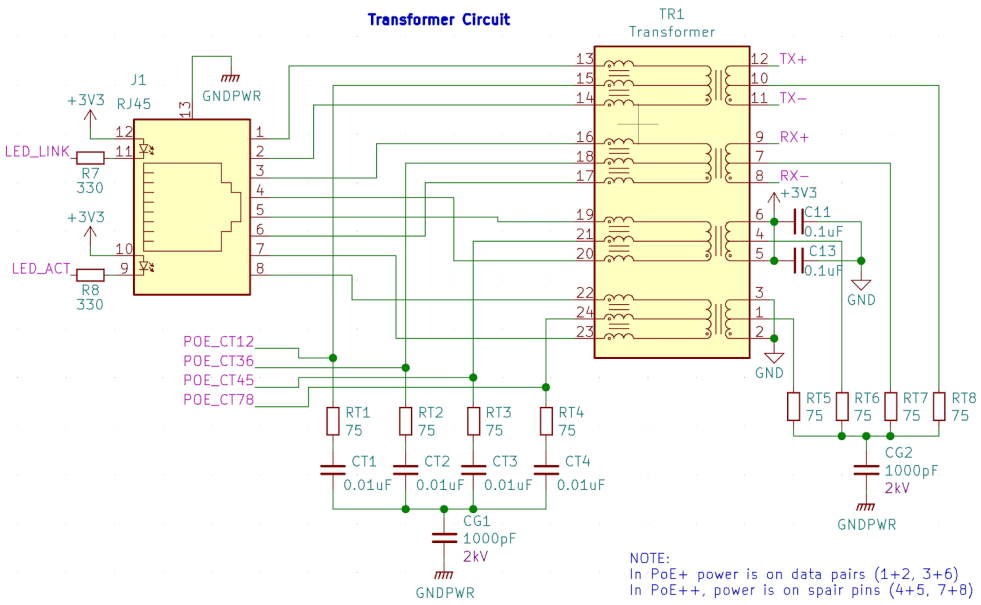


Fig. 2: Ethernet transformer circuit schematic.

To account for the power being on different pins, A “diode” bridge is needed to rectify the power. While a passive rectifying bridge with actual diodes can be used, a more efficient method is to use MOSFETs. In this case, each center tap signal is connected between the source and drain of two MOSFETs. A controller IC (Analog Devices LT4321) detects which signals correspond to positive and negative voltage and drives the gates of the MOSFETs to connect positive voltage to the VPORTP rail and negative voltage to the VPORTN rail.

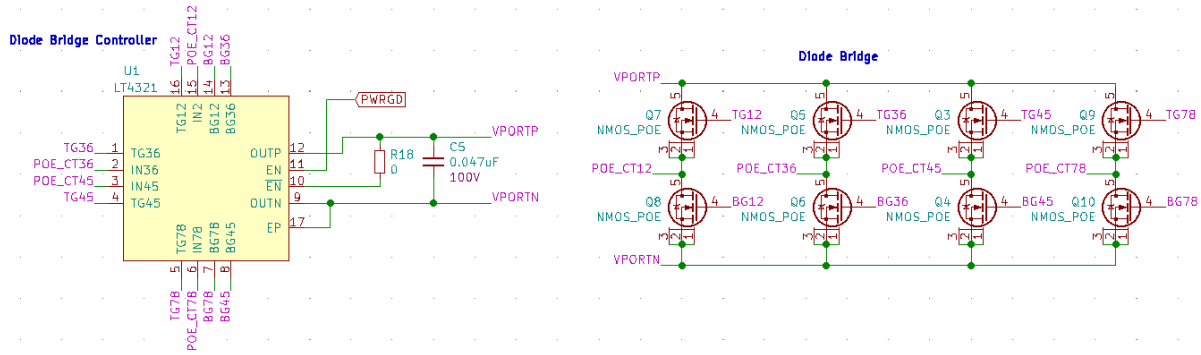


Fig. 3: Diode bridge circuit schematic

Before power can be supplied to the device, a negotiation must occur between the PSE and the PD (powered device). For this, we used the Analog Devices LT4293.

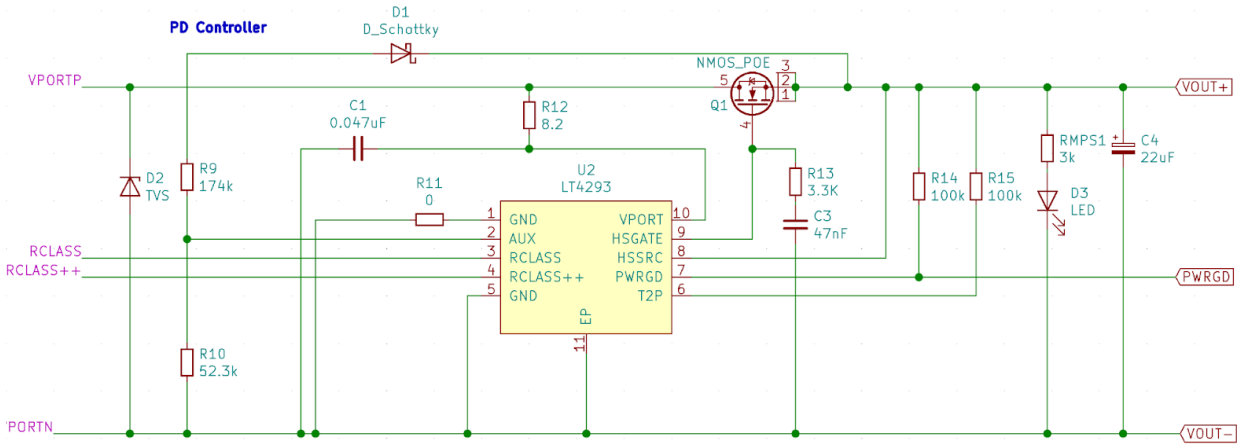


Fig. 4: PD controller schematic.

For a negotiation to occur, the PSE first applies two voltages between 2.7V and 10.1V, measuring the current response to verify a characteristic 25 kΩ resistance that demonstrates a PD is present. This is the DETECT voltage shown in Fig. 5. After this test, the first classification voltage is sent between 14.5V and 20.5V. The PSE measures the signature current, and depending on the reading, more classification voltages can be sent. More cycles correspond to more power being allowed to the PD. for PoE++, a max of three classification voltages can be sent, as seen in Fig. 5.

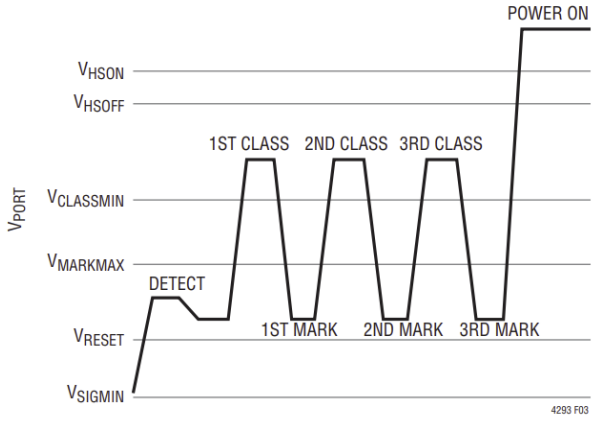


Fig. 5: Class 3 PoE negotiation diagram.

On the data side, the Tx and Rx pairs that are coupled through the transformer connect to the ethernet PHY. PHY stands for physical layer transceiver, and this is where the AC ethernet signals are interpreted into a digital interface that can communicate with the MCU. We used the DP83848 PHY which is a standard 10/100 Mb/s ethernet PHY. As seen in Fig. 6, the Tx and Rx pairs connect to the top left of the IC. RMII, which stands for reduced MII, is the protocol used to communicate with MCU. These pins can be seen on the right side of the schematic, including TXD0, TXD1, TXEN, TXD0, RXD1, CRS_DV, MDC, and MDIO. Finally, RMII requires a 50 MHz oscillator shared between the PHY and the MCU which can be seen in the bottom right of Fig. 6.

