Controllable, User-Friendly 3-Phase Inverter

By

Alex Chirita

Shyam Peden

Johnathan Vogt

Design Document for ECE 445, Senior Design, Spring 2026

TA: Frey Zhao

27 February 2026

Project No. 81

Contents

1. Introduction.......... 3
1.1 Problem.......... 3
1.2 Solution.......... 3
1.3 Visual Aid.......... 3
1.4 High-Level Requirements List.......... 4
2 Design.......... 5
2.1 Block Diagram.......... 5
2.2 Subsystem Overview and Requirements.......... 5
2.2.1 Boost Subsystem.......... 5
2.2.2 Bridge Subsystem.......... 8
2.2.3 User Interface Subsystem.......... 9
2.2.4 Input Control Subsystem.......... 11
2.2.5 Switching Control Subsystem.......... 12
2.2.6 Low-Voltage Systems.......... 14
2.3 Tolerance Analysis.......... 15
3. Cost and Schedule.......... 18
3.1 Cost Analysis.......... 18
3.2 Schedule.......... 19
4. Ethics, Safety, and Societal Impact.......... 20
4.1 Ethics.......... 20
4.2 Safety.......... 20
4.3 Societal Impact.......... 20
Appendix A. Full-Size Block Diagram.......... 21

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 will create an inverter system that is capable of creating three AC waveforms with controllable phase angles. Phase A will serve as the reference 0-degree phase, while the B and C phases will be controllable with respect to this reference phase. This will be achieved using analog control, likely via potentiometers. The PCB will function as a normal 3-phase switching inverter, with switching control handled by the microprocessor, which takes input from analog signals to control the output AC waveforms. There will be 5 main subsystems: input stage with a buck-boost topology, 3 MOSFET H-bridges, Encoder, CONFIRM button, and a small OLED Display as the user interface, and a TI-C2000 microcontroller, as it has enough PWM channels and high resolution timers, whose firmware will include the user input control, and the power control for the bridge.

1.3 Visual Aid

Figure 1 shows a very basic visual representation of our solution.

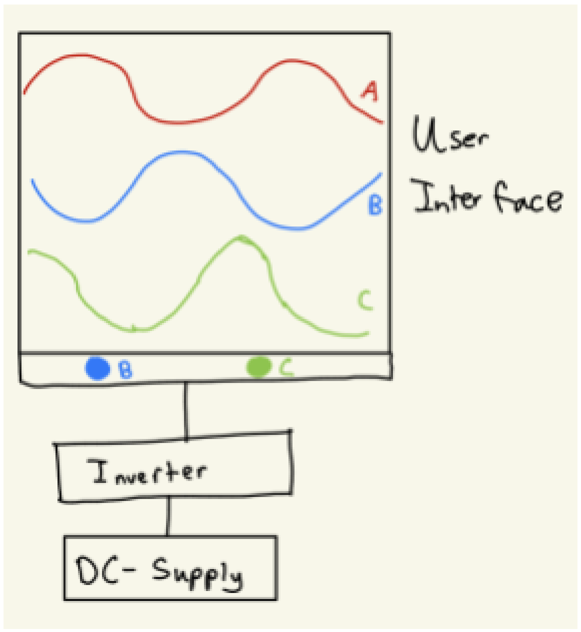


Figure 1: Illustration of our proposed solution with a DC supply, inverter, and user interface, all shown

1.4 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 will be tested on a Wye-Connected resistive load.

1. The Output RMS voltage will be 120V ±5%.
2. The inverter will output up to 0.28 A per phase, corresponding to 100 W of 3-phase output power.
3. The microprocessor will respond rapidly 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.

