Controllable, User-Friendly 3-Phase Inverter

By

Alex Chirita

Shyam Peden

Johnathan Vogt

Final Paper for ECE 445, Senior Design, Spring 2026

TA: Frey Zhao

6 May 2026

Project No. 81

Abstract

This report covers the design and implementation of a controllable, user-friendly 3-phase inverter for ECE 445. This project looks to help test small machines that operate in fault conditions. Sometimes, 3-phase systems are not ideal, causing an unbalanced system. Our solution can help test these systems, opening up new research and commercial opportunities. We take in a set input voltage that takes the place of a potential solar panel that would have variable voltage. From there, we scale up the voltage, calculate the switching logic, and feed it into 3 H-bridge systems. The resulting output is 3 modified sinusoidal waveforms that form a 3-phase power system. There are two encoders that allow the user to modify two of the three phases.

Contents

1. Introduction

1.1 Problem

While normal 3-phase AC power systems operate with consistent phase differences of 120 degrees, these systems are not always perfect. There may be occasions (fault conditions) where the power system becomes unbalanced. In order to test small machines under these conditions, one might want to create controllable AC waveforms with adjustable phase angles.

1.2 Solution

We created an inverter system that is capable of creating three AC waveforms with controllable phase angles. Phase A serves as the reference 0-degree phase, while the B and C phases are controllable with respect to this reference phase. This is achieved using analog control, likely via potentiometers. The PCB functions as a normal 3-phase switching inverter, with switching control handled by the microprocessor, which takes input from the encoders to control the output AC waveforms. There are 5 main subsystems: input stage with a boost converter, 3 IGBT H-bridges, Encoders, CONFIRM button as the user interface, and a TI-C2000 microcontroller, as it has enough PWM channels and high resolution timers, whose firmware includes the user input control, and the power control for the bridge.

1.3 High-Level Requirements List

Our high-level requirements in order to consider the problem solved are shown below. In addition, qualitatively, we would like the user interface to be able to properly display all 3 waveforms without fail, with Phase A corresponding to a 0-degree phase and Phases B and C properly updating. The inverter is tested on a Wye-Connected resistive load.

1. The Output RMS voltage is 120V ±5%.
2. The inverter outputs up to 0.5A across 3 phases, corresponding to 60W total power output and 0.28 A per phase, corresponding to 33.6W power of a single phase
3. The microprocessor responds in less than 1 second to input or phase adjustments, switching the DC-DC converter as necessary to maintain constant amplitude and the inverter to maintain phase.

2 Design

2.1 Block Diagram

Below is the block diagram for our proposed solution.

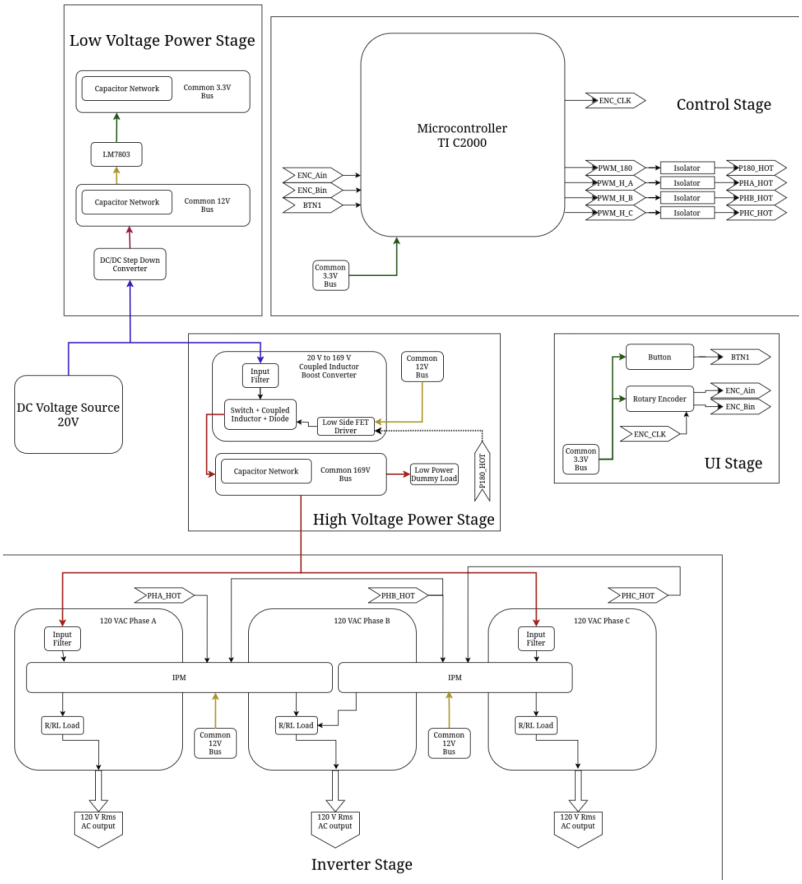


Figure 1: Block diagram of our proposed solution

2.2 Subsystem Overview

2.2.1 Boost Subsystem

The main purpose of this project is the inverter stage, while the input is a DC power source that mimics a solar panel. We did not focus on it being powered from a solar panel during the semester, and left it as a modification/addition to the inverter part for further development. The input voltage needs to be converted to a setpoint and fed into a common DC bus. This is done with a boost converter. A good quality of this system is the freedom to choose which variables to control. The circuit were able to respond to quick changes in input voltage,

but this semester, we used a constant DC power supply instead of a solar panel to reduce cost and complexity. Therefore, the boost converter is run by a switching algorithm with a fixed input voltage. Later, the converter could be used as an MPPT if needed.

The plan A was to use a flyback topology because a regular synchronous boost, in reality, can give a maximum boost ratio of around 4, where it drops down later. Flyback would be able to utilize a transformer to increase the multiplier further without the limitation of duty ratio. The flyback converter has an output voltage equal to (D)/(1-D) * (Ns/Np).