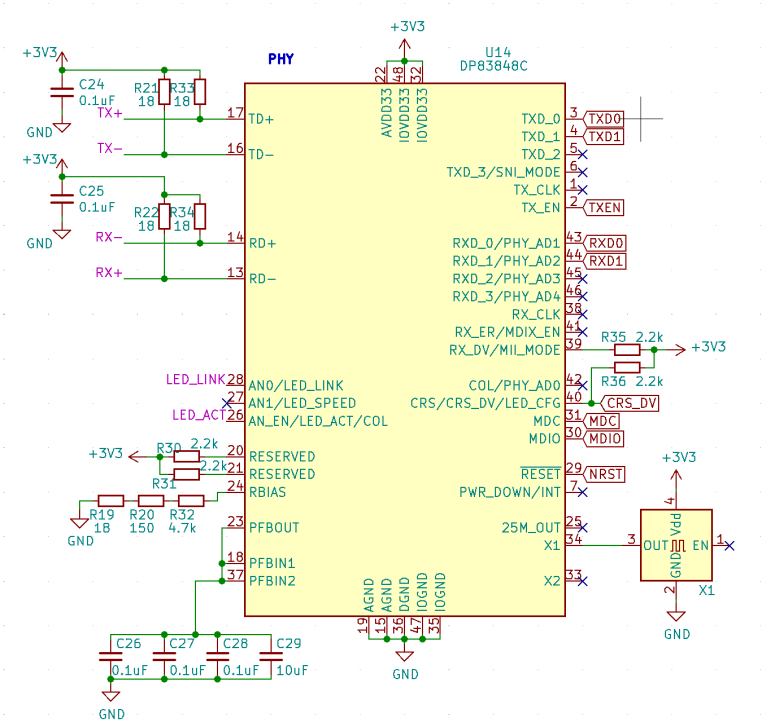


Fig. 6: DP83848 ethernet PHY schematic.

As far as PCB layout for this subsystem, we followed the same general design as the DC2911A board. However, this board was 6 layers, while ours was only 2, so many adjustments were made to complete the routing. The full PCB layout can be referenced in Appendix E. The DC2911A simply had an RJ45 port for output data, so on our board we routed the data pairs directly to the DP83848 PHY. It is important that these traces are approximately the same length to match impedance. Additionally, the PHY has outputs for status LEDs on the RJ45 port that signify data transfer, which we added.

2.2 Power Subsystem

2.2.1 Power Subsystem Design Procedure

For our power subsystem, we used the E48SC12010NRFA for our 12 V DC/DC converter, which is capable of providing 120 W (10 A) of power. This provides the 12 V supply for the motors, and we needed enough current to provide a max of 3 A per motor phase, or 6 A total. The 12 V is stepped down to 3.3 V to power the ICs, and for this we used a simple linear regulator, the NCP1117, which can provide up to 1 A. A potential alternative to this was a switching buck converter, but we determined that we did not need this extra efficiency, and we did not want to introduce extra noise into our circuit that might interfere with current sensing by the motor drivers.

2.2.2 Power Subsystem Design Details

This subsystem is rather simple in design. First, the outputs from the PoE circuitry are VOUT+, VOUT-, and PWRGD. VOUT+ is positive PoE voltage, while VOUT- is ground. PWRGD serves as an enable pin for the power supply, recommended for the most reliable operation. These outputs connect to the Vin+,

Vin-, and ON/OFF pins of the 12 V DC/DC converter, respectively. +12 V is then provided between the Vout+ and Vout- pins. For standard 12V operation, the E48SC12010 datasheet instructs connecting the Sense+ pin to Vout+ and Sense- to Vout- [2]. The TRIM pin is left unconnected.

As mentioned above, the 12 V is only used for powering the motors, and must be stepped down to 3.3 V to power the ICs. Our 3.3 V regulator is a simple linear regulator. The input 12 V is connected between the VI and GND pins, and the output is between VO and GND. 10 uF capacitors are needed on the input and output pins, and we also added a status LED to show power.

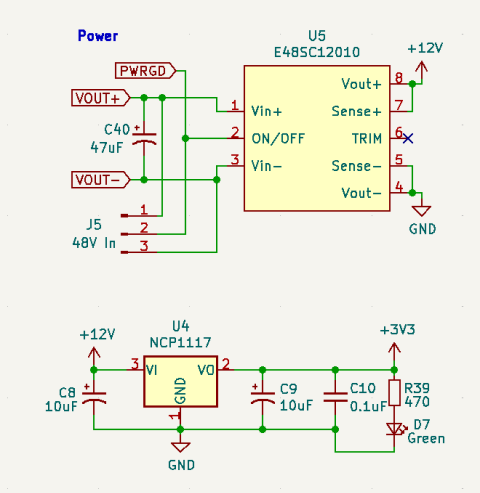


Fig. 7: Power subsystem schematic.

For the layout of this subsystem we initially thought about putting the 12 V converter on our main board, however, its large size made it difficult to fit while keeping our board compact. As a result, we decided to have a separate PCB for power shown in Fig. 8. This connects to our main board through jumper wires.

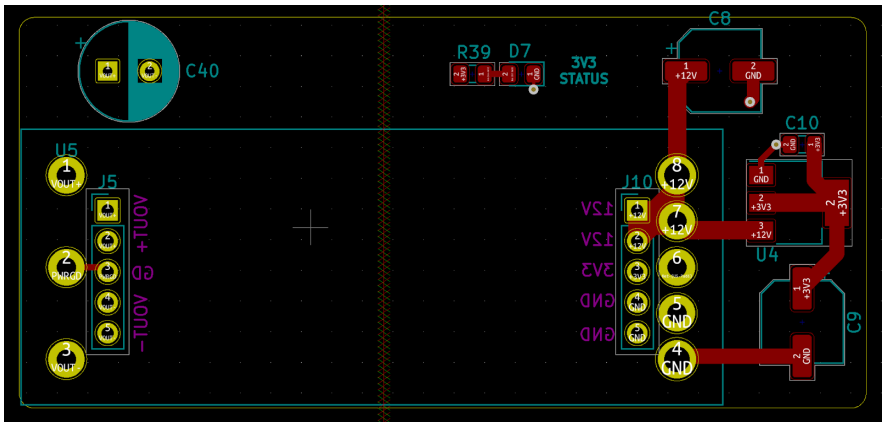


Fig. 8: Power board PCB layout.

2.3 Motor Control Subsystem

2.3.1 Motor Control Design Procedure

We used two main ICs for the motor control subsystem: the TMC429 and the TMC2660. These are made by the German company Trinamic Motion Control, which specializes in laboratory automation. They are known for their precision and reliability, and their drivers are used frequently at Argonne. Once we knew we wanted to use Trinamic drivers, there were a few options for the configuration we used. We had the option between a stepper motor gate driver plus external power MOSFETs or a driver that had integrated MOSFETs. This is a trade-off between simplicity and power. We opted for the latter option to simplify the board layout while maintaining a reasonable amount of power. The TMC2660 is Trinamic's most powerful driver IC, with a maximum of 2.8 A of current per motor phase. This is more than enough for most steppers at Argonne.

Given that our driver would control the stepper motors that are used for alignment and automation for the synchrotron beamline where x-ray diffraction experiments are conducted we also aimed to have precise drivers. With this in mind, we also chose the stepper driver that allowed for the needed accuracy with the ability for 256 microsteps. For maximum accuracy, Trinamic advises that these drivers are coupled with a motion controller IC. In the case of the TMC26x series, the TMC429 is recommended. The purpose of this chip is to offload critical calculations and look up tables from the MCU and make interfacing with the driver IC easier and more reliable. Rather than sending STEP/DIR pulses directly to the driver, the MCU can write parameters such as position and velocity through SPI and not have to worry about timing.

2.3.2 Motor Control Design Details

Both of the motor control chips require SPI for the initialization of certain parameters. During operation, though, The MCU only sends SPI commands to the TMC429, which in turn sends STEP/DIR signals to the TMC2660. Step and direction (STEP/DIR) is the standard digital interface for stepper motor drivers. First, the value of DIR indicates whether the motor should move clockwise or counterclockwise. The positive edge of STEP tells the driver to move the motor one unit in the direction specified by DIR. The unit is determined by the driver IC and can either be a full step, half step, or microstep. In our case, the TMC2660 is configured for max resolution, which is 256 microsteps per full step. Therefore, one STEP pulse moves the motor by 1 microstep, which is

$$\frac{1\,fullstep}{256\,microsteps} * \frac{1\,rotation}{200\,fullsteps} * 360° = 0.00703125°$$

The SPI and STEP/DIR inputs can be seen on the bottom left of the TMC2660 schematic symbol in Fig. 9. Next, there are power inputs for each motor phase, VSA and VSB, as seen on the top of the symbol. As specified in the TMC2660 datasheet [1], There should be both electrolytic and ceramic capacitors on these inputs, as well as a 220 nF cap between VHS and 12 V, and a 470 uF cap between 5VOUT and GND. On the right side of the symbol in Fig. 9 are the motor outputs. Since there are two coils, there are 4 connections total: OA1 and OA2 for coil A and OB1 and OB2 for coil B. Each of these connections has 4 pins on the TMC2660 due to large current output. On the left side of the symbol are the current sensing resistors. The full motor current runs through these resistors, and the driver measures the voltage across them and limits the current if it measures a high enough voltage.