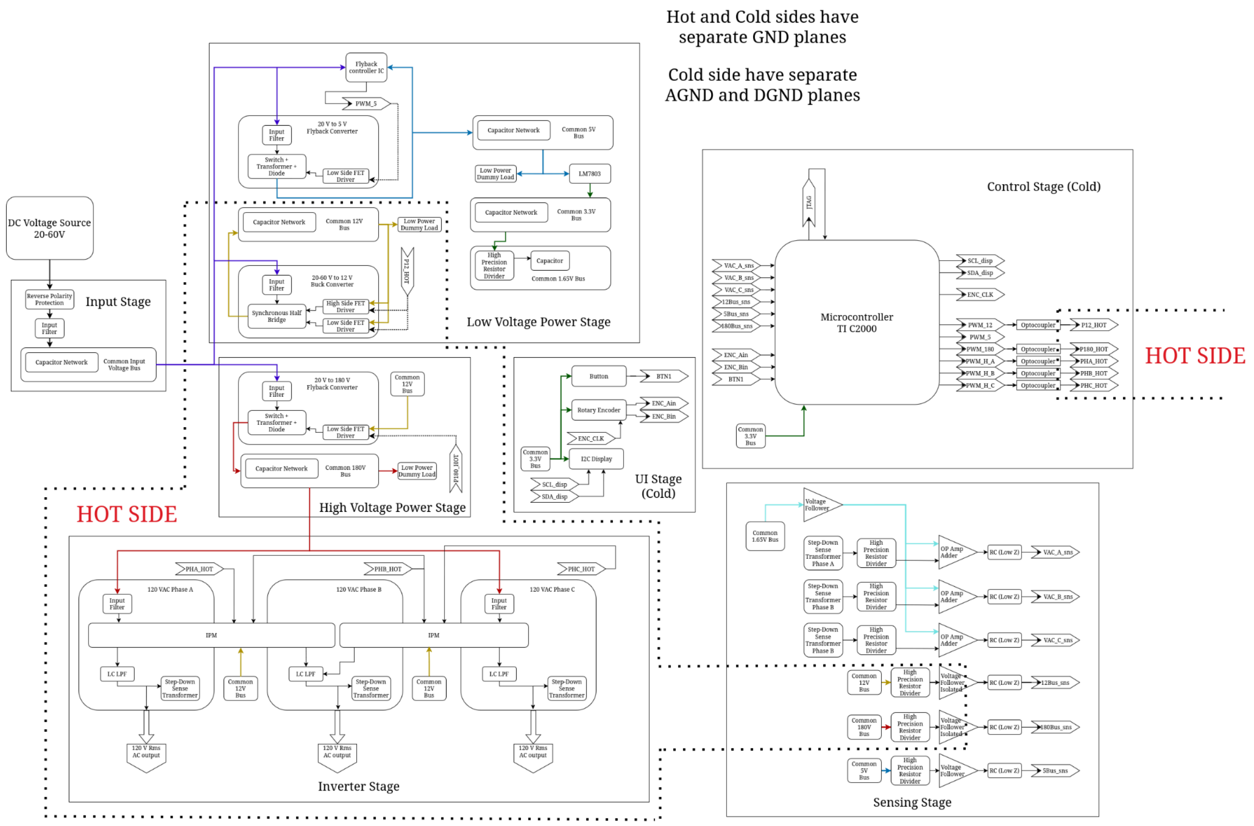


Figure 2: Block diagram of our proposed solution

2.2 Subsystem Overview and Requirements

2.2.1 Boost Subsystem

The main purpose of this project is the inverter stage, while the input will be any sort of DC power that mimics a solar panel. We will not focus on it being powered from a solar panel during the semester, and leave 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 will be done with a half-bridge buck-boost converter. A good quality of this system is the freedom to choose which variables to control. The circuit will be able to respond to quick changes in input voltage, but this semester, we will be using a constant DC power supply instead of a solar panel to reduce cost and complexity. Therefore, the boost converter will be run by a switching algorithm with a fixed input voltage. Later, the converter could be used as an MPPT if needed.

The boost converter will be 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 will 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).

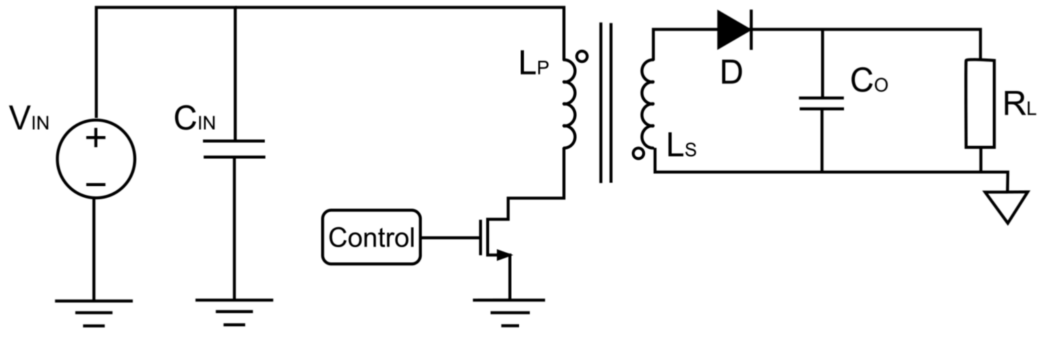


Figure 3: Flyback converter

With a transformer ratio Ns/Np of 1:9 and D = 0.5, we will be able to obtain 180V DC output at 20V input. At 100W output power, we will be pushing this topology to its limits, as generally flybacks are used up to 100W. With a careful magnetic design and high voltage MOSFETs, we have a high chance of success. However, if real hardware testing shows poor performance, losses in the core, too much heat dissipation on the switch, or poor load regulation, we have plan B and plan C.

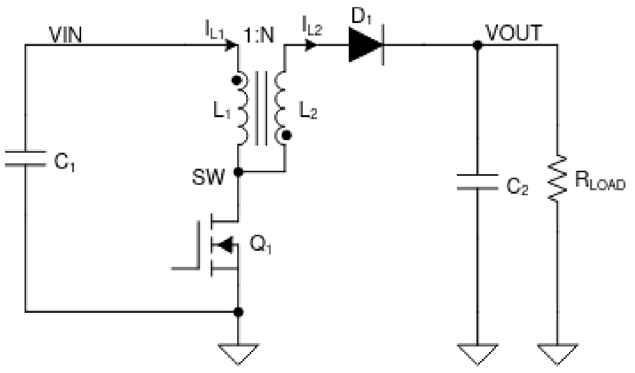


Figure 4: Coupled inductor boost converter

Plan B is 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.

Plan C is to chain 2 synchronous boost converters. One would get the voltage up to 80-100V and the other would do the final 180V. This is doable, and each stage will run at D = 0.33 when no power load is connected (low power dummy load must still be present). With higher loads, duty ratio will increase, and will still stay within acceptable range.

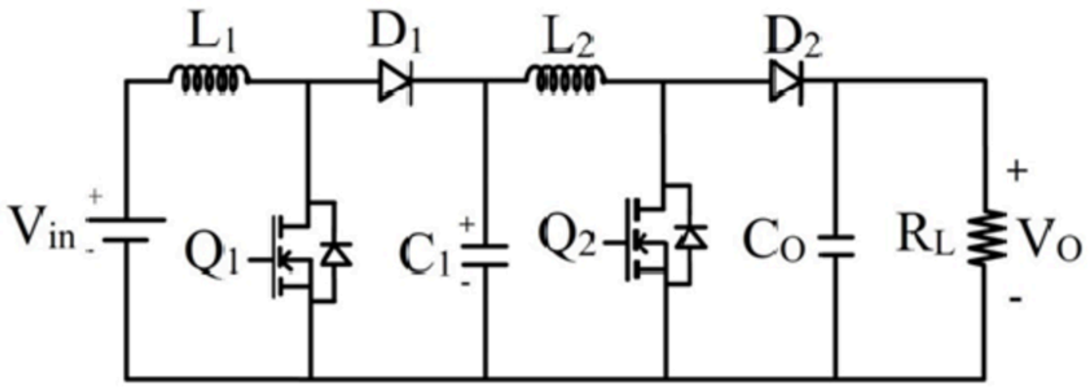


Figure 5: 2-Stage boost converter

We do have options in this case, and while this might be the hardest part of the whole design to actually get working and push efficiency, the latter solutions are pretty robust. We would get a lower duty ratio and proper boosting voltage by trading the size on the PCB and more losses in the switches. However, we do not have efficiency in high-level requirements, so this would be totally acceptable in our case if needed.