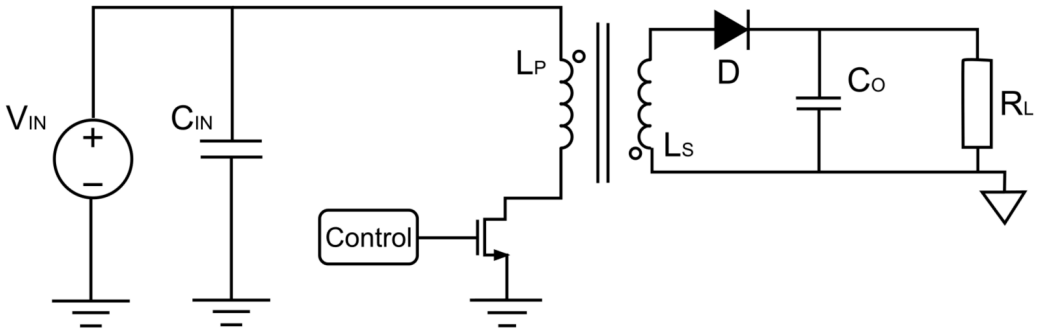


Figure 2: Flyback converter

The plan B was a coupled inductor boost converter. It is not very complicated and provides more boost capability than a regular boost, with output voltage Vout = (1 + Ns/Np * D) / (1 - D) * Vin. We can easily play with the inductor turns ratio to ease the duty ratio.

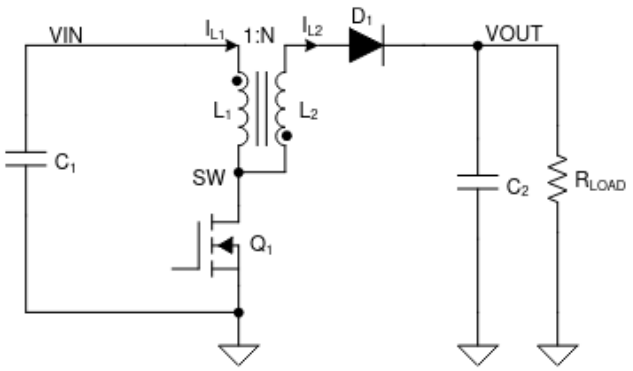


Figure 3: Coupled inductor boost converter

Controlling the voltage on-demand is not in the scope of the project and will require more expertise in control theory as well as full galvanic isolation between high voltage side and microcontroller side, because using a simple voltage divider to fit the sensing within 3.3V limit of the MCU might destroy it from voltage spikes during transistor switching. Isolated OP-Amps

will be a future work outside of the class and will require complete redesign of the PCB and 2 independent ground planes. We will be running all of our converters as well as inverter phases in an open-loop configuration and manually setting the operating point.

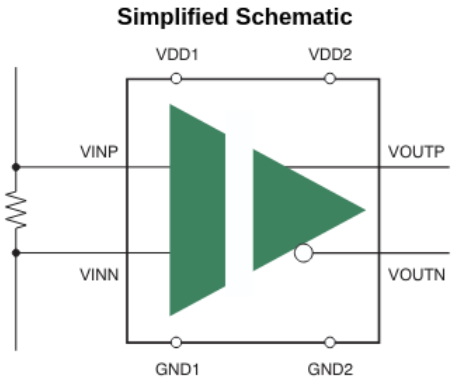


Figure 4: Isolated OP-Amp

We decided to use our Plan B, the asynchronous coupled inductor boost converter. The reason we chose this over our initial flyback converter plan was that the topology required a large number of parts that were all analog. We came to the conclusion that the best topology would be the coupled inductor boost converter with the turns ratio of 20:60. We operated the converter at 100 kHz, which was in our range of frequencies that we were brainstorming on using. Once we had our topology figured out, the next step was to construct the converter. Since the process for making the converter work was long and full of bugs and errors, the final revision was made long after the fourth round of PCB orders, and we used a so-called LIT [Laser (printer) - Iron Technology] to make a DIY PCB, which is explained in more detail in Appendix B.

Once everything was finally put together, we were able to test it using our TI C2000 microcontroller. We adjusted the frequency it ran at to 100 kHz and ensured that the duty ratio was set to two-thirds, or 67%. After powering the boost converter, we were able to see a square wave that was switching at 100 kHz, just as we intended. Below in Figure 5 is the image of our PWM square wave. As you can see, we got a square wave with a period of 1 microsecond. This corresponds to 100 kHz.

Figure 5: PWM square wave for boost converter

Using the boost converter PWM, we now had to test the performance of the boost converter. We powered the boost converter using the Keithley Instruments in the power lab of the ECE building. For the converter, we used a load of 500 ohms. We started at 5 V of input, and slowly stepped it up all the way to 20 V. Using the Yokogawa digital power meter, pictured below, we got an initial voltage reading of around 44 V. Once we stepped it up all the way, our output voltage jumped up to 174.15 V. We also measured the current of 0.3444 A, giving us an output power of 60 W. We were also able to calculate an efficiency of around 87%.