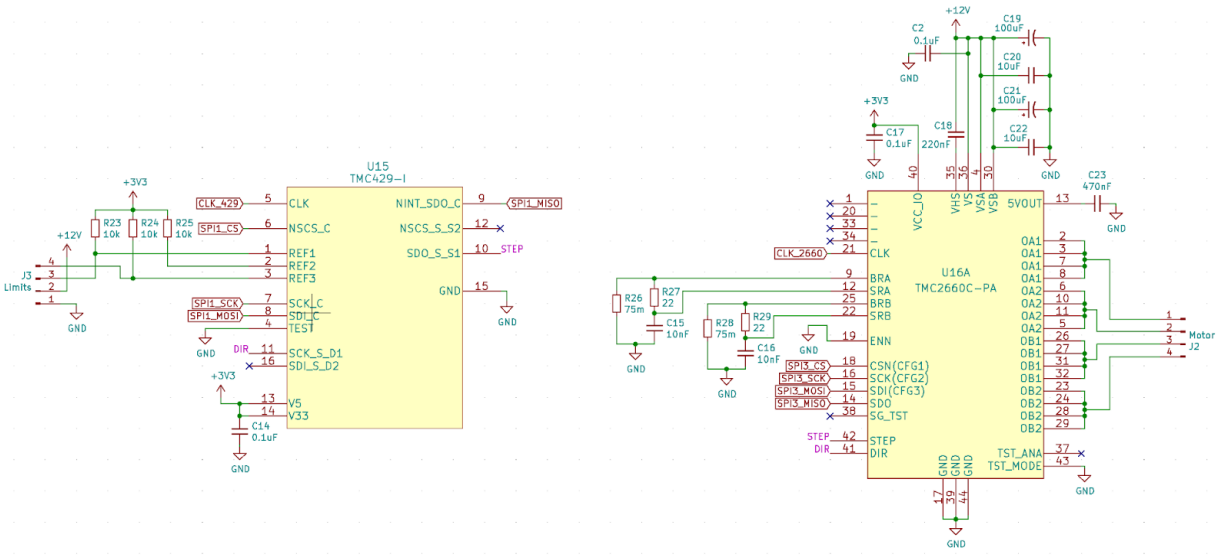


Fig. 9: Motor control subsystem schematic.

The TMC429 is rather simple in terms of external circuitry. In addition to the SPI inputs, it has STEP/DIR outputs, a clock input, and three reference switch inputs. The reference switches are used as emergency stop switches to stop the motor if they are pulled to ground. The TMC429 can drive up to 3 motors, however since we are only driving 1, we configured REF1 to be the left stop switch and REF3 to be the right stop switch. In other words, if the motor is moving counterclockwise and REF1 is activated, the motor will be stopped. Conversely if the motor is moving clockwise and REF3 is activated, the motor will also be stopped.

Since there is a considerable amount of current traveling to the motor, the PCB layout required particular attention. We created 4 large planes on the bottom layer of the board corresponding to each motor coil output, with several vias transporting this current to each plane. Then, thick traces connected these planes to the motor connector.

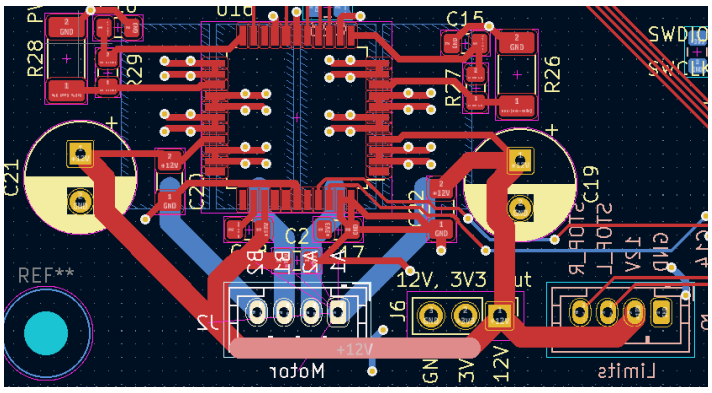


Fig. 10: TMC2660 PCB layout.

The commands sent from the microcontroller communicate through SPI commands with datagrams of 32-bits. In Fig. 9 the aforementioned communication can be noted with the SPI inputs on both the chips.

Using the TMC429 datasheet [4] it can be seen that the most significant bits hold a 4-bit register value and a read/write bit. In order to move the motor certain registers must be chosen and certain bits in the lower 24 bits must be configured to alter the register contents. A high level flow of this process is presented in Fig. 11 below.

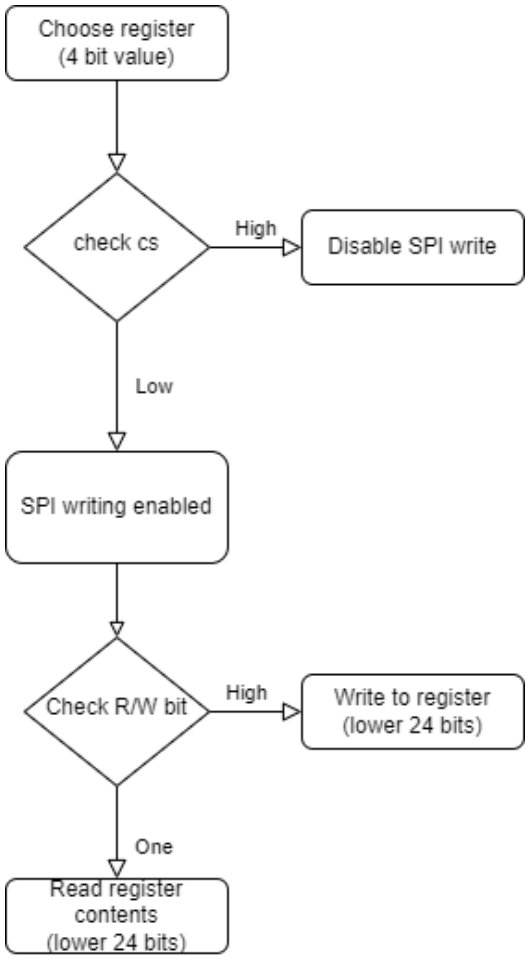


Fig. 11: Flow chart of motor control code.

For example, register zero (4'b0000) controls the target position with its lower 24 bits. To get it to move to a target position a 24-bit value representing the position must be sent. This command is sent through the HAL library command.

$$HAL\_SPI\_TransmitReceive(\&hspi1,\ DATA\_SENT,\ DATA\_RECIEVED,\ NUM\_BYTES\_WRITE, TIMEOUT);$$

Where hspi1 is the SPI handler that is used to refer to the TMC429 and the 32 bits are sent in DATA_SENT. In this case, DATA_RECIEVED is not important as it was marked as a "write" command;

however, for register reads this is where the output will be stored. NUM_BYTES_WRITE is self-explanatory and $TIMEOUT$ is the amount of time it waits for data to be sent and received before continuing program execution.

With all this in mind, we decided to create a couple of functions that would accomplish tasks such as setting the positions, getting the position, setting the velocity, getting the velocity, and changing the current scaling that determines how much current the motor is driven with. These functions would then pair nicely when receiving external commands from ethernet.

2.4 MCU

2.4.1 MCU Design Procedure

The MCU is not its own subsystem but straddles the ethernet and motor control subsystems. Its job is to receive/transmit the ethernet data via the RMII interface from the PHY, interpreting this data into motor controls to send through SPI.

We decided to use an STM32F407 for our MCU, due to its built-in ethernet MAC and ample processing power. Additionally, we first thought we might use a motor control API provided by Trinamic, which was targeted for ARM processors. Another potential option could be the Microchip PIC18 series, some of which have a built-in Ethernet MAC and PHY. This would decrease design complexity, but would come at the cost of reduced processing power and peripheral support.