Controlling the voltage will be a challenging task. A good way to do it is through the current mode because effectively it only has one pole. Because it is a DC voltage, an isolated amplifier is pretty much required to deliver the sense voltage to the MCU. It is not very expensive, and provided we need the isolation anyway for the user control, so we do not accidentally inject line voltage into the knobs and buttons in case of a failure, this does not fundamentally change the design of the inverter.

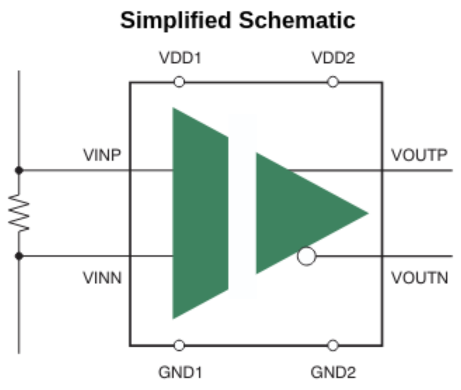


Figure 6: Isolated OP-Amp

Whichever topology will be used for this part, we will run it at a decently high frequency of 100-200 KHz because ripple currents will be huge. We have to choose beefy power inductors/transformers and high frequency to keep CCM operation. However, anything over 200KHz, and we are fighting with the switching losses, so the frequency will be kept in the midrange of industry power converters.

2.2.2 Bridge Subsystem

The bridge subsystem will contain three IGBT H-bridges, each corresponding to one of the phases. Each of the phases will 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 will allow us to generate a good-quality sine wave when averaged out and filtered with series inductors, which will be external to the board. The sine wave will be generated from -Vmax to +Vmax, and the sign will be decided by using a proper pair of MOSFETS from the 4 available. Each MOSFET will have a corresponding low-side or high-side gate driver, which will receive its PWM control signals from the microcontroller. Each phase will have an LC low-pass filter at the end to reduce switching harmonics. Our target output from the bridge system will be 120 Vrms with a tolerance of 5%. This is also one of our high-level requirements.

One of the practical ways to limit the size of the H-bridges is to use dedicated motor controllers (and/or IPMs). An AC motor with 3 phases is controlled by 3 half bridges in the vast majority of cases, so 2 IPMs will be enough to be wired as 3 H-bridges, while providing very high voltage ratings (600+ V). IGBTs have higher Q on the gate, which will require more power to drive the gates, compared to MOSFETs, but with a dedicated 12V bus from a buck converter, this will not be a problem at all.

Sensing the output voltage in this case will not be a problem because, especially after the low-pass filter, it is inherently an AC wave, so a small signal transformer can be used for fully isolated sensing, while preserving the wave shape. A 1.65V offset will be added to the sine wave through regular OP-Amp which will allow for the complete range of the -169.7 to +169.7 V to fit within 0-3.3 V ADC of the MCU. 1.65V can be easily achieved by a resistor divider connected to 3.3V main low-voltage bus. This is a signal bus, not a power bus, so we can use high value resistors to limit current. The voltage will come into the high-Z inputs of the OP-Amps.

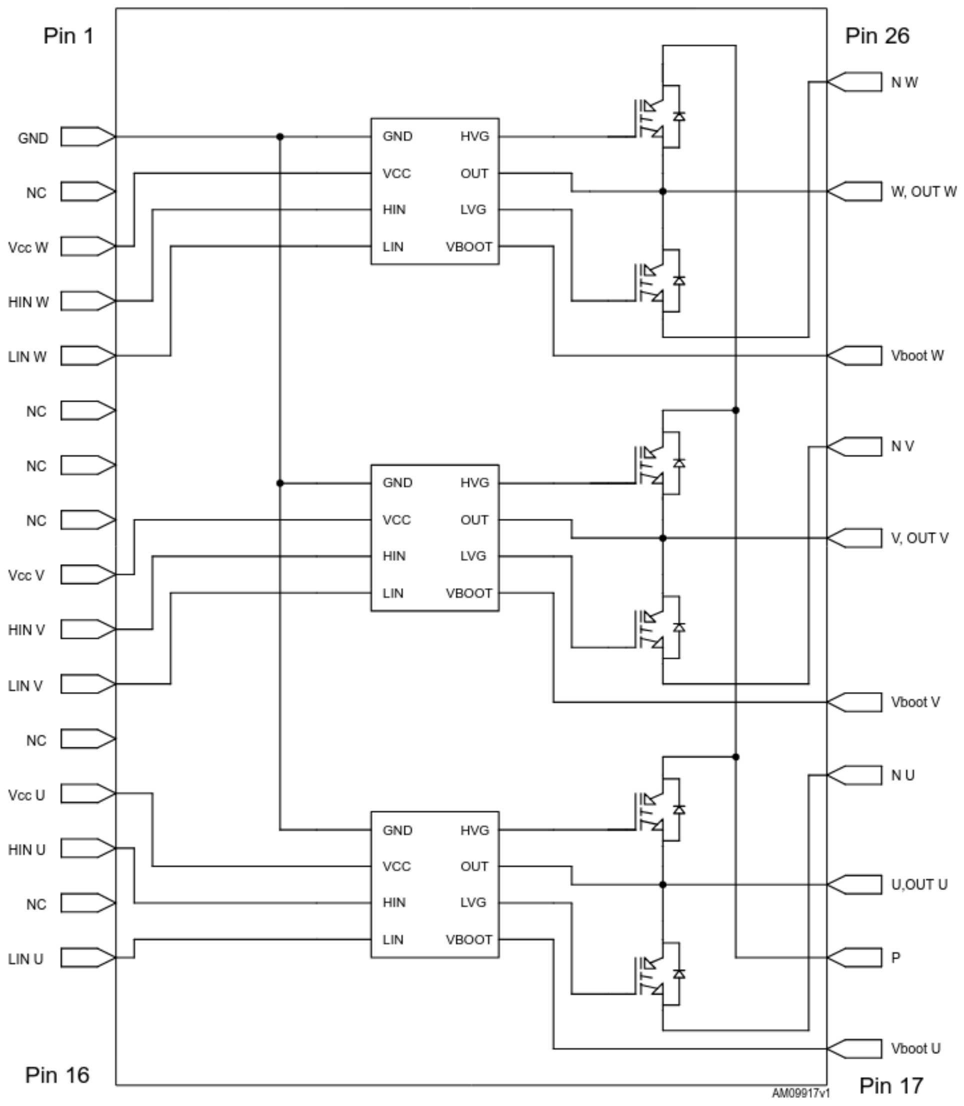


Figure 7: IGBT IPM for 3-phase motors

Outputs of 2 half bridges will be connected together, while bootstrap circuits are already included in the IPM, so apart from the power to the chip itself, we will only have to provide signal level PWM, which can come directly from the optocouplers. The input voltage into the transistors will be 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 sine wave by setting D at the boost stage, and we fully control SPWM timing of the sine generation where we can artificially decrease RMS voltage of the wave through timing.