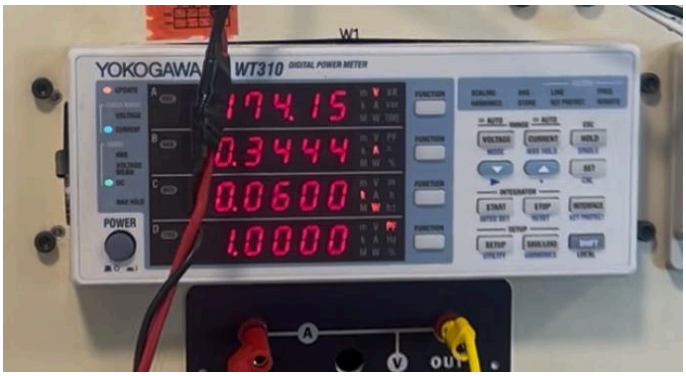


Figure 6: Boost Converter Output Measurement

Overall, we were able to construct our boost converter that scaled the voltage up by around 8.7. In addition, it generated minimal heat. We had placed a heat sink to account for any heat that we would generate, but we were extremely pleased to find that that seemed unnecessary. We were able to touch the heat sink, and it felt cold. Finally, the converter maintained a high efficiency regardless of the output voltage. This is shown in Figure 7, which is a plot of the efficiency based on the voltage being outputted. As you can see, the efficiency stayed fairly high regardless of output voltage. This converter satisfied our design requirements as we got an output voltage greater than 120 Vrms, which corresponds to around 169 V.

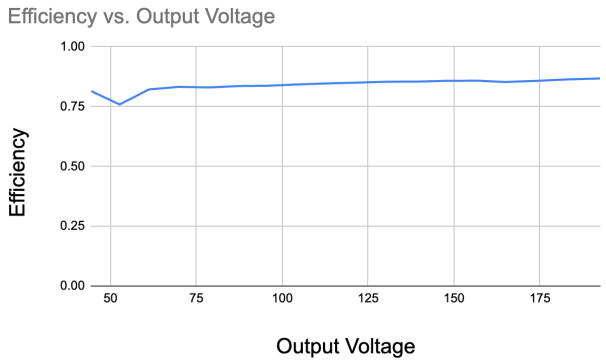


Figure 7: Efficiency vs. Output Voltage Plot

2.2.2 Bridge Subsystem

The bridge subsystem contains three IGBT H-bridges, each corresponding to one of the phases. Each of the phases have a similar layout since the control is only achieved by the gate signal, which is fully generated by the microcontroller. We decided to go with an H-bridge because it's a good middle ground between multi-level bridges and a half-bridge. The H-bridge allows us to generate a good-quality modified sine wave when averaged out and filtered with series inductors, which are external to the board. The modified sine wave is generated from -Vmax to +Vmax (a few volts lower than the boosted DC bus), and the sign is decided by using a proper pair of transistor switches from the 4 available. Each switch pair (usually called a half-bridge) has complementary low-side and high-side gate driver signals, which are handled by the microcontroller. Each phase has an LC low-pass filter at the end to reduce switching harmonics. Our target output from the bridge system is 120 Vrms with a tolerance of 5%. While we could achieve this output using the full boosted DC voltage, safety concerns led us to test the H-bridge at a lower voltage.

One of the practical ways to limit the size of the H-bridges is to use dedicated motor controllers integrated circuits. 3-phase AC motors are controlled by 3 strings of half bridges in the vast majority of cases, so 2 IPMs were enough to be wired as 3 H-bridges, while providing very high voltage ratings (600+ V). The IGBTs included in these half-bridge ICs are more lossy than other transistor options but provide unmatched flexibility and power capability.

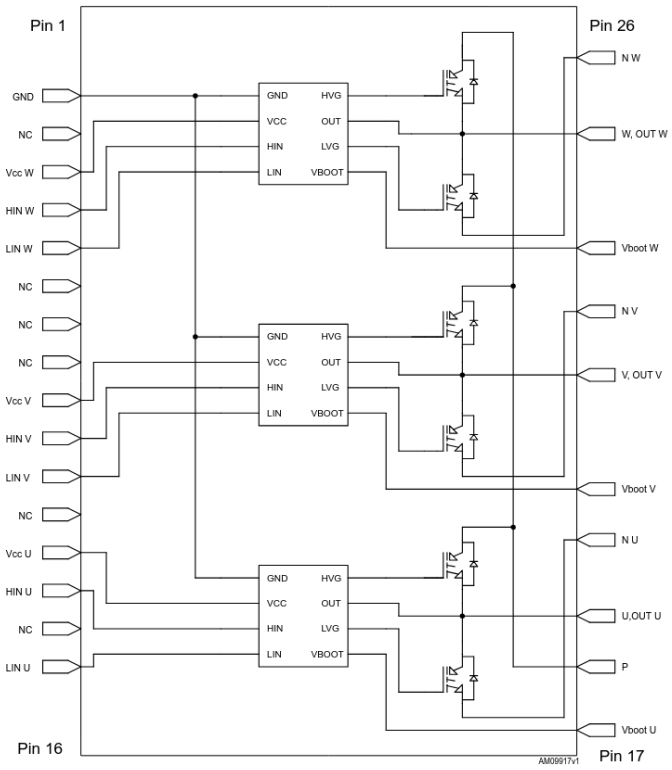


Figure 8: IGBT IPM for 3-phase motors

The outputs of 2 half bridges are connected together across each 1-phase load, while bootstrap circuits are already included in the IPM, so apart from the power to the chip itself, we only had to provide signal-level PWM, which can come directly from the optocouplers. The input voltage to the transistors is taken directly from the output of the boost stage. This gives us 2 free variables to control. We can choose our own Vmax for the peak of the modified sine wave by setting D at the boost stage, and we fully control PWM timing of the modified sine generation, where we can artificially decrease the RMS voltage of the wave through timing.