2.4.2 MCU Design Details

For the circuitry surrounding the MCU, we first allocated pins for the RMII input. Next, we had two SPI outputs for the two motor control ICs. GPIOs were used for the CS pins in each SPI channel. Both Trinamic ICs require a clock input, and to keep the layout simpler, we decided to have 16 MHz PWM outputs from the MCU rather than an external oscillator. We also added an 8 MHz crystal for the MCU to ensure accurate timing, some status LEDs, and a UART interface for debugging purposes. Finally, our method of programming was an SWD interface, which requires two signals: SWCLK and SWDIO, in addition to 3.3V and GND.

3. Design Verification

The high-level requirements for the project have all been achieved except for ethernet data transmission. We were able to run multiple different stepper motors with a requirement of 3 A or lower, able to step down PoE voltage so that it powers the rest of the board, and ensured that there was no overheating. To break down the projects, there are requirements and verifications we have to achieve and conduct as can be seen in appendix A. The results are further discussed in this section.

3.1 Ethernet Verification

Table 1: Ethernet subsystem output voltage.

Voltage Measured (V)
54.3172
54.3171
54.3171
54.3171
54.3171

The first ethernet subsystem requirement is that it must be able to output 47.5 - 48.5 VDC. In order to verify this requirement, we used the GS110TUP PoE network switch that is connected to the RJ45 port on our board. Table 1 illustrates the value measured on the output of the ethernet subsystem and it averages 54.3171 V. This requirement is partially achieved because, for the purposes of this project, this fulfills the minimum required voltage for the design to operate. The measured voltage is higher than the requirements because of the voltage supplied by the PoE network switch. On the network switch's datasheet [5], it states that it uses a 54 V and 4.7 A power supply hence the measured voltage. From initial research, we thought that PoE only transmitted at 48 V but in actuality, it is within a range of 48 - 57 V. [6]. It is important to note that this poses no issue for our device, as the voltage we measured is still within the range of the DC/DC converter maximum input voltage. However, as this is still outside of the requirement, we conclude that this requirement is partially achieved.

The second requirement in the ethernet subsystem is that it must be able to receive TCP packets accurately through the RJ45 port into the MCU. We were unable to get the ethernet software working hence this requirement was not achieved. While developing the software, we saw that the LEDs were blinking on the ports hence, it signifies that data were transmitting. However, as the packet transmission was investigated, we did not see any of which went to the development board's MAC address hence, we were unable to connect to it. We believe that CPU memory barriers for ethernet were not properly configured. After further research, we found that ethernet uses DMA (Direct Memory Access) in order to reduce CPU utilization. Ethernet uses memory descriptors to identify which parts of memory can be used for DMA but in STM32's generated code, descriptor memory is not necessarily configured as the

ethernet's device memory. Hence, ethernet fails when trying to read within the memory barrier because the STM32 has not set the OWN bit there to signify that it can be read by this specific peripheral.

3.2 Power System Verification

Table 2: Voltage and current output from 12 V DC/DC converter with a load of 2 Ω.

Voltage Measured (V)	Current Measured (A)
12.001	5.783
12.002	5.776
12.003	5.785
12.007	5.784
12.011	5.777

The first requirement discussed is that the power subsystem must be able to receive a 47.5-48.5 VDC and output 11.5-12.5 V with a current of 5.5-6.5 A. In order to verify this requirement, we used a DL83021A electronic load in constant resistance mode at 2 Ω so that we could measure the appropriate current and voltage. To power up the power module, we used a DP831 programmable DC power supply at 48 V and 2 A. Table 2 shows the results that were obtained after 5 trials, averaging to 12.005 V and 5.781 A. Hence, it concludes that the design achieves the requirement.

Table 3: Voltage output of 3.3V linear regulator.

Voltage Measured (V)
3.301
3.301
3.302
3.300
3.301

The second requirement in the power subsystem is that the power subsystem must be able to receive a 47.5-48.5 VDC and output 3.0-3.6 V. In order to verify this requirement, we again used the DP831 programmable DC power supply at 48 V and 2 A. We also used a multimeter to measure the voltages of the 3.3 V output headers of the power board. The result can be seen in Table 3 which averages to a reading of 3.301 V. Hence, this requirement has been achieved.

3.3 Motor Control Verification

The first requirement tested is that the motor control subsystem must be able to accept SPI commands from MCU and convert the commands to step/dir pulses that are accurate enough for microstepping 256 microsteps per full step. Testing the Motor Control Subsystem was performed by sending specific SPI commands from the microcontroller to the TMC429 and displaying the returned information using UART to a serial monitor. For example, a target position of 52800 was set to the motor to move a full rotation. The reason this number corresponds to a full rotation is because the TMC2660 is configured to a resolution of 256 microsteps per full step, and there are 200 full steps in a full motor rotation so 256*200 = 52800 microsteps per rotation. We verified visually that the motor shaft did in fact move a full rotation, and the position was then read back on the serial monitor and verified to see if the set position was the current actual position.

Another requirement we had was to provide 3 A to the motor without exceeding 125° C. Unfortunately, we were only able to get an infrared thermometer meant to measure human temperature and at one point we exceeded the temperature limit. The limit only allowed us to measure up to 43° C. With this in mind, we plotted the data shown below with Fig. 12 and assuming linearity we used the best fit line equation to estimate the temperature at 3 A and arrived at the result of 73.4° C. This temperature is well below our target of 125° C.

Given the fact we are assuming linearity and using an equation that may not show the full picture, we can only partially verify this subsystem. Moreover, we tested this on multiple different motors and they showed similar results; however, none are rated for 3A so even if we had the ability to accurately measure the temperature we would still feel unsafe in providing a current that our motors were not intended for.

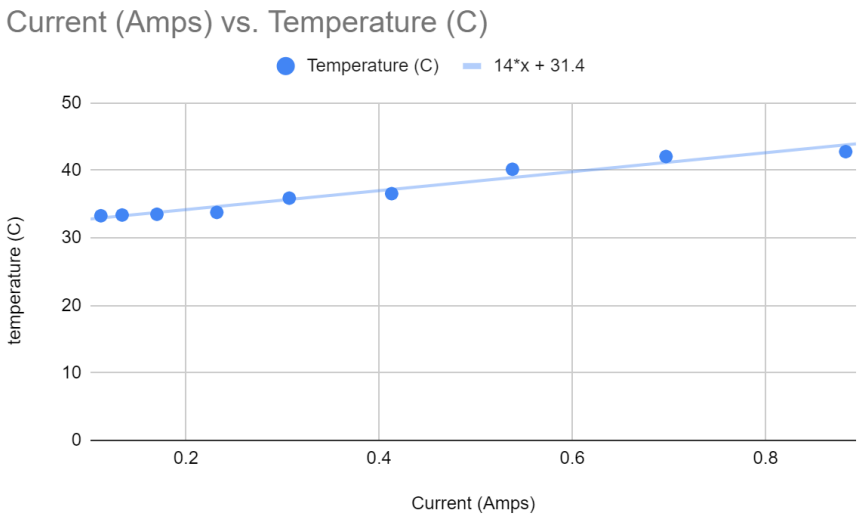


Fig. 12: Plot of the current against the measured temperature of the TMC2660.

4. Costs

4.1 Parts

The total cost of parts totals to $146.84. For a complete list of parts refer to Appendix B.

4.2 Labor

To estimate the labor costs, it would be important to note how much an ECE graduate from University of Illinois at Urbana-Champaign earns on average. An electrical engineering graduate earns $79,129 per year and a computer engineering graduate earns an average of $99,145 per year [7]. Averaging between these two majors results in $89,137 per year which roughly translates to $44/hr with the assumption of 40 hour work weeks and 50 working weeks per year. It has been observed that this project takes 150 hours to complete therefore, the labor cost can be calculated using the following formula and presented by Table 4 below:

$$2.5 * 150\,hr * \$44/hr * 3\,people = \$49,500$$

Table 4: Estimated labor costs.