2.2.3 User Interface Subsystem

The user interface will consist of a display, an encoder, and a confirm button. The user will 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 want the user interface to update as the input changes. More specifically, we want the interface to update within around 250 ms of a change in input.

The encoder will be connected to a clock, as shown in Figure 8. The reason for this is that the output of the encoder will be used as an input for the input control subsystem. Since the microcontroller needs to constantly update the input value and adjust accordingly, the clock will assist in rapidly changing the encoder output. It is also important to note that we will have 2 encoders since we want to be able to adjust both Phase B and Phase C

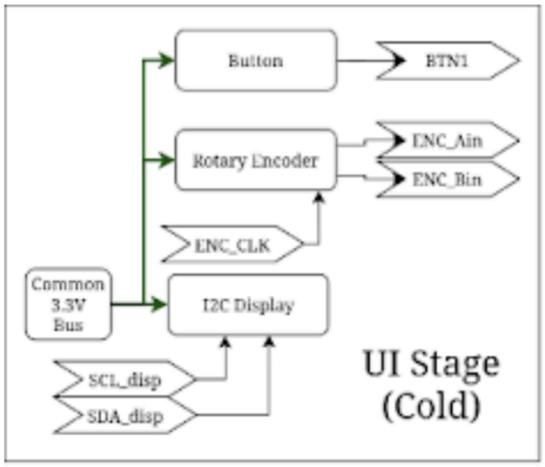


Figure 8: User Interface Subsystem Block Diagram

The button's functionality is quite simple. When it is pressed, it will output a high signal to the microcontroller indicating that it should use in the encoder inputs to update phases B and C. Afterwards, it will output a low value, indicating the button is not being pressed. Figure 9 shows the desired output of our button. As we can see, when the button is pressed, it outputs a high value, and otherwise it is a low value. In our sketch, we put 2 button presses, each having different amounts of time of the button press. This is because in this example, we would have held the button down longer the first time than the second time. The output is labeled as BTN1, which is shown in Figure 8.

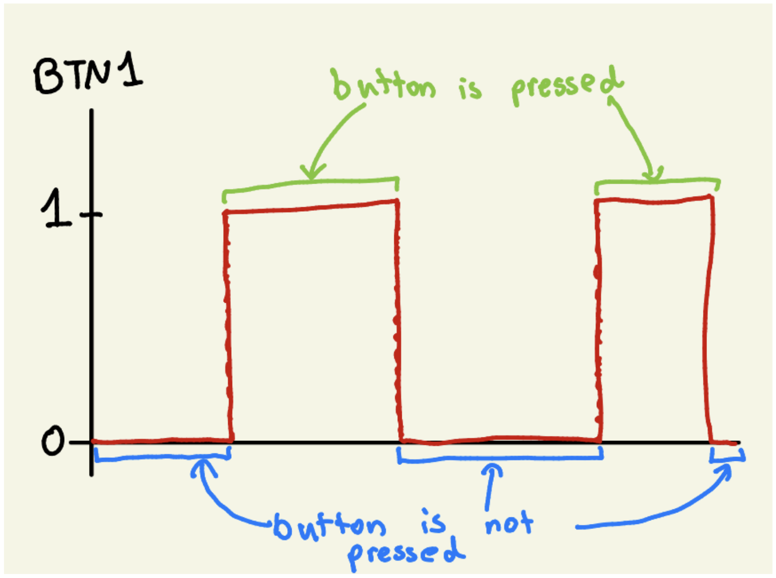
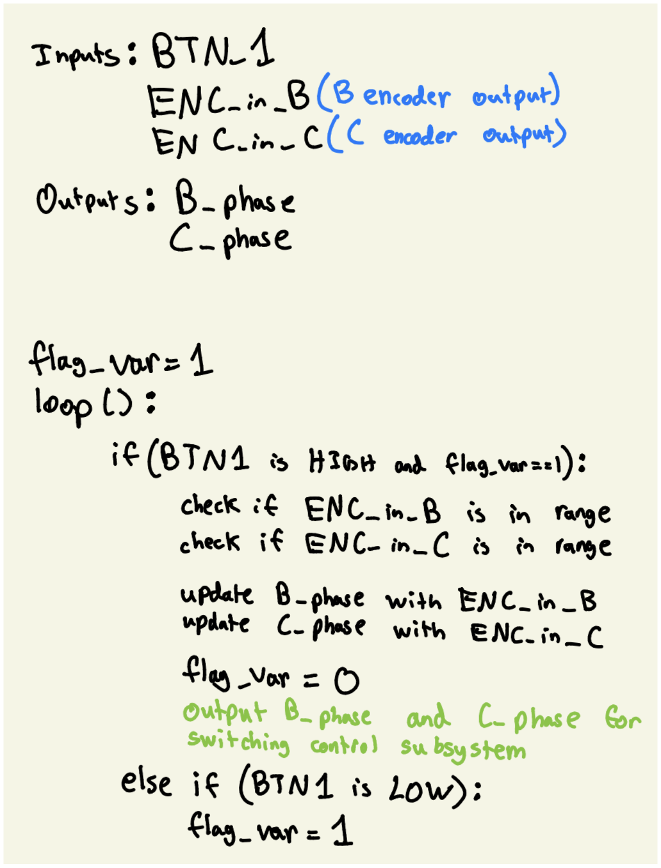


Figure 9: Sketch of Graph of button's output, BTN1 with 2 button presses

2.2.4 Input Control Subsystem

The microcontroller will run polling input from the button and encoders, run a loop that checks whether the phase is within bounds (-180 to +180 degrees with respect to phase A), and override the proper variables, which will be used by the switching subsystem as the target phase angle. We will need this system to update and override the variables rapidly for one of our high-level requirements.