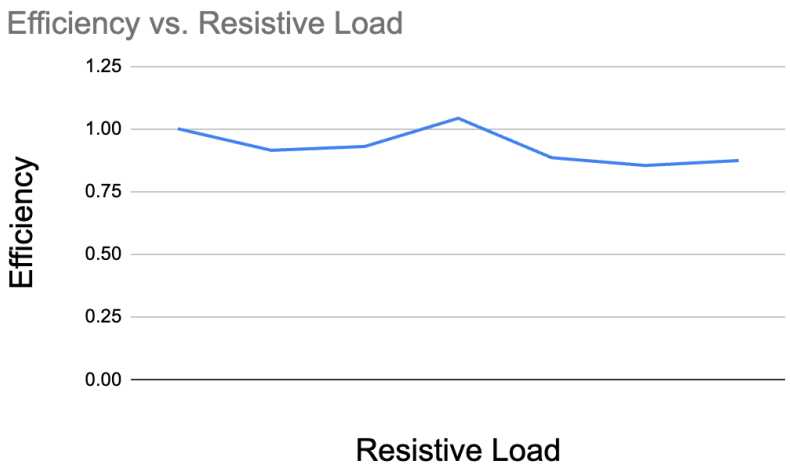


Figure 9: Efficiency vs. Resistive Load Plot

Our H-bridge subsystem was effective as we were able to generate 3 waveforms that depicted a common 3-phase waveform. We were able to increase and decrease the voltage and see the amplitudes of all 3 waveforms change accordingly. We also were able to test the efficiency of our system across a variety of loads. Figure 9 below shows the graph of efficiency with respect to the resistive load we were using. Our voltage stayed constant at close to 40V for this example. As you can see, our efficiency was able to stay high regardless of the resistive load that we were using.

2.2.3 User Interface Subsystem

The user interface consists of two encoders and a confirm button. The user can use the encoder and confirm button to navigate the user interface, where the phase angle will be set for phases B and C in relation to phase A, which is static. Users can also choose to use an autoset, where the microcontroller will default to 120 degrees between each of the phases. We made the user interface to update as the input changes. More specifically, we made the interface to update within around 250 ms of a change in input.

The encoder is connected to a clock, as shown in Figure 10. The reason for this is that the output of the encoder is used as an input for the input control subsystem. Since the microcontroller needs to constantly update the input value and adjust accordingly, the clock assists in rapidly changing the encoder output. It is also important to note that we have 2 encoders since we want to be able to adjust both Phase B and Phase C.

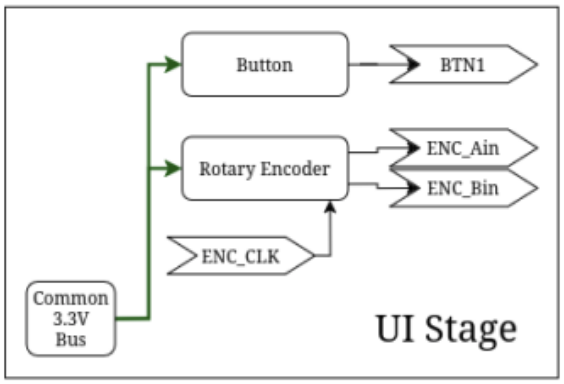


Figure 10: User Interface Subsystem Block Diagram

We wanted to incorporate a 12C display into our design as well. Our original plan was to program our microcontroller to display phases B and C on the LEDs. However, we were exploring how we could implement the 12C programming, and given our microcontroller software, we would struggle. The display is significantly more friendly with Arduino MCUs as opposed to TI microcontrollers. If we had more time, this definitely would have been an additional feature to display, as it would have further improved our project visually.

2.2.4 Input Control Subsystem

Our input control subsystem had a fairly simple flowchart. The microcontroller reads the input, the code checks if the input is in an acceptable range, and it is finally passed into the switching control subsystem.

The method that we employed to read the encoders is quadrature decoding logic. We need the encoder to read a numeric value, but the outputs of a 5-pin rotary encoder are only digital. So, we needed to implement this logic to give us a value that can span from -180 to 180 degrees. We had the microcontroller rapidly poll the A and B square waves from each encoder using GPIO reads. For every time it checks, we want to see if the A and B waves are different. So, we can easily generate a total count of the number of times these waves differ. This count corresponds to the phase we want to pass into the switching control subsystem. Using simple print statements, we were able to verify the functionality of both encoders.

Once we get this information from the reading encoder logic, our code is straightforward. We use if statements to keep the encoder input in the bounds of -180 to 180 degrees. Then, once the button is pressed, the phase values update.

2.2.5 Switching Control Subsystem

We used MCU for the control of the most onboard SMPS, except the 5V bus. For our switching logic, we needed every phase to have 4 signals, meaning 12 total outputs. For each

phase, we had a phase 1 high, a phase 2 high, and their respective complements. We need these switches to work in complement with each other to avoid shorting the circuit. We used a system of states for our switching control subsystem. 2 states signified the two ways of achieving no voltage, a positive voltage state, and a negative voltage state. If everything goes as planned, we end up having each switch operate at a duty ratio of 50%. The only difference between the switches is the time that each one is turned on. The program that we wrote uses a timer module to achieve this. Starting with the computing phase A, we set a delta value that we want our modified sine wave to go positive. This value is purely up to us because our phase A is the reference phase angle, so we chose a delta of 25 degrees. After this, based on how far into the timer we are, the program sets the phase A_1 high and phase A_2 high to either 0 or 1. Similarly, we did a similar operation for phases B and C. The only difference is that we are now getting a desired phase difference angle from the input control subsystem. The default values that we programmed were 120 and -120 for phases B and C, respectively. Figure 11 shows the default output waveforms before any input adjustments.

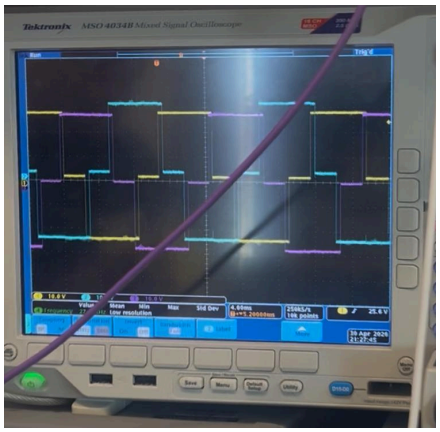


Figure 11: Final 3-Phase Output Waveforms

The primary difference between our phase A and our other phase calculations was that we are shifting the state transitions by a certain number of ticks. We converted the phase into a number of ticks and simply had to change our if statements to account for the added or removed phase. The most difficult part was that we had to include wrap-around logic. If the timer plus a delta amount of ticks is above our timer period, it will reset to the start of the period and act accordingly. Based on this logic, we were able to produce switching logic for a 3-phase system.