Name	Hourly Rate	Total Hours	Expense Multiplier	Total Cost/Person
Armando	$44.00	150.00	2.50	$16,500.00
Bryan	$44.00	150.00	2.50	$16,500.00
Putra	$44.00	150.00	2.50	$16,500.00
			Labor Total	**$49,500.00**

Adding the costs for labor and parts, it can be estimated that the total cost for this project is: **$49,646.84**

4.3 Schedule

Refer to appendix C for the schedule.

5. Conclusion

5.1 Accomplishments

In the end, we were able to test and verify all of the subsystems except for ethernet communication. To specify we were able to get the PoE circuitry to work as it was able to output 54V, signifying that the proper PoE negotiation had taken place. This worked for both PoE+ and PoE++ standards. Moreover, we were then able to successfully step down the 48v received from PoE into 12V and 3.3V usable for the motor and ICs. We were able to get all the ICs involved in motor control to communicate with each other as the microcontroller was able to send commands to the motion controller which were interpreted into step directions pulses that were used to drive the motors.

5.2 Uncertainties

Overall we have ideas as to why the ethernet subsystem did not work as intended. The ideas have been touched on before but we are still unsure as to exactly which of those if any is the solution to our problem. Possible ways to fix this will be discussed in the future work section below.

As a result of this system, we were not able to work through the full integration of the software. This means that we are still uncertain about the steps we would need to do to get it working on the SPEC system that Argonne, our end users, currently use.

5.3 Ethical considerations

To ensure safety and to be in accordance with IEEE Code of Ethics #9 [8], we must be sure to avoid injuries. Our project has two potential hazards one of them being the high voltages used and the other being the heat of various components, especially on our power board. To mitigate the risks of working with high voltages for our end-user we have an enclosure so the power pins are not directly exposed. An additional precaution we have taken is insulating for potentially dangerous components.

5.4 Future work

Moving forward with the project it would be pertinent to first address the only part that was not able to work - ethernet. To get this system working it would be best to address the issues that we had while working on it. Seeing as two PHY chips were shorted and we believe the cause to be transient voltages on the Tx/Rx lines, we would include some TVS diodes on the PHY data inputs. Moreover, we would use a different PHY that is also included on an STM32 dev board. Once that is tested and is proven to work, we could move those same components with the working software onto our custom board. Going forward, it would be ideal to integrate our device with the SPEC control software that is currently being used at Argonne. This is proprietary software used specifically for synchrotron beamline control. From a design and usability standpoint, we would also switch to a 4-layer PCB design to increase ease of routing, signal integrity, and optimized heat management. Additionally, instead of connecting our power board to our main board through jumper wires, we would add stackable headers so that the power board could fit neatly into a socket. This would also result in better current conduction. To increase performance of the motor control subsystem, we would add a dedicated 16 MHz oscillator for both the TMC429 and TMC2660 clock inputs. Due to high amounts of noise that resulted from the PWM outputs from the MCU, we were forced to use the internal clock for the TMC2660, which is not recommended. Finally, to allow for even higher power motors and a generally more robust design, we would use one of the Trinamic gate driver ICs such as the TMC262 coupled with external MOSFETs.

References

[1] *Trinamic, "TMC2660C datasheet - trinamic.com" 024-Feb-2022. [Online]. Available: https://www.trinamic.com/fileadmin/assets/Products/ICs_Documents/TMC2660C_datasheet_rev1.03.pdf.*

[2] "DC2911A PoE Development Board Schematic," *Analog Devices*. [Online]. Available: https://www.analog.com/media/en/technical-documentation/eval-board-schematic/DC2911A1-SCH.pdf.

[3] *"Delphi Series E48SC12010 Datasheet," Delta Electronics. [Online]. Available: https://filecenter.deltaww.com/products/download/01/0102/datasheet/DS_E48SC12010.pdf.*

[4] *Trinamic, "TMC429 datasheet - trinamic," 03-Mar-2022. [Online]. Available: https://www.trinamic.com/fileadmin/assets/Products/ICs_Documents/TMC429_datasheet_Rev2.05.pdf.*

[5] *"GS110TUP - 10-Port Gigabit ethernet ULTRA60 poe++ smart desktop switch with 1 SFP and 1 copper uplink," NETGEAR. [Online]. Available: https://www.netgear.com/support/product/GS110TUP.aspx.*

[6] *"Power over ethernet (poe) explained," POE Explained - Understanding and using Power over Ethernet. [Online]. Available: https://www.veracityglobal.com/resources/articles-and-white-papers/poe-explained-part-2.aspx.*

[7] *T. G. C. of E. Communications, "Salary Averages," ece.illinois.edu. https://ece.illinois.edu/admissions/why-ece/salary-averages*

[8] *"IEEE code of Ethics," IEEE. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed: 04-May-2022].*

[9] *DP83848C/I/VYB/YB PHYTER™ QFP Single Port 10/100 Mb/s Ethernet Physical Layer Transceiver Datasheet- Ti*. https://www.ti.com/lit/ds/symlink/dp83848c.pdf?ts=1645736599241.

Appendix A: Requirement and Verification Table

Ethernet Subsystem Requirements and Verification Table

Requirement	Verification	Verification status
The ethernet subsystem must be able to receive TCP packets accurately through the RJ45 port into the MCU.	Setup and verify IP and MAC address of external PHY chip and MCU and then send a command from laptop to turn the LEDs on the microcontroller development board on.	N
As the system is designed for an ethernet at a PoE standard, the ethernet subsystem must be able to output 47.5 - 48.5 V DC to the PD controller.	Firstly, plug a PoE ethernet connection then, using a voltmeter, measure the voltage of the output diode bridge and verify that it is within the range of 47.5 V - 48.5 V.	Partially achieved

Power Subsystem Requirements and Verification Table

Requirement	Verification	Verification status (Y or N)
The power subsystem must be able to receive a 47.5 - 48.5 VDC output from the ethernet subsystem and step it down to 11.5 - 12.5 V output with a current of 5.5 - 6.5 A.	Input a 48V DC into the PoE PD controller. Then measure that the appropriate output voltage and current of the converter should be within the range of 11.5 - 12.5 V and 5.5 - 6.5 A respectively.	Y
Must be able to step down voltage received from the ethernet subsystem that is in the range of 47.5 - 48.5 V DC and step it down to an acceptable range of 3.0 - 3.6 V.	Input a 48V DC into the PoE PD controller. Then measure that the appropriate output voltage and current of the converter should be within the range of 3.0 - 3.6 V.	Y

Motor Control Subsystem Requirements and Verification Table

Requirement	Verification	Verification status (Y or N)
Must be able to accept SPI commands from MCU and convert the commands to step/dir pulses that are accurate enough for microstepping 256 microsteps per full step.	We will print the SPI responses from the TMC429 and TMC2660 chips on a serial monitor and match them with the datasheets, as well as microstep 51200 times and see if this matches a full rotation.	Y
Must be able to provide up to 3A per motor phase without going above 125° C.	We will be able to monitor the current provided using an ammeter and also monitor its heat using an infrared thermometer.	Partially achieved

Appendix B: Parts List

Part #	Manufacturer	Description	Quantity	Cost	Cost/Unit
C1608X7S2A473K080AE	TDK Corporation	CAP., 0.047uF, X7S, 100V, 10%, 0603	2	$0.58	$0.29
CL21B104KACNNNC	Samsung Electro-Mechanics	CAP., 0.1uF, 25V, 0805 (decoupling)	30	$1.20	$0.04
C0805C473K1RAC7800	KEMET	CAP., 0.047uF, X7R, 100V, 10%, 0805	1	$0.32	$0.32
100SEV22M8X10-5	Rubycon	CAP., 22uF 100V 20%	1	$0.73	$0.73
CC0603JRNPO9BN200	YAGEO	CAP CER 20PF 50V C0G/NPO 0603	2	$0.20	$0.10
106SML050M	Illinois Capacitor	CAP ALUM 10UF 20% 50V SMD	2	$1.02	$0.51
C0805C103J5GEC7800	KEMET	CAP CER 0805 10NF 50V C0G 5%	3	$0.90	$0.30
UMK212B7224KG-T	Taiyo Yuden	CAP CER 0.22UF 50V X7R 0805	1	$0.15	$0.15
50ZL100MEFCT78X11.5	Rubycon	CAP ALUM 100UF 20% 50V RADIAL	2	$0.82	$0.41
CL31A106MBHNNNE	Samsung Electro-Mechanics	CAP CER 10UF 50V X5R 1206	2	$0.60	$0.30
CL21B474KAFNNNG	Samsung Electro-Mechanics	r	1	$0.10	$0.10