Figure 10 shows the pseudocode that we would like to implement for our input control subsystem. The three inputs are the button output and the two encoder outputs. The two outputs will be the two variables representing the phases that we want the switching control subsystem to implement. We used B-phase and C-phase for these to make them easy to read. We also incorporated a flag variable. We initially set this variable equal to 1. The purpose of this variable is to ensure that once the button is pressed, the variables update for switching and then stop sending a signal until the next button press. For example, if we hold the button down, we can actively keep changing the phases. With the flag variable, this is not possible. Now, in the loop function, we look for the flag being equal to 1 and the button output to be high. From here, the function checks if the encoder outputs are within range before we update B-phase and C-phase. If both of these are true, we update the variables and send it to the switching control subsystem. We also set the flag variable to 0. Now, on the next clock cycle when the button is still likely pressed, the variables will not update. Finally, if the button is not pressed anymore, the flag variable will be set back to 1. Therefore, it is waiting for the next time the button gets pressed.



```
Inputs: BTN_1
        ENC_in_B (B encoder output)
        ENC_in_C (C encoder output)

Outputs: B_phase
         C_phase


flag_var = 1
loop ():
    if (BTN1 is HIGH and flag_var==1):
        check if ENC_in_B is in range
        check if ENC_in_C is in range

        update B_phase with ENC_in_B
        update C_phase with ENC_in_C

        flag_Var = 0
        output B_phase and C_phase for
        switching control subsystem
    else if (BTN1 is LOW):
        flag_var = 1
```

Figure 10: Input Control subsystem pseudocode

2.2.5 Switching Control Subsystem

We plan to use MCU for the control of the most onboard SMPS, except the 5V bus. The output voltage sensing in case of DC/DC converters will be transferred through isolated OP-Amps, and AC phases will be transferred through signal transformers. The PWM signal to the systems will be delivered through optocouplers. This provides full galvanic isolation between MCU and its subsystems, and the converters/phases.

All DC/DC converters will run in current mode control, because it makes a first order system with effectively one pole, instead of a second order system with 2 poles in case of voltage mode. 2 Control loops will be running on the MCU. The voltage loop will run at low frequency (500-2000 Hz) and will be checking the scaled down output voltage from the converters. It will run a PI controller and set up the allowed peak current. Then a high-frequency current loop will be running at f_{SW} and will be adding slope compensation on top of it, and checking against maximum allowed current set by the voltage loop. If it exceeds it, PWM turns low, otherwise it turns high. The current is read as the voltage drop on the shunt resistor (with calibration)

AC phases will run sPWM. The MCU generates an ideal sine wave internally with a set phase. The sensing will be done by measuring the full period of the wave as a frame with accumulating squares of the sensed voltage (effectively integral for the RMS calculation). Then after the frame is collected, the sum is divided by the number of samples collected within the window frame, and square root is taken. This gives us sensed RMS voltage. We then compare it to desired RMS to calculate the error. Then the PI controller is run on the error. It will set a new reference wave, and the cycle is repeated. We use this reference to generate normalized modulation by comparing it to the desired wave. This loop runs at a much higher frequency than the wave itself (20-50KHz). Each cycle the modulation is used to adjust the duty ratio on the fly. The high-frequency loop then sets PWM to high or low based on the duty ratio. This variable duty ratio encodes a low-frequency sine wave within the high frequency PWM carrier. To generate 2 additional phases, we simply need to shift 2 additional sine waves if the load is balanced. Unbalanced loads are not a part of high-level requirement, but can be achieved by running 3 independent controllers. We can extract the 60Hz wave using a low pass filter at the output of the H-bridges. It is not the job of the MCU.

The ideal sine wave will need to be calculated within the MCU. The easiest part of the switching control subsystem is the switching control for phase A. Based on our knowledge of PWM, we know the timings of how we want our output voltage at every point in time. Figure 11 shows our desired output as a function of time for a half-wave. It is in terms of T, which is just 1 divided by our switching frequency. It is important to note that thus is just for a half-wave, so it depicts from 0 to T/2. It does not show the desired output of the full period. In addition, it is also important to note that from T/2 to T, we will see a similar wave, it will just be negated. The proper way to relate it is that $v(t + T/2) = -v(t)$. Now that we have the first phase down, the slight tweak we must make to Figure 11 is just that we add some additional delay to the timings, thus mimicking the desired phase. For example, suppose we have a phase of -45 degrees, which is equivalent to T/8. We simply just have to adjust the times by that T/8 amount to display the new in phase wave.

$$\begin{cases} V_{pk}/2, & \frac{T}{16} \le t \le \frac{3T}{16} \text{ and } \frac{5T}{16} \le t \le \frac{7T}{16} \\ V_{pk}, & \frac{3T}{16} \le t \le \frac{5T}{16} \\ 0, & \text{every where else} \end{cases}$$

Figure 11: Desired output of PWM half-wave

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 will use a flyback converter (Fig. 3) to step down 20V to 5V. We will be running 2 separate grounds for the hot and cold side. We will be running isolation on the cold side in case we want to Earth reference the outputs, and also because the cold side will be low power (<10W), so flyback will be a perfect fit here. We will not be able to run the control on the MCU because it will generate the power for the MCU, so we need an offline controller. It will still run in current mode, because most of the control ICs are designed for it. We will most likely be running it in DCM to reduce the complexity of clamping circuits and magnetic components. DCM will ensure current goes to 0 before each cycle ends and will not destroy other components from inducing high voltage on the transformer winding.