2.2.6 Low-Voltage Systems

The MCU and user interface must be isolated in order to be safe to operate. While proper design is a must, any failure can potentially inject high voltage into a non-isolated system. For this reason, we used a DC/DC integrated circuit to step down the 20 V input to 5V.

The 5V bus is connected to a 3.3V linear regulator. This requires headroom in input voltage, so we went with 5V. The 3.3V bus feeds the MCU and input devices.

12V for the gate drivers is generated using its own integrated buck converter, which comes inside a small package and does not require any control or coils. All the circuitry is provided by the manufacturer inside the package. The input voltage could vary in this case without disrupting the voltage supply to the gate drivers and motor controller ICs.

3. Design Verification

3.1 Boost Subsystem

Our boost converter satisfied all of our requirements. We sought to scale the input voltage by at least 7. At one point, we managed to scale the voltage by a factor of 9.425, 20V input to 188.49 V output. This was significantly higher than what we had needed, so we had a good amount of room to play with. We decided to use a high load, which gave us a final scale ratio of 8.7. Again, this met our requirement that we had set out. In addition, we used the proper precautions when dealing with the converter. We avoided rapidly increasing the input voltage from 5V to 20V to avoid any ethical violations.

Requirements	Verifications
The converter is able to get a step-up ratio of at least 8.48 to achieve 120V * $\sqrt{2}$ = 169.7V	We managed to scale the voltage by a factor of 8.7 at high load
The converter is treated with caution when it is being powered	We treated the converter with caution by gradually increasing the voltage and taking electrical safety precautions

3.2 H-Bridge Subsystem

We also met all of our requirements for our bridge subsystem. We ran into a small hiccup with this system as both of our isolators and our ICs got fried. This resulted in us having to replace these parts. However, once we resoldered everything, we were able to display our desired waveform. We sought out an output voltage of 120 Vrms, which corresponds to around 169V. Under high load, we were able to get 174.15V. So, we surpassed this requirement by about 5V. Even under the lower load, we were able to get an even higher voltage of around 189V. The amplitudes of our output waveforms were also able to be adjusted according to a change in the DC voltage.

Requirements	Verification
Creates 3 waveforms representing the 3 phases	Displayed 3 accurate waveforms that correspond to phases A, B, and C
Output voltage is as close to 120 Vrms as possible	Achieved a voltage of 174.15 V, surpassing 120 Vrms = 169.71 V

3.3 User Interface Subsystem

Our requirements for the user interface subsystem were the easiest to check. All we had to do was make sure these parts worked and were given the appropriate voltage. We were sure not to power them with 5V to avoid any issues with frying something on the microcontroller. Instead, we used the 3.3V port.

Requirements	Verification
Encoders are powered properly	Encoders received the 3.3V and were grounded properly
Correct signals are sent into the microcontroller rapidly for the input control subsystem	Verified that the correct signals are sent into the microcontroller

3.4 Input Control Subsystem

The requirements that we had set for our input control subsystem were all met. Using our quadrature decoding logic, we were able to read our encoders correctly. To ensure that we were getting the correct value, we used print statements for phase B and phase C. We were also able to see that the value stayed within the -180 to 180 degree phase.

Requirements	Verification
Input received from the encoders	Verified using print statements
The input is managed and passed into the switching control subsystem	Correct values were passed into the switching control subsystem
Inputs should stay within the range of -180 to 180 degrees	Inputs did not stray away from the input range despite pushing the encoder past the bounds

3.5 Switching Control Subsystem

Our switching control logic was able to accurately account for phases B and C, and send the correct signals into the bridge subsystem with respect to time. As a result, we met all of the design requirements that we had created at the start of the semester. We verified all of this using the oscilloscope that we had access to. For the switching square waves that were supposed to be complements of each other, we were able to verify that the two waves operated exactly opposite to each other. We were also able to account for the wrap-around logic. We had phases B and C sometimes staying in a given state from the end of one cycle to the start of the next one. However, this was not a problem for us, and the switching signals still operated as we wanted them to.

Requirements	Verification
Switching logic should produce a square waveform based on our software	Oscilloscope verified that all 12 switching waveforms were accurate
Switching logic accounts for phases B and C and passes the right values into the bridge subsystem	Waveforms were completely accurate, and wrap-around logic was perfectly executed

4. Costs

4.1 Parts

Our total cost on parts was $83.54 not including shipping, reordering components due to failures, and TI Launchpad, which the team kept for future projects. Below is a complete list of parts that we used for the project:

- IRS2186STRPBF
- TI C2000 F2800157SPN
- LM1085IT-3.3/NOPB
- ECS-200-18-30B-AGN-TR
- CC0603FRNPO9BN270
- STGIPN3H60A
- R-78E12-0.5
- RLS-397-R
- EFDKA45K806F024DH
- B66368W1020T001
- B66368A2000X000
- B66367G2000X187
- B66367G0000X187
- IPP60R037CM8XKSA1
- C3D08065A
- KAPTON-TAPE20MM
- Generic 5-pin rotary encoder (Students have some available)
- Generic 2-row LCD display (Students have some available)
- Generic push button (Students have some available)
- 0805 Resistors and Capacitors (Excess at Electronics Services Shop)
- THT Resistors and Capacitors (Excess at Electronics Services Shop)
- Generic toroidal magnetic cores (Excess at Power Electronics Lab)
- Generic TO-220 Heatsink (Sourced from an old workstation power supply)
- 470 uF cylindrical electrolytic capacitor (Sourced from an old workstation power supply)

4.2 Labor

Upon doing some research, we came to the final number that the average ECE graduate from UIUC will make around $50 an hour. Using this number, we can compute the total labor cost per group member for this project. We also need the total time needed to complete the project. A safe estimate for the number of hours per week we worked on the project is 8 hours per week. Given that there were seven weeks of work after our design document, this gives a total labor cost of $8400.