CL21A106KAYNNNE	Samsung Electro-Mechanics	CAP CER 10UF 25V X5R 0805	1	$0.20	$0.20
ESW226M100AG3AA	KEMET	CAP ALUM 22UF 20% 100V RADIAL	1	$0.38	$0.38
CL21B105KAFNNNE	Samsung Electro-Mechanics	CAP CER 1UF 25V X7R 0805	1	$0.10	0.1
885342210001	Wurth Elektronik	CAP., 1000pF, X7R, 2000V, 10% 1808	2	$0.94	$0.47
C0805X7R201-103KNE-CT	Venkel	CAP, 0.01uF, X7R, 200V, 10%, 0805	4	$0.20	$0.05
PMEG10030ELPX	Nexperia	DIODE,SCHOTTKY,100V,3A,2-pin SOD-128,AEC-Q101	1	$0.50	$0.50
PTVS58VP1UP-115	Nexperia	DIODE, TVS, 58V, 600W, SOD128	1	$0.51	$0.51
L171L-GC	American Opto Plus LED	LED, Green, 0805	5	$1.90	$0.38
MM3Z12VC	onsemi	DIODE, Zener,12V, 200mW, SOD-323	4	$1.00	$0.25
J1B1211CCD	WIZnet	RJ45 Port	2	$6.84	$3.42
RMCF0603JT30K0	Stackpole Electronics	RES., 30k, 1/10W, 5%, 0603	1	$0.10	$0.10
RMCF0805FT1K00	Stackpole Electronics	RES., 1.00K, 1/8W, 1%, 0805	2	$0.20	$0.10

RMCF0805FT64R9	Stackpole Electronics	RES., 64.9, 1/8W, 1%, 0805	1	$0.10	$0.10
RMCF0805FT76K8	Stackpole Electronics	RES., 76.8, 1/8W, 1%, 0805	1	$0.10	$0.10
RMCF0805FT37K4	Stackpole Electronics	RES., 37.4, 1/8W, 1%, 0805	1	$0.10	$0.10
SWR201-NRTN-S04-SA-WH	Sullins Connector Solutions	Motor/Limit Connectors	2	$0.36	$0.18
NREC002SABC-M30RC	Sullins Connector Solutions	2.54mm 2-pin Headers	11	$1.21	$0.11
NREC003SABC-M30RC	Sullins Connector Solutions	2.54mm 3-pin Headers	2	$0.32	$0.16
PRPC002DAAN-RC	Sullins Connector Solutions	2x2	2		
PRPC004DAAN-RC	Sullins Connector Solutions	2x4	1		
PRPC005DAAN-RC	Sullins Connector Solutions	2x5	1		
PRPC040SAAN-RC	Sullins Connector Solutions	1x3	4		
PRPC002DAAN-RC	Sullins Connector Solutions	2x2 Male Headers (for PoE selection)	4	$0.60	$0.15

BUK7M12-60EX	Nexperia	N-MOSFETs	10	$9.80	$0.98
RMCF0805FT470R	Stackpole Electronics Inc	RES 470 OHM 1% 1/8W 0805	4	$0.40	$0.10
RMCF0603FT220R	Stackpole Electronics Inc	RES 220 OHM 1% 1/10W 0603	1	$0.10	$0.10
RMCF0805FT34R8	Stackpole Electronics Inc	RES 34.8 OHM 1% 1/8W 0805	1	$0.10	$0.10
RMCF0805FT140R	Stackpole Electronics Inc	RES 140 OHM 1% 1/8W 0805	1	$0.10	$0.10
RMCF0805FT46R4	Stackpole Electronics	RES., 46.4, 1/8W, 1%, 0805	1	$0.10	$0.10
RMCF0805FT330R	Stackpole Electronics Inc	RES 330 OHM 1% 1/8W 0805	2	$0.20	$0.10
RMCF0603FT174K	Stackpole Electronics	RES., 174k, 1%, 1/10W, 0603	1	$0.10	$0.10
RMCF0603FT52K3	Stackpole Electronics	RES., 52.3k, 1%, 1/10W, 0603	1	$0.10	$0.10
RMCF0603ZT0R00	Stackpole Electronics	RES., 0, 1/10W, 0603	2	$0.20	$0.10
RMCF0805FT8R20	Stackpole Electronics	RES., 8.2, 1/8W, 1%, 0805	1	$0.10	$0.10
RMCF0603JT3K30	Stackpole Electronics	RES., 3.3k , 1/10W, 5%, 0603	1	$0.10	$0.10

RMCF0603JG100K	Stackpole Electronics	RES., 100k, 1/10W, 5%, 0603	2	$0.20	$0.10
RMCF0805FT22R0	Stackpole Electronics Inc	RES 22 OHM 1% 1/8W 0805	4	$0.40	0.1
RMCF0805FT18R0	Stackpole Electronics Inc	RES 18 OHM 1% 1/8W 0805	5	$0.50	$0.10
RMCF0805FT150R	Stackpole Electronics Inc	RES 150 OHM 1% 1/8W 0805	1	$0.10	$0.10
RNCP0805FTD10K0	Stackpole Electronics Inc	RES 10K OHM 1% 1/4W 0805	4	$0.40	$0.10
WFMB2010R0750FEA	Vishay Dale	RES 0.075 OHM 1% 2W 2010	2	$2.76	$1.38
RMCF0805FT2K20	Stackpole Electronics Inc	RES 2.2K OHM 1% 1/8W 0805	4	$0.40	$0.10
CRGCQ0805F4K7	TE Connectivity Passive Product	RES 4.7K OHM 1% 1/8W 0805	1	$0.10	$0.10
RMCF2512JT3K00	Stackpole Electronics	RES., 3.0k, 5%, 1W, 2512	1	$0.30	$0.30
RMCF0603JT75R0	Stackpole Electronics	RES., 75, 1/10W, 5%, 0603	8	$0.80	$0.10
				$0.00	
B3SL-1002P		Push button switch	1	$0.90	$0.90

7490220122	Wurth Elektronik	Transformer	1	$6.77	$6.77
LT4321HUF	Analog Devices	PoE Diode Bridge Controller	1	$5.11	$5.11
LT4293HMS	Analog Devices	PoE PD Controller	1	$6.15	$6.15
STM32F407 VET6	Songhe	MCU	1	$19.88	$19.88
NCP1117LPS T33T3G	onsemi	3.3V DC/DC Converter (Linear)	1	$0.45	$0.45
E48SC12010 NRFA	Delta Electronics	12V DC/DC Converter (Switching)	1	$33.39	$33.39
DP83848	Waveshare-M odule	Ethernet PHY	1	$13.98	$13.98
TMC429-I	Trinamic	Motion Controller	1	$12.55	$12.55
TMC2660C-P A	Trinamic	Stepper Driver	1	$7.32	$7.32
SXO53C3A0 71-50.000MT	Suntsu	50MHz Oscillator (for PHY)	1	$0.52	$0.52
AS-8.000-20	Raltron Electronics	8MHz Oscillator (for MCU)	1	$0.18	$0.18
		Total Cost		$146.84	