Output sensing will be done with a divider circuit around TL431. After we set the desired voltage on the divider, when it reaches it, it will allow the optocoupler to transmit current on the hot side. This optocoupler, though, will have to be transistor type instead of logic type. At the hot side we will have the infrastructure around the controller copied from the datasheet of the IC itself, with adjusted values of resistors and capacitors. The work is already done, and we need to properly adapt it to our use case.

The 5V bus will be then connected to a 3.3V linear regulator. They require headroom in terms of input voltage, so we went with 5V. The output noise of the flyback will be slightly reduced by the linear regulator, because it effectively burns the excess voltage multiplied by the current flowing through it as heat, which will not be a problem in our case. The 3.3V bus will feed the MCU and use input devices. A simple resistor divider will generate 1.65V DC to add the sensed phase sine waves on top of it. This will not be a problem because it will be fed into OP-Amps, and they have high-Z inputs, so these resistors can be high value. Precision 1.65V will also not be particularly difficult because +-0.5% resistors are decently cheap.

12V for the gate drivers will be generated using its own buck converter, though running control on the MCU, because it is already powered from offline 5V flyback. We will need this buck because we will have more than 15 gates to be driven, 12 of which are IGBT. We will be sending pretty high currents consistently, so the complete 12V modules, even switching, will not be enough. However, one such module is enough to power the gates of MOSFETS of the 5V flyback, and this 12V buck. It has only 3 gates, and transistors are FETs, so the gate currents will be in acceptable range for the module. The 12V bus will be on the hot side because all the gates are on the hot side. Again, sensing will be performed with isolated OP-Amp, and PWM will be delivered through logic optocoupler.

Sensing will be the hardest noise-wise, because it requires a low-ripple voltage supply to the OP-Amps, stable and precise 1.65V sine wave mid-point, and isolated OP-Amps will be crossing cold and hot grounds, as well as be placed on the analog ground. Though, the package on most of them is chosen so that the hot and cold side legs are far apart to provide multi-KV isolation rating. Circuit-wise it will be pretty straightforward, because these are simple voltage adders.

2.3 Tolerance Analysis

One aspect of our design that could pose an issue to the successful completion of the project is the cutoff frequency of our low-pass filter. We need to use the values of our inductor and capacitor such that our cutoff frequency is much smaller than the switching frequency. We simply have to design our low-pass filters with values that account for this. In Figure 12, we can see mathematical equations for this design. We can use a switching frequency of 20-50 kHz. If we choose 20KHz, we can see, even considering the tolerances of the capacitor and inductor, we will still get a value that is much smaller than 20 kHz.

$$f_c = \frac{1}{2\pi\sqrt{LC}}$$

Let's use $C = 2.2\,\mu F$

and

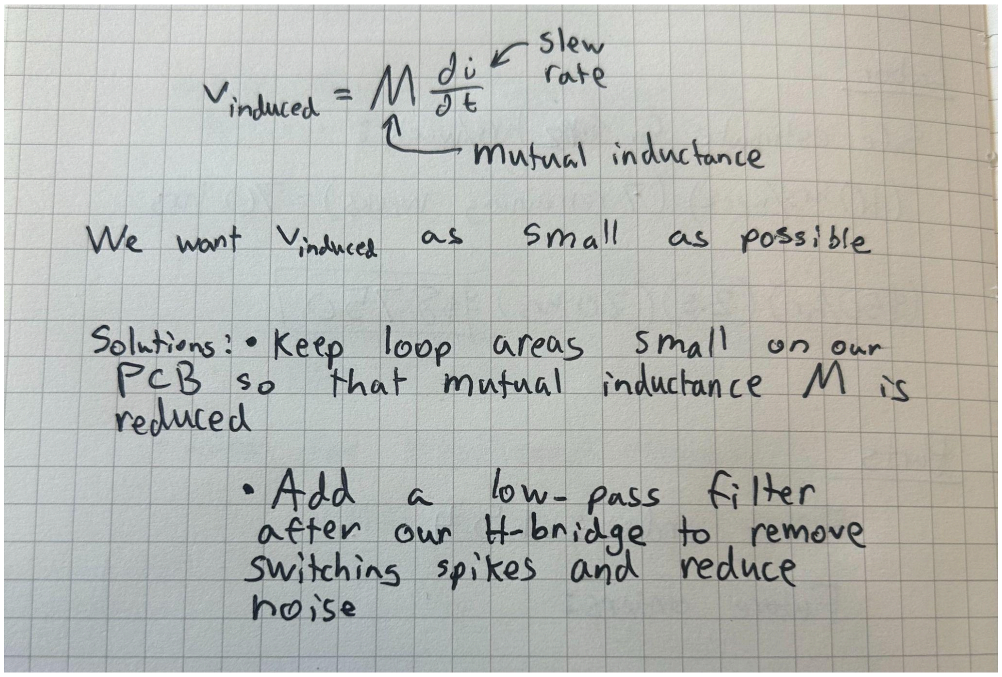
$L = 2\ mH$

$$f_c = \frac{1}{2\pi\sqrt{(2\cdot 10^{-3})(2.2\cdot 10^{-6})}}$$

$$= 2.4\text{ kHz} << \underset{f_s}{20\text{ kHz}}$$

Figure 12: Mathematical Analysis of calculating the cutoff frequency of low-pass filters

One other aspect of our design that could pose an issue for our project is the possibility of magnetic interference. Due to the fast switching currents in the MOSFETs, this could result in the generation of magnetic fields that could lead to reduced performance by our system. Figure 13 shows a description of the potential magnetic interference problem. As we can see, we need our induced voltage from the switching to be as small as possible. To do this, there are a few actions that we can take. First, we can keep our loop areas small for our H-bridge. By doing this, we reduce our mutual inductance, therefore reducing the induced voltage. Another possibility to combat this is that we can add a low-pass filter after the H-bridge. By doing this, we remove switching spikes and reduce noise. As a result, we are able to minimize the induced voltage, hence dealing with the potential magnetic interference problem. Our block diagram shows the implementation of the low-pass filters after each H-bridge.



$$V_{induced} = \underbrace{M}_{\text{mutual inductance}} \overset{\text{slew rate}}{\frac{di}{\partial t}}$$