5. Conclusion

5.1 Accomplishments

Overall, we met the goals that we had set at the start of the semester with our project. We were able to design and construct a working 3-phase inverter. It was able to rapidly update according to changes to the input voltage and the phase. In addition, we were able to create an extremely efficient boost converter that produced minimal heat. Despite the issues that we ran into with fried ICs, parts on backorder, etc., we were able to deliver a finished product that we are very proud of. Below is a picture of our final setup with the converter, PCB, MCU, and stack of resistor boxes.

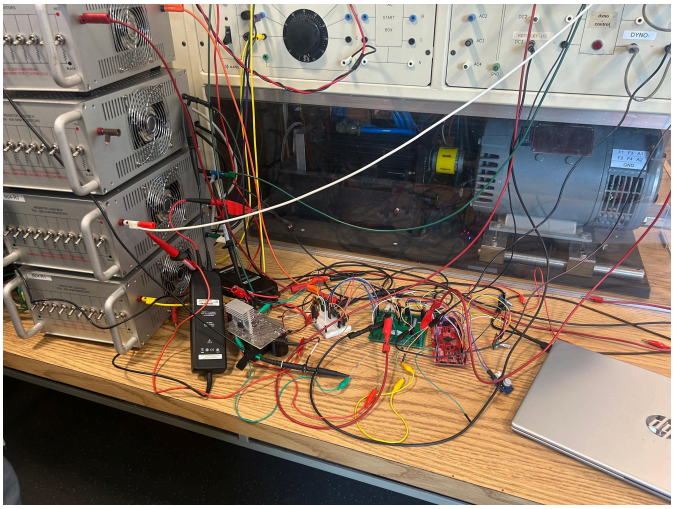


Figure 12: Final Demonstration Setup

5.2 Uncertainties

The biggest uncertainties were our lack of an enclosure and that we were not able to fit everything onto a single PCB. These problems somewhat go hand in hand with each other. We had designed and printed an enclosure, but we realized that it was not large enough to hold all of our subsystems. So, we had to demo our project without the enclosure. We also wanted to keep everything on a single PCB, but this was not realistic, as we would not have been able to adhere to the maximum size limitations for the PCB orders.

5.3 Ethical Considerations

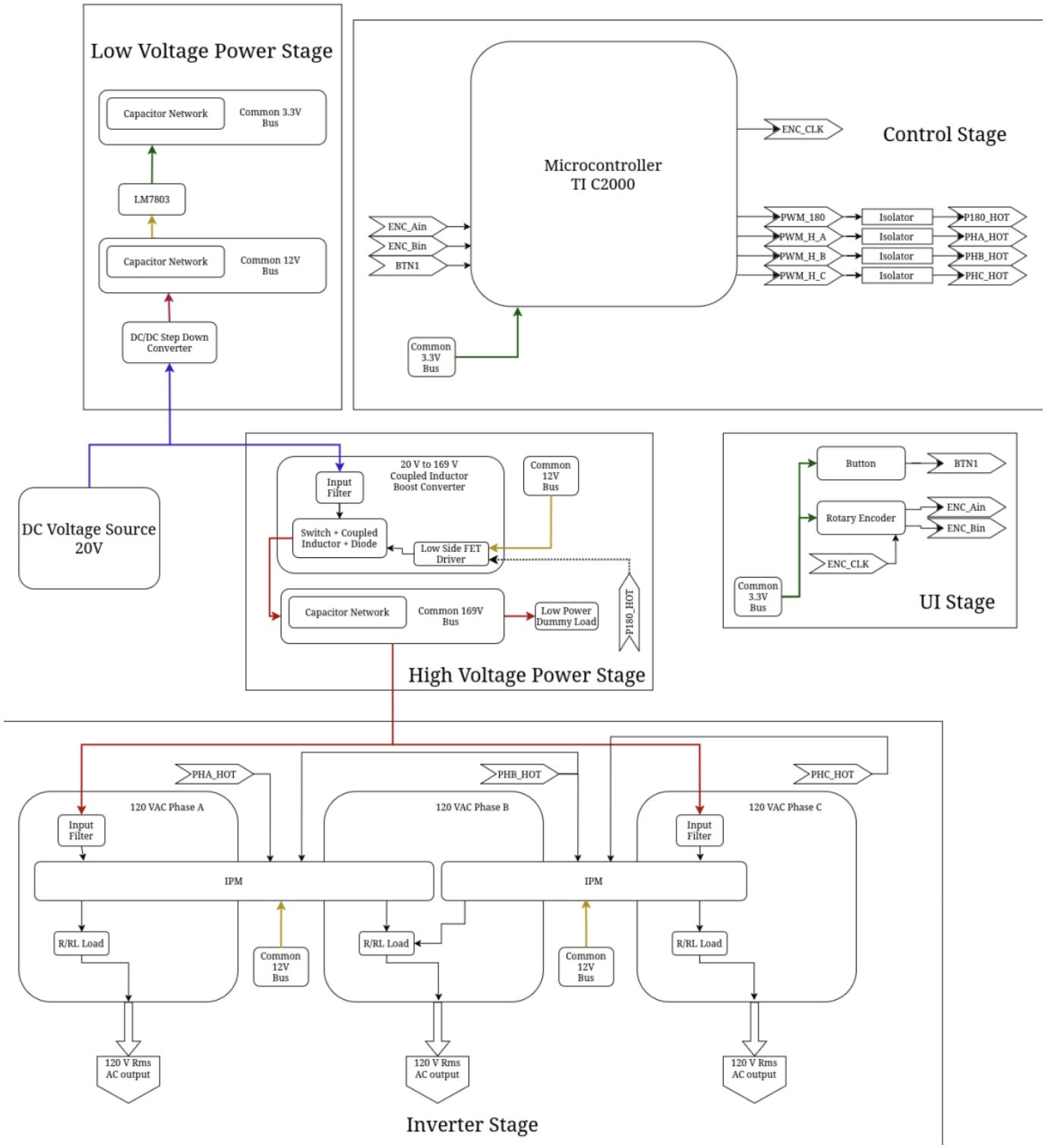
We adhered to all of the ethical codes of conduct during our project. The primary potential issues that we could have run into were the risk of the user getting harmed and the risk of a lack of integrity. However, we were able to avoid both potential ethical violations. We upheld integrity with every aspect of our project, and we took the proper precautions to ensure that the user would not be put in any sort of harm.