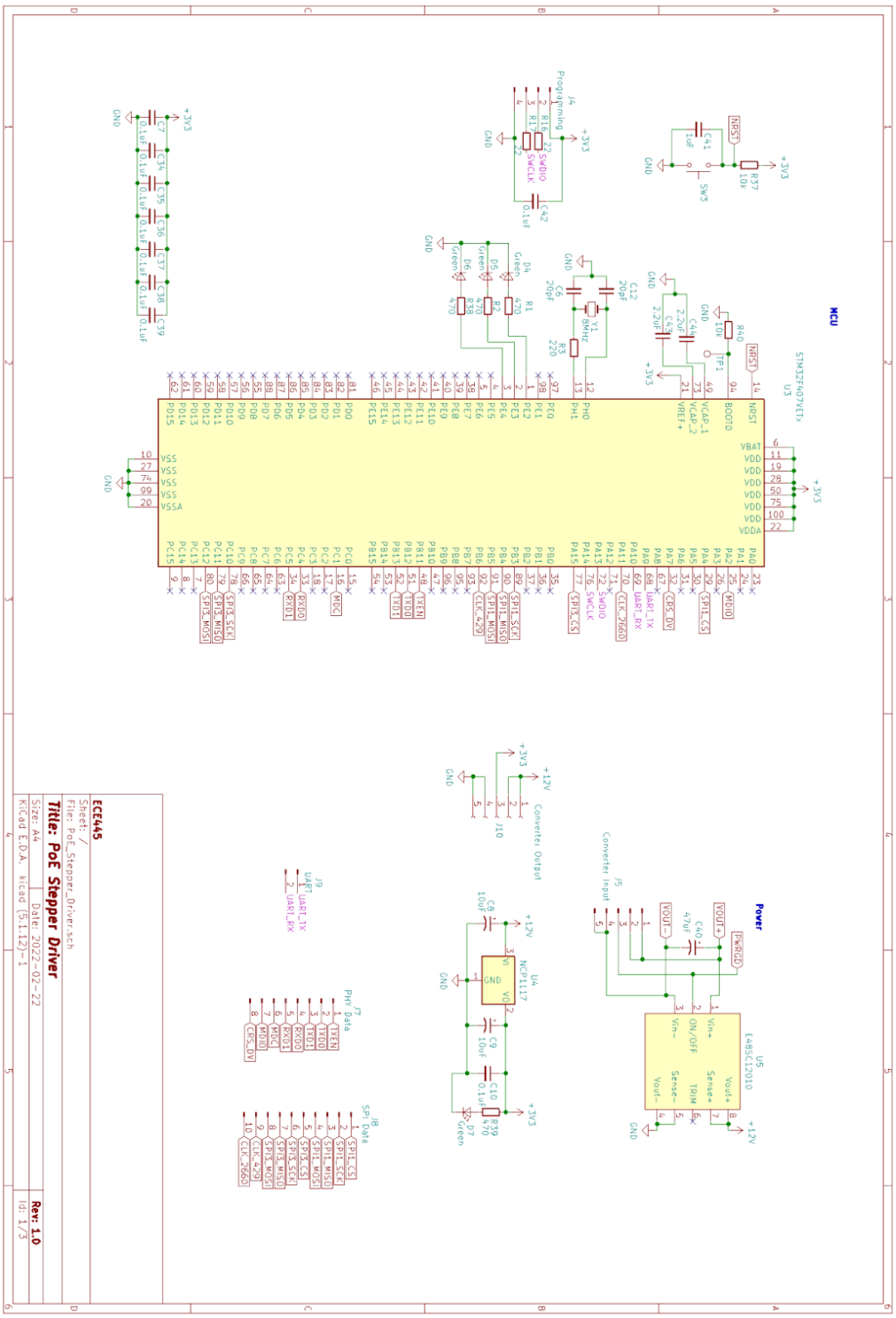
Appendix C: Schedule

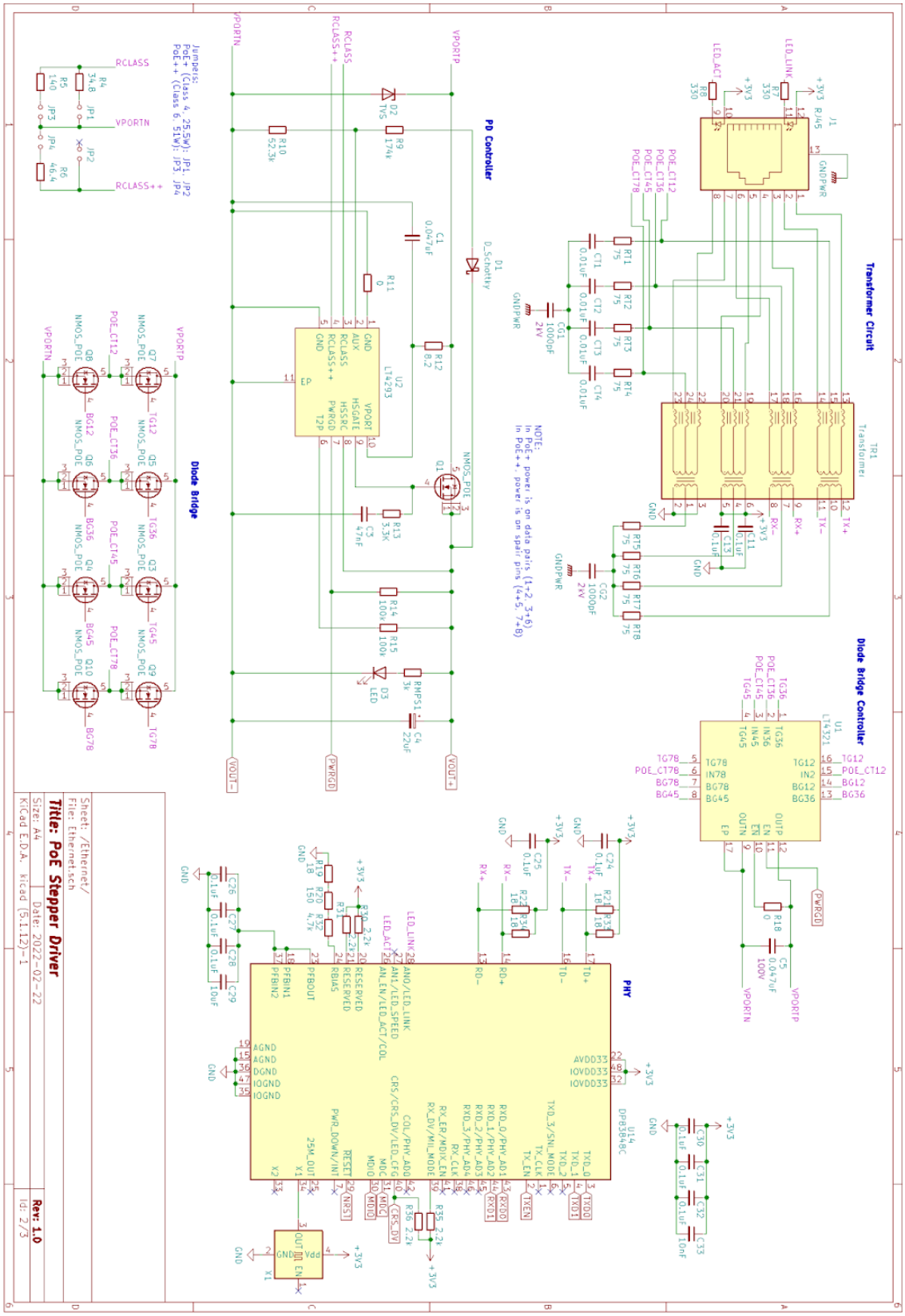
Week	Bryan	Putra	Armando

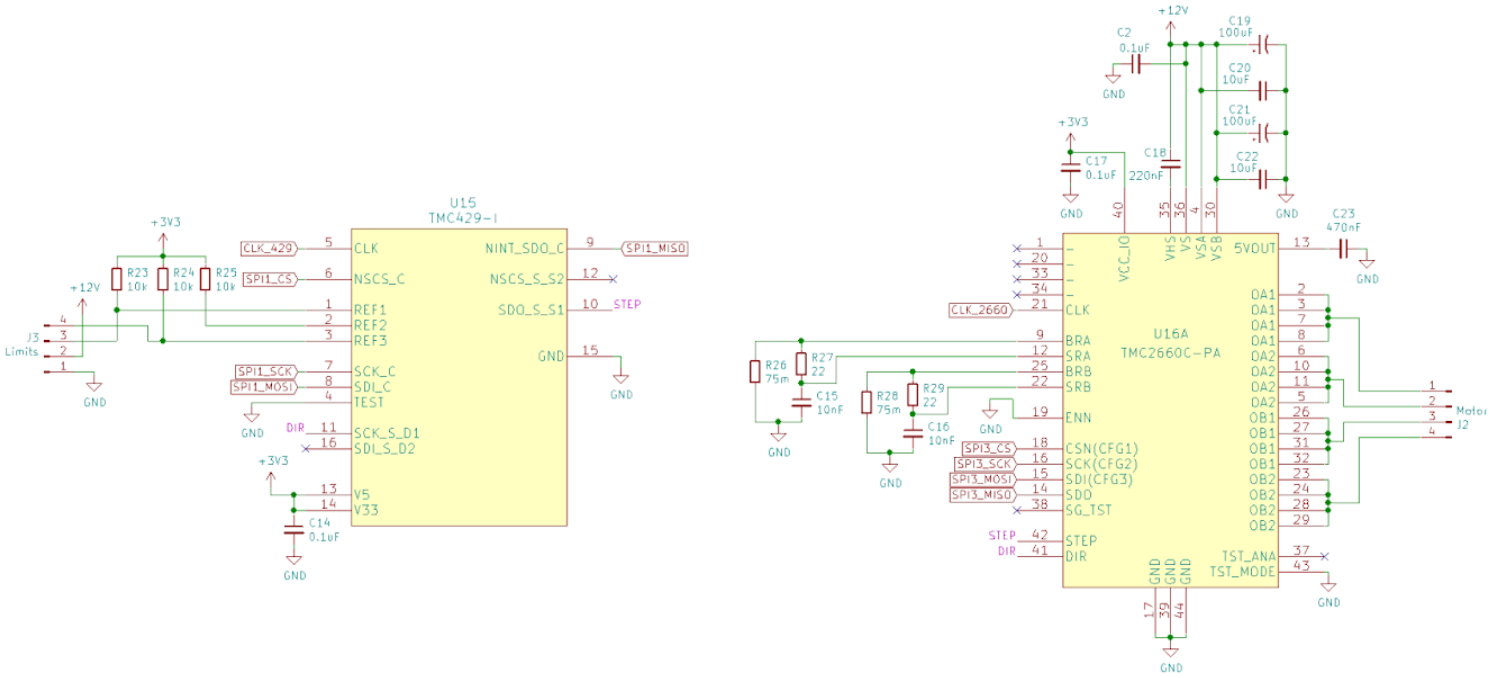
2/28	Finish 1st draft of PCB Layout	Research on ethernet communication for STM32F4 chips and how to run TCP transmission	Research on configuring and controlling TMC2660 and TMC429 chips
3/7	Finalize PCB and submit order	Run ethernet and test examples on an STM32F4 evaluation board with an ethernet PHY chip board	Research on configuring and controlling TMC2660 and TMC429 chips; examine how TMC API interfaces with STM32 or C code.
3/14	Spring Break	Spring Break	Spring Break
3/21	Solder parts onto the board and assemble the project	Debug ethernet software and get it to work for ping tests	Solder parts onto the board and assemble the project
3/28	Verify the power board against requirements Debug PoE circuitry to troubleshoot PoE negotiation problems	Verify the power board against requirements. Continue debug ethernet software and look investigate it at a lower level Help troubleshoot PoE negotiation problems	Verify the power board against requirements Start writing code for the motor control subsystem while referring to the TMC API.
4/4	Continue debugging PoE subsystem so that it functions for PoE+ and PoE++ standards and verify it against requirements	Debug PoE subsystem so that it is functional for PoE+ and PoE++ standards and verify it against requirements	Continue writing code for the motor control subsystem while referring to the TMC API