We want $V_{induced}$ as small as possible

Solutions:
- Keep loop areas small on our PCB so that mutual inductance M is reduced
- Add a low-pass filter after our H-bridge to remove switching spikes and reduce noise

Figure 13: Magnetic Interference Potential problem and how to prevent it

One final potential issue to our design was with the size that we will need for the full design. After having a meeting with Johnathan Ashbrook, one of the UIUC alumni mentors, he brought up the fact that we might have to simplify some of the parts to reduce the size of the PCB needed. Based on our block diagram, he emphasized that a 100mm by 100mm PCB spec would not fit our entire circuit. As a result, he gave us some suggestions as to parts that we could buy in order to simplify the circuit and reduce space needed on the PCB. We also were told we are allowed to use multiple PCBs, so we used this information to plan for two PCBs that will be able to hold our entire circuit schematic. We can also use 2 100x100mm PCBs connected together with 10x2 signal connectors, and thick insulated wires for power lines as an alternative. This would give us freedom in building wide or building tall, depending on the required airflow.

3. Cost and Schedule

3.1 Cost Analysis

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 will work on the project is 10 hours per week. Given that there are 7 more weeks of school left, this gives us a total estimated labor cost of $8750. The next cost we have to take into consideration is the cost for the parts. As of right now, our first order of parts was $54, but the total allowed parts cost is $150. The remaining estimated cost for the rest of the parts needed is another $70, giving us a total cost of $124 for parts. Below is a complete list of parts that we think we will be using for the project (later revisions might occur):

- IRS2186STRPBF
- GSFH06100
- SURS8360T3G-VF01
- TI C2000 F2800157SPN
- PMEG3050EP-QX
- LM1085IT-3.3/NOPB
- 1N4002
- ECS-200-18-30B-AGN-TR
- CC0603FRNPO9BN270
- UCC28C43DR
- STGIPN3H60A
- IXFP72N30X3
- 530002B02500G
- EL357C
- TL431BIDBZR
- R-78E12-0.5
- RLS-397-R
- 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)
- Generic toroidal magnetic cores (Excess at Power Electronics Lab)

Adding up all of the costs, we get a grand total of $8874 including potential labor costs.

3.2 Schedule

Below is our estimated timetable for completing our project.

- Week 6: Finish design document, submit a test PCB for round 1 (rough draft, mainly to check the spacing between chosen components and estimating the full required size). Construct an offline 5V flyback, start calculating the values for the components for infrastructure around the controller. Order the first batch of components. Start calculating inductor and transformer cores, look if any are already available at home, lab, RSO to save cost.
- Week 7. Construct a 12V buck. Finish schematic for buck, offline flyback and phases. Start the “golden” PCB, route power lines. Submit flyback, buck, and a boost solution (can be rough) onto the “non-golden” PCB for round 2. Go to the design review. Construct all working subsystems onto breadboards/protoboards. Finish test control loops on arduino with artificial sense signals. Can be independent firmwares. Start porting for TI C2000.
- Week 8. Finish all required breadboard/protoboard parts that are needed to receive credit. Keep porting control algorithms to TI C2000 in one unified firmware. Finish final decision on boost topology. Try to finish offline flyback. Route everything as “golden” PCB. Submit whatever is finished to round 3. Construct the analog sensing circuit. Start looking for cores for signal transformers.
- Week 9. Spring break. Whenever anybody has free time, one might choose to make some progress towards the project. No significant progress is expected from any team member.
- Week 10. Route the final revision of “golden” PCB for round 4. All subsystems that are finished have to be routed as is, at least to an ok working condition, but as a whole system. Start minimizing sizes of magnetics. Finish open loops for every system control on TI C2000.
- Week 11. Finish closed loops for every system control on TI C2000. Work on fixing bugs. Pick enclosure, work on fitting everything there. Check the cooling system, check radiator temperatures. If required, install a 12V fan to be powered from a 12V buck converter.
- Weeks 12-13. Finalize the project. Fix final revision PCB mistakes by drilling incorrect traces, soldering any unrouted traces, soldering THT components legs to each other if PCB space is not initially designed for them. Avoid new SMD components. If the MCU subsystem fails, carefully route everything to the TI dev board.
- Week 14. Mock Demo. Work on final presentation and final papers. Fix any leftover bugs as the project requires.
- Week 15. Final Demo. Finish final presentation. Keep working on final papers.

- Week 16. Lab checkout. Finish final papers. Add last changes to lab notebooks.

4. Ethics, Safety, and Societal Impact

4.1 Ethics

The primary ethical issue we could run into with this project is that we must avoid harm to the user, as mentioned in Article 1.2 in the ACM Code of Ethics. We must prioritize the safety of the user while trying to maintain functionality. If we failed to do this, we would run into an ethical breach. We can do this through electrical isolation. In our situation, electrical isolation would prevent any unwanted transfer of DC voltage from the input and the buck-boost system to the AC currents we are displaying. By doing this, we can avoid a potential ethical issue.

Another ethical issue we could face is upholding integrity during this entire design and implementation process. The first article of the IEEE Code of Ethics and Article 1.3 of the ACM Code of Ethics discuss being honest and trustworthy when reporting measurements. This is incredibly important in a professional environment because failure to report results with integrity could result in a future user getting harmed. This would result in another ethics breach. False results also prevent progress on the project. To avoid this, we will be honest when reporting our results through every stage of our project.

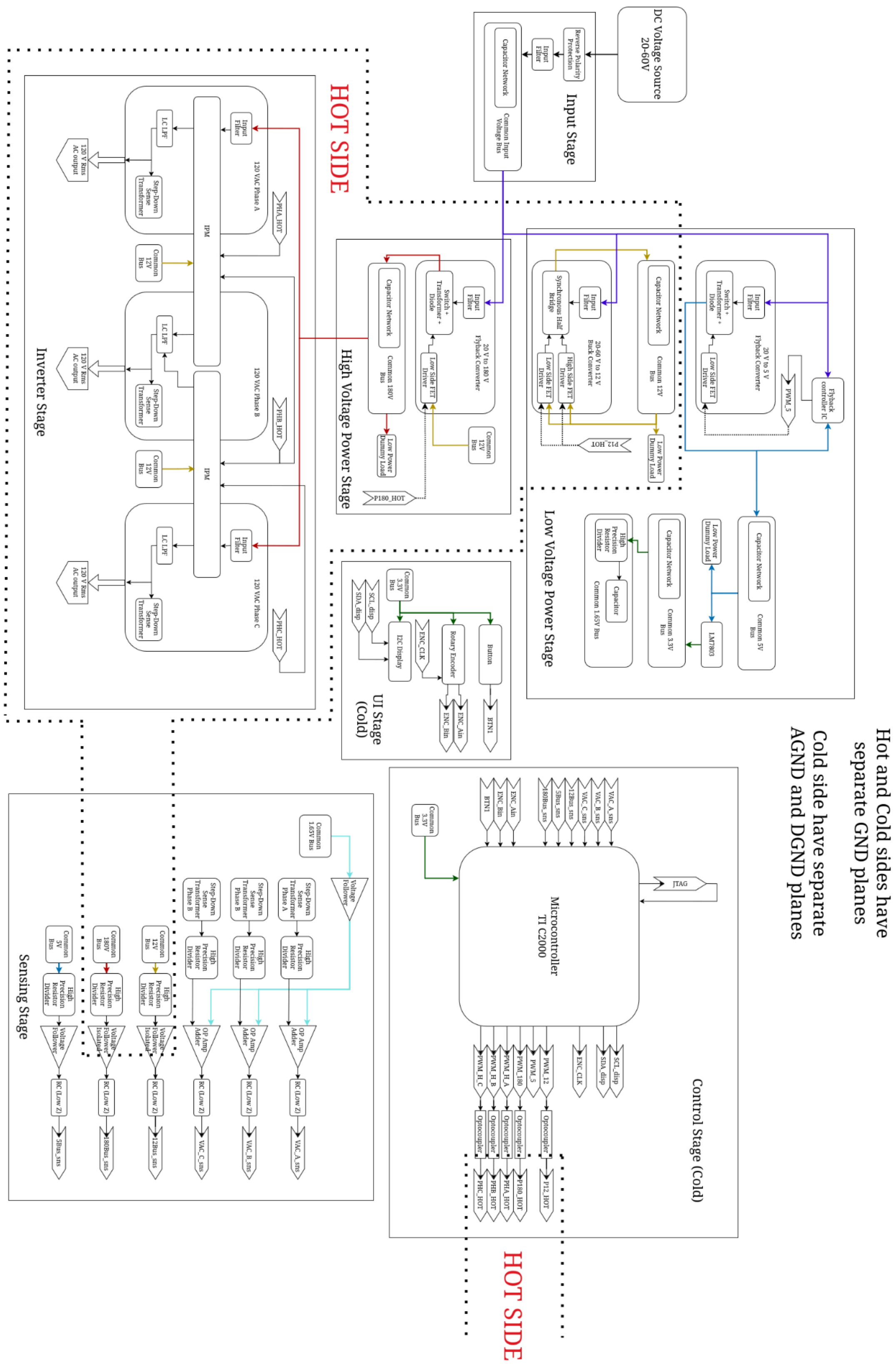
4.2 Safety

The biggest safety issue for this project is dealing with high voltages. Since we are dealing with high voltages, we must take heavy precautions when designing and working with our circuit. By taking proper precautions, we lower the risk of electric shock, electrocution, thermal burns, etc. The first step for us is to get certified to operate with high voltages. By completing this certification, we will be given instructions on proper high-voltage etiquette.

4.3 Societal Impact

Our solution could impact testing small machines by preventing fault conditions where they are unbalanced. Being able to account for every type of situation when testing a small machine would be extremely beneficial since testing of these three-phase systems can be more controlled. From an economic standpoint, our project can potentially help lower costs on early-stage research for smaller machines. This is obviously very beneficial since costs are lowered. Finally, from a global context, 3-phase systems are used in a variety of industries. An improvement in testing these systems has a positive impact globally.

Appendix A. Full-Size Block Diagram


DC Voltage Source 20-60V
Input Stage
Reverse Polarity Protection
Input Filter
Capacitor Network
Common Input Voltage Bus
HOT SIDE
Inverter Stage
120 VAC Phase A
120 VAC Phase B
120 VAC Phase C
IPM
Input Filter
LC LPF
Step-Down Sense Transformer
120 V Rms AC output
Common 12V Bus
PHA_HOT
PHB_HOT
PHC_HOT
High Voltage Power Stage
20 V to 180 V Flyback Converter
Switch + Transformer + Diode
Low Side FET Driver
Capacitor Network
Common 180V Bus
Low Power Dummy Load
P180_HOT
Low Voltage Power Stage
20-60 V to 12 V Buck Converter
Synchronous Half Bridge
High Side FET Driver
Low Side FET Driver
Common 12V Bus
P12_HOT
20 V to 5 V Flyback Converter
Flyback controller IC
PWM_5
Capacitor Network
Common 5V Bus
LM7803
Common 3.3V Bus
High Precision Resistor Divider
Capacitor
Common 1.65V Bus
UI Stage (Cold)
Common 3.3V Bus
Button
Rotary Encoder
I2C Display
ENC_CLK
SCL_disp
SDA_disp
BTN1
ENC_Ain
ENC_Bin
Hot and Cold sides have separate GND planes
Cold side have separate AGND and DGND planes
Control Stage (Cold)
Microcontroller TI C2000
JTAG
VAC_A_sns
VAC_B_sns
VAC_C_sns
12Bus_sns
5Bus_sns
180Bus_sns
PWM_12
PWM_5
PWM_180
PWM_H_A
PWM_H_B
PWM_H_C
Optocoupler
P12_HOT
P180_HOT
PHA_HOT
PHB_HOT
PHC_HOT
Sensing Stage
Voltage Follower
Step-Down Sense Transformer Phase A
Step-Down Sense Transformer Phase B
Step-Down Sense Transformer Phase C
High Precision Resistor Divider
OP Amp Adder
RC (Low Z)
Common 12V Bus
Common 180V Bus
Common 5V Bus
Voltage Follower Isolated