We also maintained safety throughout the semester. Since we were dealing with high voltages, we had to take heavy precautions when designing and working with our circuit. By taking proper precautions, we lowered the risk of electric shock, electrocution, thermal burns, etc. By completing our high-voltage certification, we were given instructions on proper high-voltage etiquette and improved our safety compliance.

5.4 Future Work

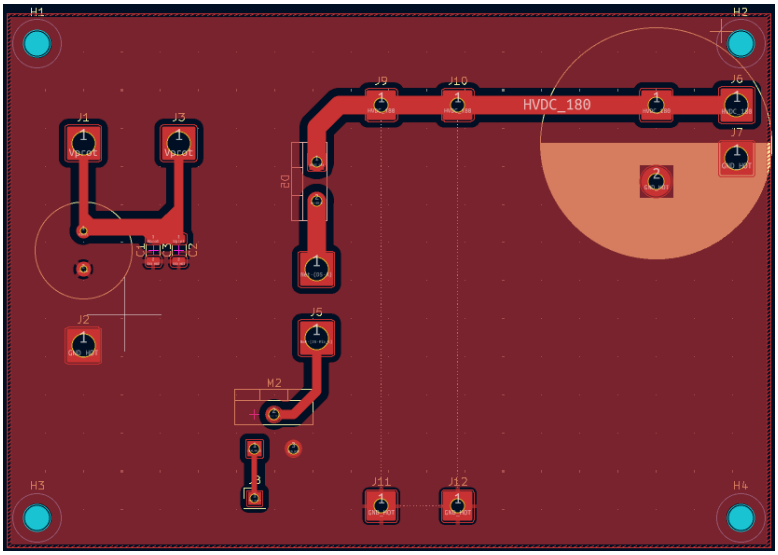
One next step that we could try to implement in the future is using a larger inductor when measuring our output voltage. Currently, it resembles a modified sine wave. If we wanted a really smooth sinusoid wave, we would just have to use a very large inductor and measure the output or use SPWM generation with a smaller inductor at the output. Another potential feature we could add in the future is the use of the I2C display to show the user phases B and C. One final potential future work would be the integration of closed-loop DC/DC conversion. This would involve the microcontroller reading the AC output and producing the needed DC voltage. This was out of the scope of the project for this semester, but it is definitely something that can be incorporated in the future.

Appendix A. Full-Size Block Diagram



Appendix B. Custom PCB Fabrication

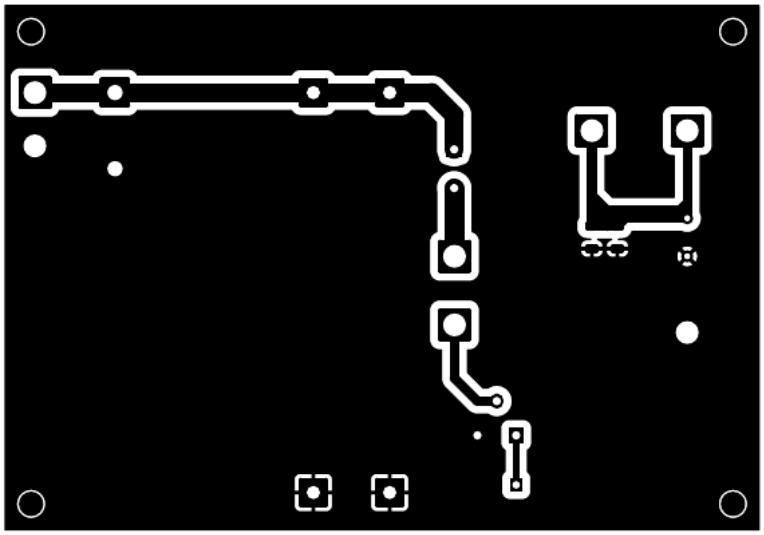
A DIY PCB is less technologically advanced than ordering from a factory. Most of the fiberglass copper plates are one-sided, and the copper plate is very thin. This requires bigger margins between the ground plane and traces as the entire plate will have to be tinned.



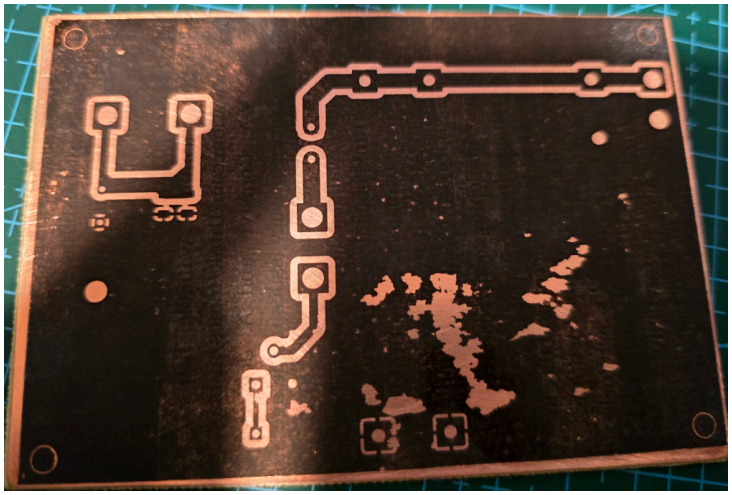
It also means that the density of the traces will have to be small, and these kinds of PCBs are good for individual parts of the project or smaller projects altogether.

There are 3 parts to the process: Masking the copper, dissolving the copper, and final preparation.

The mask must be inverted and mirrored in order to be transferred onto the copper plane



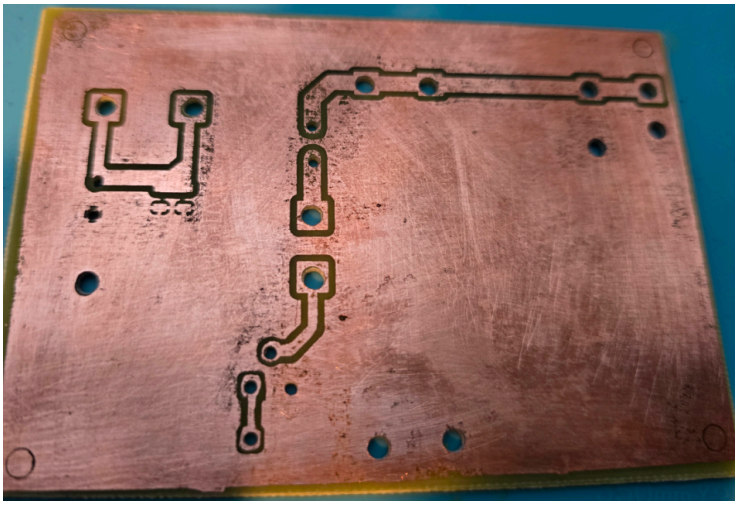
The easiest and cheapest way to produce the mask is to print it onto a piece of glossy paper with a laser printer and melt it with an iron. The toner will stick to a sanded and degreased copper better than the paper, and most of the mask will be transferred after 2-3 minutes of heating



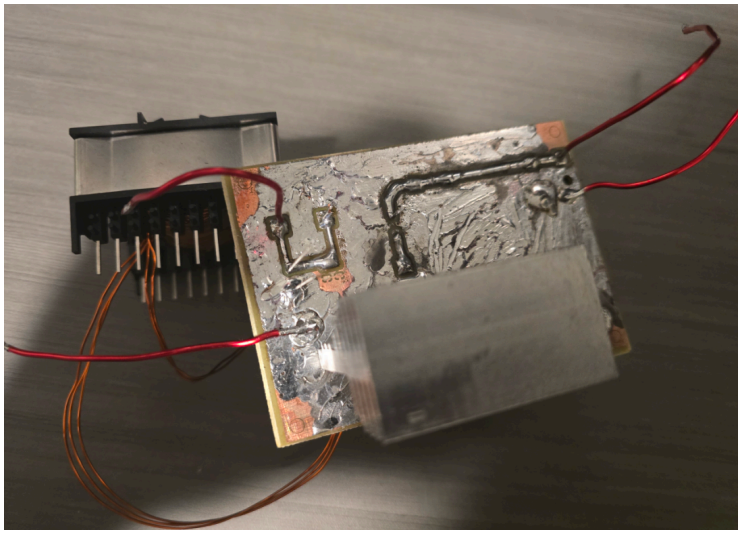
The toner will now allow the solution to dissolve the copper underneath. All the imperfections can be fixed with a permanent marker, it works similar to the toner.

After fully masking the copper plane, a solution of 3% hydrogen peroxide, citric acid and table salt in respective proportions of 100 ml : 30 g : 5 g per 100 cm^2 with copper thickness of 35 um is used to dissolve the exposed copper in 15 minutes. The solution is not toxic compared to Ferric Chloride which was commonly used by enthusiasts in the past.

The end result is washed with soap and degreased with isopropyl alcohol and is ready to be tinned.



The entire plate is treated with liquid flux and 63/37 solder wire to achieve a smooth and moderately thick conductive surface.



After that, all the components can be soldered onto the board on either side, preferably the side with no copper plane. If a double-sided plate is used, the vias can be made by using regular household or automotive rivets. More than 2 layer PCB production is impossible, and final green mask and print labels are not feasible for most people, and require precision technology, which makes the additional effort not worth it. An advantage of such DIY PCB with no mask, though, is a possibility of making the traces artificially thick, and therefore rated for 10-20+ amps of current by simply adding more solder.

References

- Information on 3-Phase Power Systems
    - https://www.accuenergy.com/support/reference-directory/what-is-a-three-phase-power-system/
- Role of PWM in Electronics
    - https://www.digikey.com/en/articles/the-role-of-pulse-width-modulation-in-electronics
- Quadrature Decoding Logic
    - https://www.ti.com/content/dam/videos/external-videos/en-us/9/3816841626001/6308652805112.mp4/subassets/interfacing_with_quadrature_encoders_-_updated.pdf
- Homemade PCB production on small scale
    - https://www.instructables.com/Home-PCB-Production-Easy-and-Fast-Method/
- TI C2000 Information
    - https://www.ti.com/tool/LAUNCHXL-F28379D
- H-Bridges Information
    - https://www.modularcircuits.com/blog/articles/h-bridge-secrets/h-bridges-the-basics/
- Increasing the gain of a boost converter using a coupled inductor
    - https://www.ti.com/lit/an/snva890/snva890.pdf?ts=1778124255258&ref_url=https%253A%252F%252Fduckduckgo.com%252F