4/11	Write code to enable sending SPI commands via MCU pins Research and write code for configuring and commanding TMC2660 and TMC429 chips using SPI commands.	Write code to enable sending SPI commands via MCU pins Research and write code for configuring and commanding TMC2660 and TMC429 chips using SPI commands.	Write code to enable sending SPI commands via MCU pins Research and write code for configuring and commanding TMC2660 and TMC429 chips using SPI commands.
4/18	Debug software for configuring and commanding TMC2660 and TMC429 chips using SPI commands. Verify motor control subsystem against requirements.	Debug software for configuring and commanding TMC2660 and TMC429 chips using SPI commands. Verify motor control subsystem against requirements.	Debug software for configuring and commanding TMC2660 and TMC429 chips using SPI commands. Attempt to get ETH phy chip to work with software Verify motor control subsystem against requirements.

4/25	Debug ethernet software Implement UART communication for demo purposes to substitute ethernet communication Demo	Debug ethernet software Implement UART communication for demo purposes to substitute ethernet communication Demo	Debug ethernet software Implement UART communication for demo purposes to substitute ethernet communication Demo
5/2	Final presentation	Final presentation	Final presentation

Appendix D: Full Schematics



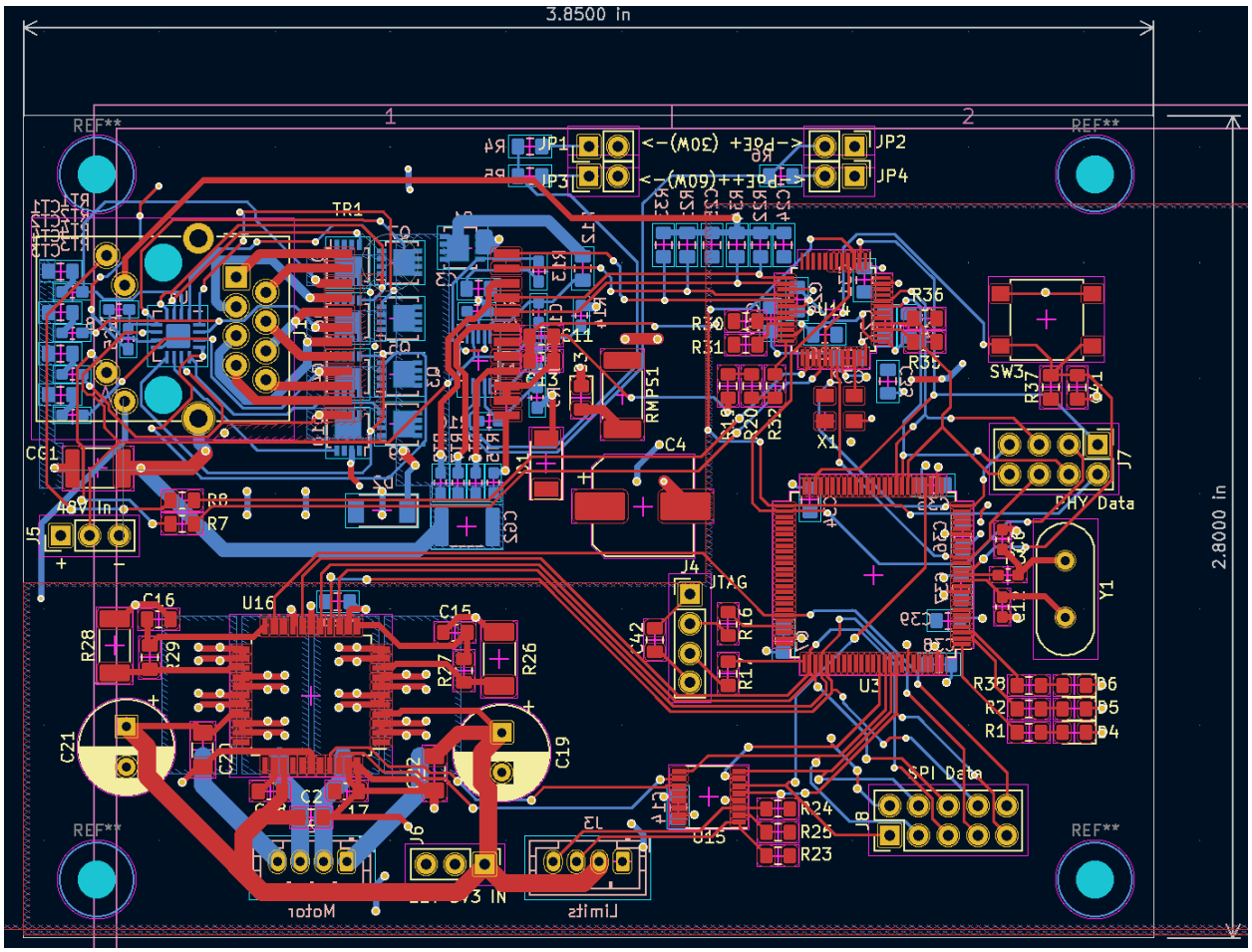

Transformer Circuit
PD Controller
Diode Bridge
Diode Bridge Controller
PHY
Title: PoE Stepper Driver
Rev: 1.0
Sheet: /Ethernet/
File: Ethernet.sch
Size: A4
Date: 2022-02-22
KiCad E.D.A. kicad (5.1.12)-1
Id: 2/3



Sheet: /Motor_Control/		
File: Motor_Control.sch		
Title: PoE Stepper Driver		
Size: A4	Date: 2022-02-22	**Rev: 1.0**
KiCad E.D.A. kicad (5.1.12)-1		Id: 3/3

Appendix E: PCB Layout

Main Board:



Power Board: