Integrated Brushless Motor Exploration Platform

By

Alex Roberts

Jason Vasko

Final Report for ECE 445, Senior Design, Spring 2025

TA: Michael Gamota

Professor: Yang Zhao

07 May 2025

Project No. 17

Abstract

This paper presents a single-board integrated brushless DC (BLDC) motor driver platform. This platform enables the user to start and stop a motor, set the desired motor speed, and choose between trapezoidal and sinusoidal control algorithms through an application running on a computer that is connected to the board via USB. The platform also displays motor phase voltage and current and system health information in real time. While the platform has significant limitations in its ability to control speed in real time, it does successfully drive a motor and allow the user to control it from the application with both trapezoidal and sinusoidal control algorithms.

Contents

1. Introduction

This paper explores the motivation, design, results, and issues related to an integrated brushless motor exploration platform. The PCB developed serves to be an educational tool in the the field of brushless direct-current motors by reducing the extreme technical knowledge requirements to meet the foundational result of getting a motor to spin, while providing a launchpad for exploration into motor control theory.

The majority of the paper will focus on the hardware and software design process, how circuits were derived from system requirements, and how the system is controlled. Following design will be system verification procedures, cost analysis, and finally the main conclusions through the design process. These conclusions will include the results currently achieved by the project, the shortcomings of the project in the form of challenges faced and uncertainties remaining in the design, ethical considerations, and finally future work to be done on the project.

1.1 Problem

As technology continues to develop, the electrification of mechanical loads continues to increase, such as the use of electric motors in robots and vehicles, where hydraulics and engines once were used. With the increased prevalence of electric motors in the coming years, there will be a growing field of study in motor controls. Currently, exploring topics in motor control requires at least a moderate knowledge of electronic hardware systems on top of the control theory being tested. Even when using commercial off-the-shelf motor drivers, system circuitry such as microcontrollers, power regulators, and power supplies still need to be properly chosen and connected together, which can be daunting.

There are individuals in math and controls heavy backgrounds, such as aerospace engineers, who are likely lacking much of the electrical engineering background needed to get a motor spinning, but they have the advanced knowledge of control systems to implement and test different algorithms for efficient motor control. There does not exist a simple solution for an all-in-one motor control platform designed for an educational use, as almost all commercial subsystems are optimized for application in products. This application focus removes all but the necessary circuitry for any subsystem to allow system designers to fit these modules in a wider array of products. This versatility however, places a problematic burden on a novice user to understand exactly how to connect every part of each subsystem, preventing people from exploring motor controls until they understand much of the electrical background behind them.

1.2 Solution

To address this problem, we created a single PCB which combines as much circuitry as possible for the operation and advanced control of common electric motors. Specifically, we have focused on brushless DC motors in our solution, as they are incredibly common and are a prime target for control theory students with subjects such as field oriented control. The hardware platform specifically combines the motor driver circuitry, microcontroller used for control, supplemental programming circuitry, and sensors required for the operation of the motor. The board is designed to use any common benchtop power

supply with a wide input voltage range to optimize versatility. The choice of a benchtop power supply was intended to limit the number of physical connections required by the user, with only a USB for communication between the controller and a computer, a benchtop power supply to power the board, and the motor phases themselves. As USBs are quite commonplace, ideally the user of our project will only need to connect two unfamiliar components, being the benchtop power supply and motor phases. The motor controller then communicates with an application on the computer, allowing users to modify and switch between the control algorithms used to spin the motor, as well as monitor real-time motor performance and PCB system health.

With a highly streamlined hardware platform, we aim to get more people interested in the field of electronic motor control. The motor driver circuitry is also built out of discrete components where possible to encourage the natural development of hardware knowledge as the user explores motors. By breaking circuits out into individual components, we allow the user to see and understand each hardware block in the system, and eventually potentially even experiment with changing hardware components as they become more advanced, such as changing FET technology or gate driver components. Ultimately, our hardware platform is meant to lower the barriers of entry to studying brushless motors by allowing users to work backwards from a spinning motor and topics in motor control to fundamentals of hardware to allow them to begin designing motor systems independently.

2 Design

2.1 Introduction

The design of this project is broken into four high-level blocks which we will refer to as “subsystems.” The categorization of system elements into the smaller subsystems was driven by logical groupings of smaller electrical components based on the purpose they fulfill in the system as a whole. This chapter will explore the overall system design, first at a high level to capture the purpose of each subsystem and how they connect to one another, then diving into each subsystem to explore how each subsystem specifically fulfills its respective goal. Finally, this project contains a significant amount of software running on the microcontroller to control the hardware correctly. The design of this software will be covered in the last section of this chapter.

2.2 Design Overview

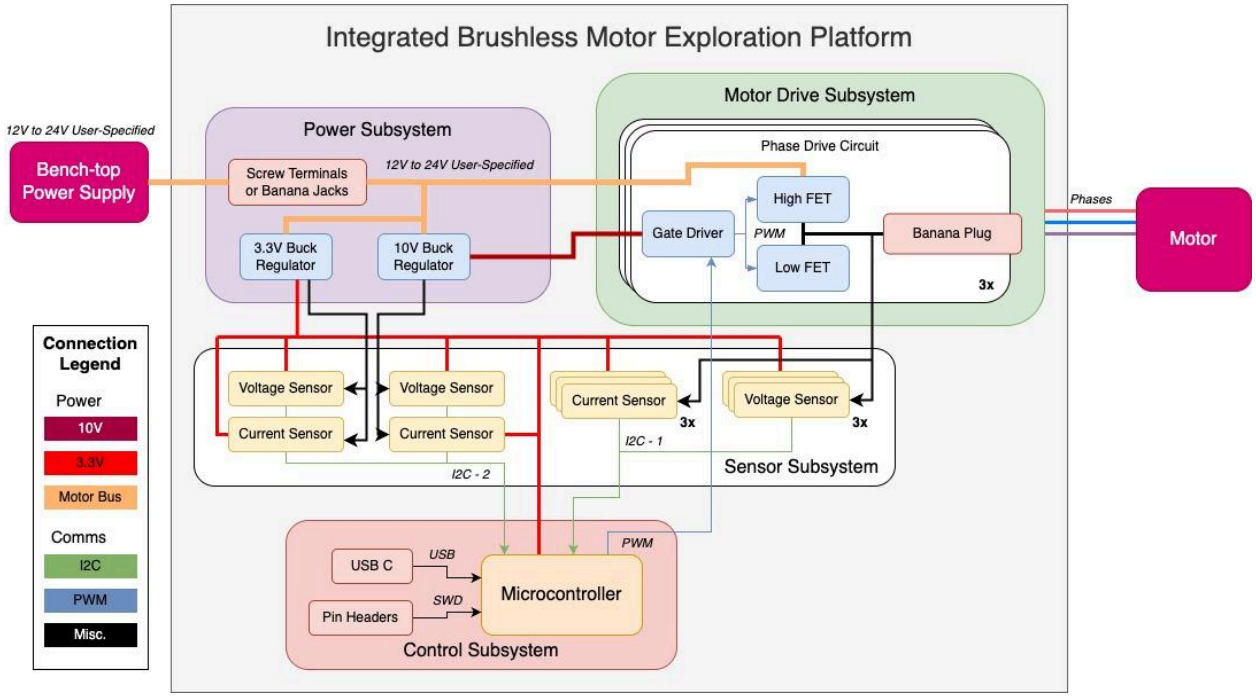

Figure 1: System Block Diagram

The high-level block diagram of the system is presented in Fig. 1. The design approach generally followed throughout the project was to start with high-level requirements, then create lower-level and more specific requirements, until a block was well defined enough to translate into a circuit and component decisions. The top-level requirements for the project as a whole are as follows:

1. *Motor operation and control* - The user should be able to start, stop, and control the speed of a brushless DC motor at any operating voltage between 12V and 24V using a GUI program, and achieve at least 1000 rotations per minute at top speed.

2. *Configurability* - The user should be able to switch between at least two distinct motor control algorithms (such as sinusoidal and trapezoidal control) and tweak algorithm parameters such as PID coefficients, PWM frequency, control frequency, and more.
3. *Motor performance and system health monitoring* - The user should be able to view motor phase voltages, phase currents, motor speed, shaft position, and voltage rail power consumption on the PCB in real time (>1Hz rate, <500ms latency) with historical graphs.

As the primary purpose of the platform is educational, we aim to emulate many of the features of motor drivers used in applications, but favor configurability and increasing the total number of possible operating conditions over system performance. These foundational philosophies are present in each of these requirements as we operate over a variable input voltage range and across multiple motor control algorithms. Naturally, allowing more operating points and modes requires development time to be split between each, and the maximum performance achieved by any individual operation profile will be reduced as a result.

The system is divided into four subsystems, each of which is labelled in Fig. 1. The core of the design is the motor drive subsystem, which contains the circuit directly responsible for connecting to the motor electrically, and powering the motor. This subsystem requires specific voltages and synchronized control signals which are provided by additional subsystems. The control subsystem is responsible for generating the control signals and running the motor control algorithm on the microcontroller. Both the motor drive and control subsystems are powered off specific voltages which are generated from the variable input voltage within the power subsystem. Finally, the motor drive and power subsystems have critical voltages and currents measured to report motor and system health statistics to the microcontroller, giving critical information that can be used for safety measures or more advanced motor control algorithms.

2.3 Subsystem Designs

2.3.1 Motor Drive Subsystem

The motor drive subsystem is the core of the hardware design and is the first step in the logical design process, as it is directly responsible for driving the motors, and all other subsystems will be designed to properly support the motor drive subsystem. Many of the design decisions for this subsystem are based in brushless DC motor theory, as the requirements of this circuit are based in how a brushless DC motor is actually driven. At motor theory is outside the scope of this design report, key takeaways will be presented alongside sources for further reading if required. Our high-level requirements also inform several subsystem requirements. The need to support the sinusoidal control algorithm enforces a 100kHz switching requirement to generate the proper waveforms, and the 12-24V operation is inherited from high-level requirements. The circuit topology chosen in early stages of design will also require cross-conduction prevention and dead-time, as discussed later. As a whole, this subsystem has the following requirements:

1. The PCB should function properly over an input voltage range of 12V to 24V.

2. The motor subsystem should be able to generate both trapezoidal and sinusoidal phase waveforms.
3. The motor drive subsystem should have hardware restrictions on shoot through, and should have at least 2ns of time between FET transitions.
4. Each half-bridge should be capable of 100kHz PWM frequency.

One element of BLDC motors which dictate the design of this circuit is the nature of these motors to have three wires, referred to as the “phases” of the motor. Each phase operates identically, has the same requirements, and has no individual differentiation, that is, the phases are interchangeable. Each phase will need to be connected to a high voltage, to ground, and electrically disconnected from the circuit at different points in operation. Ultimately phase independence and configuration requirements pushes the circuit design towards a half-bridge configuration. The requirements of each phase can be determined from many resources such as [1][2][3] and half-bridges appear as the most common topology in BLDC motor drives.

In general a half-bridge consists of two MOSFETs connected in series where the drain of one is tied to the source of the second, and this common connection is the output. The source of the second MOSFET is tied to ground, and the drain of the first is tied to a high voltage. In this configuration, if both MOSFETs are non-conducting, the output remains floating, while if either MOSFET is on at a given time, the output can be pulled to the positive supply voltage or ground. This circuit is shown in Fig. 2 and represents the main topology used in driving the motors in our application. Other methods of driving motors exist, such as multi-level converters, but these are typically far more complicated and less ubiquitous than simple 3-phase inverters based on half-bridges. Due to the educational nature of our project, scope of this class, and still impressive performance of this topology, the half-bridge base was chosen for this subsystem.

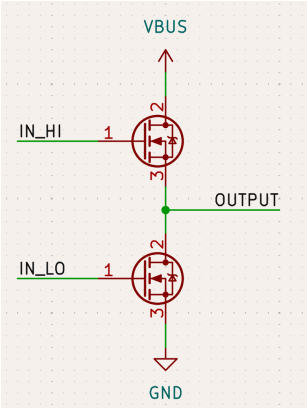

Figure 2: Simple Half-Bridge Circuit

All subsequent elements in the design as a whole serve to directly service these half-bridges, or support other components which will directly interface with these half-bridges. Figure 3 shows the subsystem schematic, with additional components, and the decisions behind each will be explored next.

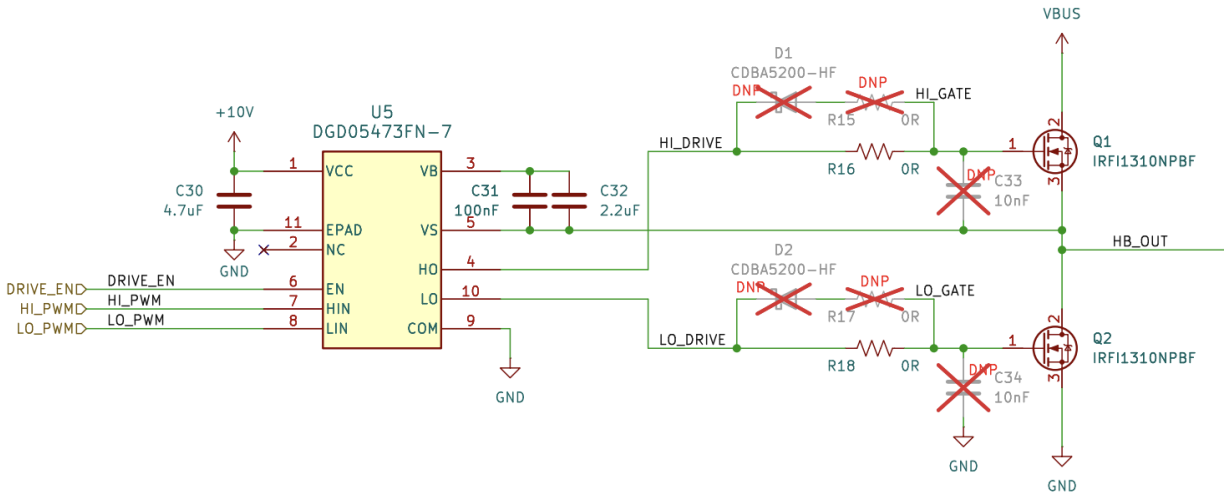

Figure 3: Motor Drive Subsystem Schematic

As seen in the simple half-bridge circuit in Fig. 2, there are two series-connected MOSFETs in Q1 and Q2 in the final subsystem circuit, as shown in Fig. 3. If both MOSFETs were ever on at the same time, the VBUS source would be shorted to ground and something in the circuit would break. To avoid this, dead-time is implemented in this system. Software dead-time is used, but the PCB was designed to include the option for hardware dead-time using D1, D2, R15-R18, C33, and C44. If implemented, the functionality examined on the high-side MOSFET is as follows: when the gate is pulled high, D1 blocks current flow through R15, so R16 and C33 form an RC network at the gate of Q1. When the gate is pulled low, D1 can conduct, so if R15 is much lower in resistance than R16, or even 0 Ω, the time constant to charge the gate is significantly lower than the constant to discharge the gate, causing a brief period of time where both MOSFETs are off when switching between states where only one MOSFET is on. Ultimately dead-time is implemented in software currently. U5 is the gate-driver IC which is required since a high-side MOSFET is present in the design. If Q1 is meant to pull the output node to VBUS, then the source voltage is VBUS, the highest voltage in our design. A high-side gate-driver uses a flying capacitor which charges when the low-side MOSFET is conducting, then changes the reference to the output node, allowing the gate of Q1 to be higher than the source (which is at VBUS) even though no voltage source larger than VBUS exists in the design.

Specific components were selected to optimize several parameters. The most important is design performance, as components must meet design requirements with margin. Following that in order is design complexity, sourcing difficulty, and finally price. If components simplify the system design significantly, or are easy for us to acquire quickly, we will choose that component even if it costs more than another option. The major components in this subsystem are the MOSFETs and the gate driver. The MOSFETs were chosen to be the Infineon IRF1310 since key requirements like maximum drain-to-source voltage, drain current, threshold-voltage, rise-time, and fall-time all met specifications, and the MOSFET was available in the Electronic Services Center, which is fast to supply and free for students. The gate driver was chosen to be the Diodes Incorporated DGD05473. Despite having a hard-to-solder package, being expensive, and in relatively low-supply, this gate-driver met all of the system requirements in a

single package, operates on a single voltage for logic and supply, and integrates cross-conduction prevention with specific enable signals, which simplified design complexity substantially.

The circuit shown in Fig. 3 is the drive for a single phase, so the entire subsystem relies upon three identical copies of this circuit. The circuit cannot drive the motor alone however, as the input signals HI_PWM and LO_PWM require precise timing in each phase, as well as between the three phases. Additionally, the gate-driver requires 10 V as an input, which we will need to derive from the input voltage range of 12-24 V. This leads into the remaining subsystems, which require far less background in explanation, as they are only supplemental to the motor drive subsystem.

2.3.2 Control Subsystem

The control subsystem aims to solve the control requirements of the motor drive subsystem. The control requirements can be broken down into three sections, controlling the timing of and between the high-side and low-side signals for each phase, controlling the timing between phase signals, and closing a feedback loop on the state of each phase to implement advanced control algorithms. The first two sections relate to advanced timing, while the third related to communication with current and voltage sensors. Since the high-level requirements for motor control algorithms also imply some need for digital logic or computation, a microcontroller was selected as the core of the control system, as advanced timer peripherals in microcontrollers can easily meet the timing requirements, and communication busses such as I2C with sensor ICs, or microcontroller ADCs can meet the sensing requirements.

The control subsystem fulfills the requirements previously listed, but also has requirements of its own to ensure smooth system operation. These requirements were chosen with a focus on user experience. Programming the microcontroller is important for changing control algorithms down the line, so multiple programming interfaces in USB and SWD were desired. Additionally, as software was being written to control the PCB with a laptop, constraints on the interface between the laptop and PCB were added. In total, the requirements for this subsystem are as follows:

1. The microcontroller should be programmable and debuggable over serial wire debug (SWD).
2. The microcontroller should be programmable over USB C.
3. The microcontroller should report data in real time, at an update rate of greater than 1Hz and a latency of 500ms to the GUI application on the computer.
4. Any failed connection between the computer and PCB should stop the motor from spinning within 2 seconds.

Aside from the microcontroller, which is central to the control subsystem's role as a whole, several other components were derived from the requirements of this subsystem. A programming header was required for SWD, and a USB C port was explicitly listed in requirement 2. An external oscillator is ultimately required to meet clocking standards for USB given our microcontroller choice and is added. Finally, several debug pin headers were included to allow for the reconfiguration of several microcontroller signals at a later date to ease software integration, such as boot modes and additional microcontroller pin outputs for debugging purposes. All of these additional components and the final schematic is presented in Figure 4.

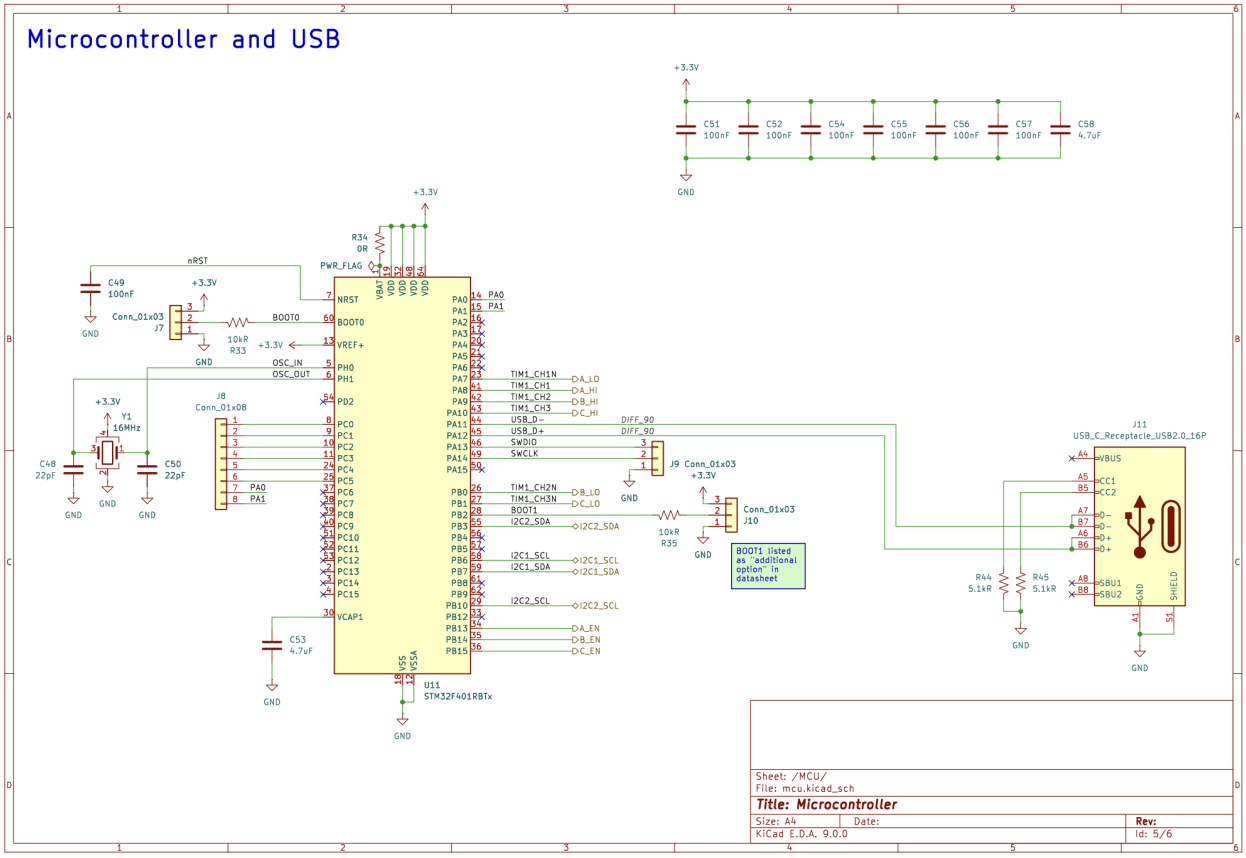

Figure 4: Control Subsystem Schematic

For specific component selections, the STM32F401RBT6 microcontroller was chosen first due to its availability in the Electronic Services Center. This microcontroller has an advanced timer peripheral, which allows for the generation of 6 PWM signals via complementary outputs on 3 channels. This lends itself extremely well to 3-phase inverters and is sufficient for our system. The USB C port was chosen for ease of soldering, as most USB C ports are rather similar in construction. The programming and GPIO headers were chosen to be 100 mil pin headers due to their ubiquitous nature and extreme ease in sourcing and compatibility with commercially available programmers and development tools. Finally, the oscillator chosen was 16 MHz in a standard package, there were many available options here, but none had a major impact on design whatsoever, there are likely hundreds of options that would have been acceptable. The microcontroller has two I2C busses which connect to sensor ICs in the sensor subsystem for phase and regulator measurements of voltage and current. The other connections to other systems are high-side MOSFET gate, low-side MOSFET gate, and gate-driver enable signals for each of the three phases.

This subsystem doesn't have many alternate design possibilities, as it's effectively required to include a microcontroller in a system of this nature. Interfacing with a computer and modifying control algorithms dynamically simply requires some form of digital computation. More advanced systems such as multiple microcontrollers with a division of responsibilities, a more powerful processor such as a CPU, or an FPGA

would have all been sufficient options as well, but all of these drastically increase system complexity in exchange for an improvement in computation power that is simply not required for this application. The sensor subsystem could have been incorporated using microcontroller ADCs, but several concerns led to separating the systems which will be explored in section 2.3.4.

2.3.3 Power Subsystem

The motor drive subsystem and control subsystem have covered most high-level and derived requirements so far, but each requires additional fixed voltages within the system. The microcontroller in the control subsystem needs 3.3 V for proper operation, and the motor drive subsystem requires 10 V for the logic and supply voltage of the gate drivers. As the overall system is designed to integrate as much functionality as possible, only one power connection to the PCB is desired, and this power connection has already been determined as a 12-24 V variable input, which allows the user to vary the motor bus voltage. Subsequently, the power subsystem was conceived to supply these necessary fixed voltages on the PCB from the wide input range, allowing the user to only connect one power supply and led the PCB handle the rest. As already discussed, the role of this system is to generate voltages, and only two are required. The requirements of this subsystem are then as follows:

1. All voltages required on the PCB except the motor bus voltage shall be generated on the PCB.
2. All voltages generated on the PCB should have an accuracy of +/-5% around their set-point.
3. The PCB should function properly over an input voltage range of 12V to 24V.

From these requirements, a power converter topology can be chosen. Since both voltages the subsystem must generate (3.3 V and 10 V) are below the minimum input voltage of 12 V, no upwards voltage conversion is necessary. The simple and extremely common buck regulator serves as an excellent power conversion topology, as it's highly efficient, only down conversion is required, and it's an extremely common circuit, allowing for online tools to be used in aiding the design process. Alternative topologies could have been used, such as a buck-boost converter, but these are typically less efficient since they also support upward voltage conversion, and are entirely indirect power converters, relying on the inductance in the circuit more heavily. Many other converter topologies exist, such as switched capacitor converters too, but ultimately they all sacrificed complexity, efficiency, or cost, and the simple buck regulator was a sweet spot for all of these variables, which is why it was chosen.

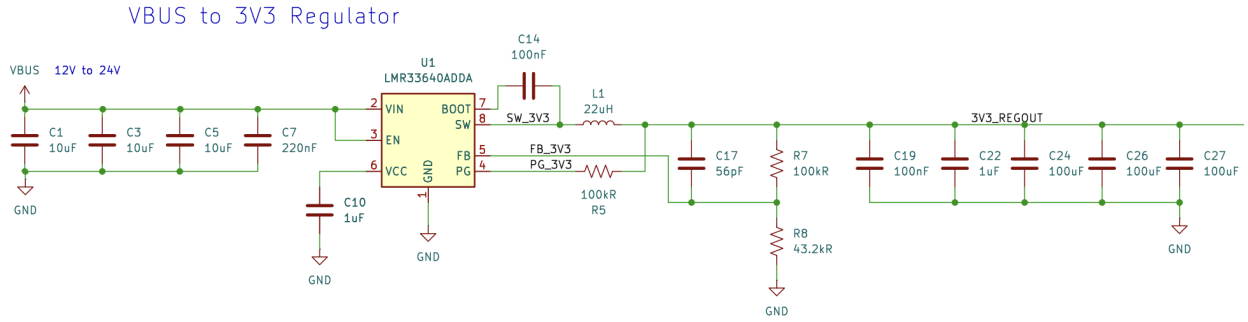

Figure 5: 3.3 V Output Buck Regulator Schematic

Figure 5 shows the schematic for the VBUS to 3.3 V buck regulator in the power subsystem. A nearly identical power converter is used for the 10 V conversion process, only several passive component values are different. U1 is the heart of the buck regulator, and contains the controller and integrated switches, only requiring filter elements and feedback resistors to be added outside the integrated circuit. This buck regulator was chosen as it had a very wide input range voltage, allowing for 12 to 24 V input, has a relatively high current capacity of 4 A continuous draw, and could be configured to complete the 3.3 V and 10 V conversions, reducing BoM complexity and price when buying in bulk. Additionally, the part was chosen since it is a Texas Instrument component, which allows the WEBENCH Power Designer online tool to be used [5]. The tool provides passive component values based on operating conditions and was used to choose component values for the 3.3 V and 10 V regulators in this design. Simulations provided by the WEBENCH tool were used to verify the component value choices manually.

The power subsystem as a whole consists of the two buck regulators and input connectors to allow VBUS to be supplied from a benchtop power supply. Additional bulk capacitance of 200 µF is included next to the connectors to smooth any fluctuations after the long power cables running between the power supply and PCB. Between the motor drive, control, and power subsystems, almost all design requirements are fulfilled, only leaving the sensing requirement to allow for closed-loop control of the phase voltages and currents, and system health monitoring of the power subsystem.

2.3.4 Sensor Subsystem

The sensor subsystem is the last subsystem and meets the remaining requirements on sensing the voltage, current, and power of each phase and voltage regulator on the PCB. Requirements on accuracy were created to bound the acceptable operation of the system, but more accurate sensors are always better. The subsystem requirements are:

1. All three phases should have voltage and current measured within 150mV and 100mA.
2. Any voltage rail generated on the PCB should have the voltage, current, and power reported. Measurements are for system health so an accuracy of 5% is acceptable.
3. Motor speed should be collected either by an encoder or through measurement of back EMF with +/-10% accuracy.

This subsystem has many different options in implementation, and selecting the components was one of the more uncertain aspects of the design process. The voltage and current sensing are chosen to be implemented using discrete sensor ICs rather than using ADC channels on the microcontroller. Additionally, the motor speed was chosen to be collected through back-EMF instead of an encoder. Both of these are major design decisions and took much consideration to make.

The choice to use sensor ICs ultimately came down to two things: complexity and reliability. Each current measurement taken on the PCB is truly a voltage measurement across a very low resistance inline resistor. To keep power losses due to this measurement small, the shunt resistance needs to be incredibly small, causing the voltage generated across the resistor to be small as well. If the microcontroller ADC has low bit resolution, the precision of the current sensing is likely poor unless an amplifier is used to amplify the measurement. Additionally, if the microcontroller is not physically very

close to the current sensing resistor, then the ADC channel traces must run across the PCB instead. These long parallel traces carrying a very small voltage are extremely susceptible to coupling interference from other switching signals in the PCB, including the 24 V phases switching on and off very quickly. This causes sensor readings to be less reliable. To solve these issues while using ADC measurements, passive filters before the ADC should be added on each channel, and voltage amplifiers should likely be added after each current measurement. Across 5 measurement locations, this amounts to 10 passive filters and 5 amplifiers, and 10 ADC channels being used on the microcontroller. The system complexity and reliability is massively increased if separate sensor ICs are used, which take sensitive analog measurements on-location to increase accuracy, then transmit the data digitally over I2C to the microcontroller to increase reliability, as I2C uses 3.3 V signaling, and digital protocols are very noise-resistant. As discrete sensors improved complexity, accuracy, and reliability, they were chosen over using the microcontroller ADCs.

The choice to use back-EMF speed sensing over a rotary encoder also came down to complexity at a system level. To use a rotary encoder, the system would need to mechanically link the rotary encoder and the motor somehow, as well as feed encoder signals to the PCB, or have the encoder mounted on the PCB, which would further increase the difficulty in designing the mechanical linkage. While a rotary encoder would provide more accurate speed measurements than back-EMF sensing, the increased accuracy of using sensor ICs was thought to make up for this loss somewhat, and the educational nature of the system deemed the performance of sensorless speed measurements acceptable. Complexity was truly the biggest factor however, as requiring the user to connect the motor at multiple more points drastically increased the burden on them to configure the platform properly, and this was a compromise we chose not to make.

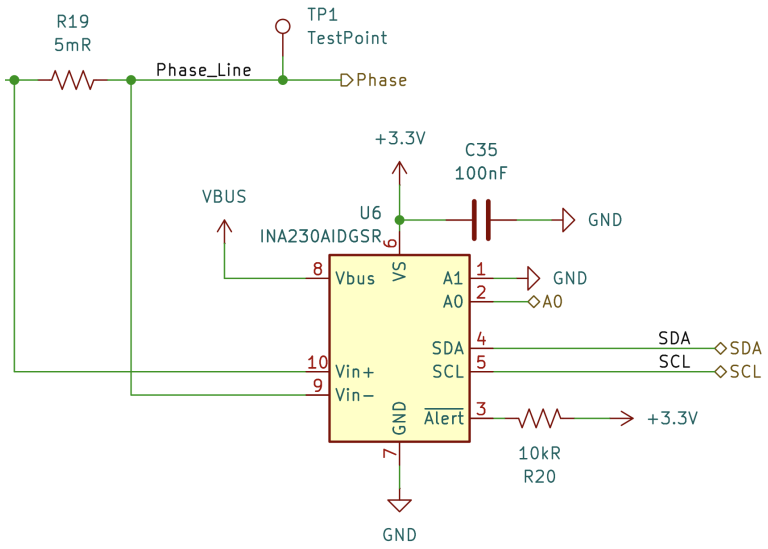

Figure 6: Single Current and Voltage Sensor Schematic

Figure 6 shows the implementation of one of the five sensor ICs that comprise the sensor subsystem as a whole. This particular instance is on a phase output. The output of the motor drive subsystem connects

to the left side of R19, and the phase of the motor connects on the right side of R19. This small inline resistor is only 5mR, so it acts almost like a wire, but produces a very small voltage across it as current flows, which is measured across the Vin+ and Vin- pins on the integrated circuit. This allows for current measurement to take place, while pin 8 measures the VBUS voltage at the phase. Similar implementations are on the 3.3 V and 10 V regulator outputs, allowing the current and voltage of each to be reported to the system. Each sensor IC is connected to one of two I2C busses, one dedicated to the motor phase sensors, and the other to the two regulator sensors. Each sensor also has different A1 and A0 connections to differentiate the I2C addresses of each sensor, so there are no conflicts. The microcontroller then reads sensor samples off of each of these ICs, completing the final system requirement and finishing the design of the PCB.

2.4 Software Design

2.4.1 Communications

We employed a basic keepalive messaging scheme to facilitate communication between the PCB and the GUI application, where one special message type, the status update request message, is used to maintain the connection between the PCB and the app. Note that the PCB's communication interface is essentially a server and will only send messages of its own in response to a message received from the GUI app. The keepalive messaging scheme functions as follows: once per configurable interval, the GUI will send a status update request to the PCB. Upon receiving this message, the PCB responds with updated values of power consumption for the board and voltage or current data for the motor phases. Thus, this message response functions as a keepalive message as well as a GUI status update request. All other message types are only sent from the GUI in response to some user interaction; these include a special message type to start and stop the motor and a message type to send parameters to the PCB.

2.4.2 Safety

To ensure safe operation, we used debounced system health flags to introduce hysteresis and ensure that one bad sensor reading will not unnecessarily stop the motor. We chose thresholds for several different flags; for example, if the 3.3V current surpassed a configurable threshold, this would be considered a fault and the 3.3V overcurrent flag would be raised. However, in order to protect our system from bad sensor readings, rather than simply setting the flag we use a counter for each flag. Each main loop iteration in which a flag is active causes its associated counter to increment, while each iteration in which it is inactive causes its associated counter to decrement. This allows us to set different thresholds for raising a flag and lowering it and makes the system more robust to noisy sensors than it would have been with normal flags.

2.4.3 Sensing

Due to the speed at which the PWM waveforms switch between high and low voltage, a high temporal resolution is necessary to view these waveforms with any accuracy. Thus, data must be collected quickly and sent to the GUI. The fastest rate at which data can be collected from the sensors on the PCB is once per main loop iteration. These data are then packaged along with the times between readings and sent to the GUI in a single dump each time the PCB receives a status update request. In contrast, only one set of system health update values are sent per status update, as the temporal resolution of these values

does not need to be as high to get useful information from them. Note that there is a tradeoff between message rate and message size; the lower the message rate, the more data there is to send since the last message, and the larger the message must be. We chose a 4Hz message rate, as this enabled the entire message to be sent in one USB buffer without incurring too much overhead by messaging too fast.

2.4.4 PWM Overview

PWM signals are generated on the PCB from the MCU using hardware timers. These timers can be configured in PWM mode to generate a PWM mode with a configurable duty cycle and period. Hardware timers can also be used to generate interrupts at consistent intervals, which proves useful when generating motor control signals.

2.4.5 Trapezoidal Control

One limitation of the hardware timers is that the channels of a single timer cannot be phase shifted relative to one another. Thus, to generate trapezoidal PWM, we used our primary timer in output compare forced output mode. In this mode, the timer's output level can be set from a bit in a register. We then use a secondary timer running at six times the desired frequency of the PWM waveform which generates an interrupt, and in this interrupt handler we change the output levels of the PWM waveforms in the other timer. Thus, we are commuting through six different stages to generate our waveforms. The only real alternative to this method is master-slave timer chaining with three advanced timers, but this would have required an updated hardware platform as our MCU only has two advanced timers. The advanced timers are important due to their ability to generate complementary outputs and software dead time between these complementary outputs.

2.4.6 Sinusoidal Control

Sinusoidal control functions by modulating the duty cycle of the PWM signal from 0 to 100 and back to 0 along a sinusoid from a precomputed lookup table, so upon applying a low-pass filter such as the inductance of the motor's coils, a sinusoidal signal is generated. This means that these PWM signals can be generated using a timer in PWM mode because the timer channels can be phase shifted through their position in the duty cycle lookup table. Similarly to trapezoidal control, a secondary timer is used to commute through the lookup table of duty cycle values and update the duty cycles of the primary timer's channels.

3. Verification

Verification was completed in a multi-phase process to avoid breaking downstream components if the entire system was tested all at once. The stages of verification are described in table 1.

Table 1 System Verification Stages

#	Stage	Description
1	Regulator Shorts	Before supplying any input power, after construction, verify that critical pins on the voltage regulators in the power subsystem are not shorted. Most importantly, 3.3 V, 10 V, and VBUS should all not be shorted to ground. Ensures construction is correct for regulator operation.
2	Isolated Regulator Voltages	Remove the shunt resistors between the regulator outputs and downstream components. Supply input power to the PCB and verify the regulators are operating correctly, the output voltages are correct, and hold steady under applied loads. Ensures regulators are ready to support the rest of the system.
3	System Shorts	Verify no other critical shorts occur on any ICs on the PCB. Namely, the microcontroller, gate-drivers, MOSFETs, and sensors should not have any pins shorted to 3.3 V, 10 V, or VBUS, as these could break the components. Pins should not be shorted to ground either for proper functionality. Ensures construction is correct for system operation.
4	System Idle & Programming	Re-solder the regulator shunts and supply power to the overall system. Check voltages again, then attempt to program the microcontroller. If problems occur, more careful analysis of construction or design is required, but if programming is successful, idle system operation and microcontroller programming is verified.
5	Theoretical Drive	Without a motor connected, run motor drive algorithms and confirm that waveforms are as expected on an oscilloscope. This ensures that the system is functioning correctly under no-load conditions, and the control algorithm is working properly.
6	Actual Drive	The full system integration. Connect a motor and try to drive the system again. If the motor spins, almost all lower-level requirements are verified implicitly.

These stages were all completed successfully as described in the table. During the assembly process, the regulators were first checked for shorts, and later verified in operation with all downstream components disconnected via removal of the shunt resistors used in the current sense circuits. The voltage rails consistently read within defined tolerances across load ranges, idling at 10.173 V and 3.319 V for the 10 V and 3.3 V regulators respectively, at 12 V input voltage.

After regulator output was confirmed, the system was checked for shorts, which required resoldering of the microcontroller. After no more shorts were detected, the shunts were added to the PCB and the entire system was powered for the first time. Verifying no components failed the “smoke test” of smoke coming from the PCB or extremely hot regions, the PCB was then programmed. At first the programming failed due to an assembly issue with the microcontroller being misoriented, but once that rework was completed, the microcontroller was programmed over SWD properly, and the system was idling as designed.

The next stage was to run trapezoidal and sinusoidal control algorithms and verify the phase output waveforms on an oscilloscope, which passed easily. The phase waveforms were exactly as expected, and the motor would theoretically spin when connected. The final test was to connect a motor, and it initially failed because the commutation was too fast, so the mismatch between the rotor speed and commutation speed caused the motor to oscillate instead of spinning. When driven at a slower speed however, the motor began to spin under either control algorithm, and high-level system functionality was completely verified.

A complete list of low-level requirements and their verifications is included in Appendix A. Several of these low-level requirements were not met. In total, the motor drive speed goal of 1000 rpm, the goal to program the microcontroller over USB, and the sensor speed measurement goal were the only requirements not met. The programming over USB was just more complex than initially anticipated, and there are software issues in configuring the USB driver running on the microcontroller. With time, we could meet this requirement without hardware changes. The other two failed requirements are directly related to one another. Motor speed sensing was not achieved at all in this implementation of the project, as the sensors were severely limited in communication speed with the microcontroller by the bandwidth of the I2C bus. The sensors were required to read and report far faster than anticipated previously, as the electrical frequency of commutation needs to be 7 times faster than the mechanical speed of the motor due to rotor magnet pole pairs. This prevented speed measurements from being feasible, and it's likely that ADC measurements should be used in a future version of the project for faster reading. Finally, as the speed measurement could not be recorded, motor speed control was open-loop, which is an unreliable, unstable, and poor method of high-level control. As a result, the commutation frequency could not be accurately controlled at high motor speeds above 700 rpm without causing a desync between the rotor and commutation frequencies, causing the motor to cog and begin oscillating. If motor speed was properly recorded and reported, the rest of the PCB would have no issues spinning a motor at 1000 rpm and meeting this requirement. The 1000 rpm requirement only failed as a result of a lower-level requirement also failing, which the 1000 rpm requirement relied upon. As outlined in this paragraph, these system failures are well understood, and possible solutions are already proposed.

4. Costs

The total cost of materials is $70.10 per device. We estimate the total cost of labor, including development time and assuming two prototype devices will be constructed, to be $25300. Thus, adding the cost of materials for two devices, we estimate a total cost of $25440.20 for this project.

4.1 Parts

Below is table 2, which is a bill of materials for the PCB and system. All components have their part number, item name, specifications, vendor (with link), price, quantity, and total cost listed. In total the entire cost of the bill of materials table is $65.80. Note that normally the INA230AIDGSR would be used for the power monitor IC, but due to stocking issues the more accurate INA226AIDGSR was used as a replacement, though the original component is sufficient to meet all design requirements normally.

Table 2 Bill of Materials

Item	Part Number	Specifications	Vendor	$ Per	QTY	Total $
10uF Cap	GRM21BC8YA106ME11L	CAP 0805 10uF 35V Ceramic	Digikey	$0.28	6	$1.68
1uF Cap	CL21B105KBFNNNE	CAP 0805 1uF 50V Ceramic	Digikey	$0.08	4	$0.32
100nF Cap	CC0805KRX7R9BB104	CAP 0805 100nF 50V Ceramic	Digikey	$0.08	19	$1.52
2.2uF Cap	C2012X7R1C225K125AB	CAP 0805 2.2uF 16V Ceramic	Digikey	$0.18	3	$0.54
4.7 uF Cap	GRM21BR61H475KE51L	CAP 0805 4.7uF 50V Ceramic	Digikey	$0.19	5	$0.95
220nF Cap	C0805C224K5RACTU	CAP 0805 220nF 50V Ceramic	Digikey	$0.10	2	$0.20
100uF Cap	GRM32ER61A107ME20L	CAP 1210 100uF 10V Ceramic	Digikey	$0.84	6	$5.04
56pF Cap	C0603C560J5GACTU	CAP 0603 56pF 50V Ceramic	Digikey	$0.10	1	$0.10
22pF Cap	C0603C220J5GACTU	CAP 0603 22pF 50V Ceramic	Digikey	$0.12	3	$0.36
100uF Cap	EEH-AZA1V101B	CAP TH 100uF 35V Alum Hybrid	Digikey	$1.42	2	$2.84
5.1k Res	RC0805FR-075K1L	RES 5.1K OHM 1% 1/8W 0805	Digikey	$0.10	6	$0.60
10k Res	RMCF0805FT10K0	RES 10K OHM 1% 1/8W 0805	Digikey	$0.10	7	$0.70
5m Res	CRF2512-FZ-R005ELF	RES 0.005 OHM 1% 2W	Digikey	$0.49	5	$2.45

		2512				
100k Res	RC0805FR-07100KL	RES 100K OHM 1% 1/8W 0805	Digikey	$0.10	4	$0.40
11k Res	CRG0805F11K	RES 11K OHM 1% 1/8W 0805	Digikey	$0.10	1	$0.10
43.2k Res	ERA-6AEB4322V	RES 43.2K OHM 0.1% 1/8W 0805	Digikey	$0.10	1	$0.10
0 Res	ERJ-6GEY0R00V	RES 0 OHM 1/8W 0805	Digikey	$0.10	13	$1.30
22uH Ind	SRR1260-220M	IND 22UH 4A 43mOHM SMD	Digikey	$0.96	2	$1.92
Banana Jack Plug	CT2220	CONN BANANA JACK THRD	Digikey	$0.95	5	$4.75
100mil Header	61300811121	PIN HEADER VERT 8POS 2.54MM	Digikey	$0.36	3	$1.08
Screw Terminals	1715721	TERM BLK 2P SIDE ENT 5.08MM	Digikey	$0.97	1	$0.97
USB C	USB4085-GF-A	CONN RCPT USB2.0 TYPE C 16+8POS	Digikey	$0.88	1	$0.88
MOSFETs	IRFI1310NPBF	MOSFET N-CH 100V 24A TO220AB FP	Digikey	$2.12	6	$12.72
16MHz Crystal	ECS-2333-160-BN-TR	XTAL OSC XO 16MHZ HCMOS SMD	Digikey	$0.84	1	$0.84
4A Adj Buck IC	LMR33640ADDAR	IC REG BUCK ADJ 4A 8SOPWR	Digikey	$1.92	2	$3.84
Half Bridge Gate Driver	DGD05473FN-7	IC GATE DRV HALF-BRDG DFN3030-10	Digikey	$1.31	3	$3.93
MCU	STM32F401RBT6	IC MCU 32BIT 128KB FLASH 64LQFP	Digikey	$3.97	1	$3.97
Current and Voltage Sensor	INA226AIDGSR	IC CURRENT MONITOR 0.02% 10VSSOP	Digikey	$2.34	5	$11.70

In addition to the cost of materials on the PCB, there is also the cost for the PCB itself, though this is quite cheap. Our PCB gerber files for the first revision were uploaded into the JLCPCB quote tool to estimate the pricing of our PCB. This quote is shown below in figure 7. Ignoring the promotional deal, it would cost $4 for QTY 5 of our PCB, with $17.50 in shipping costs. Factoring in shipping, it costs $21.50 for 5 PCBs, or $4.30 per PCB, which can be added to the previous materials cost estimate.

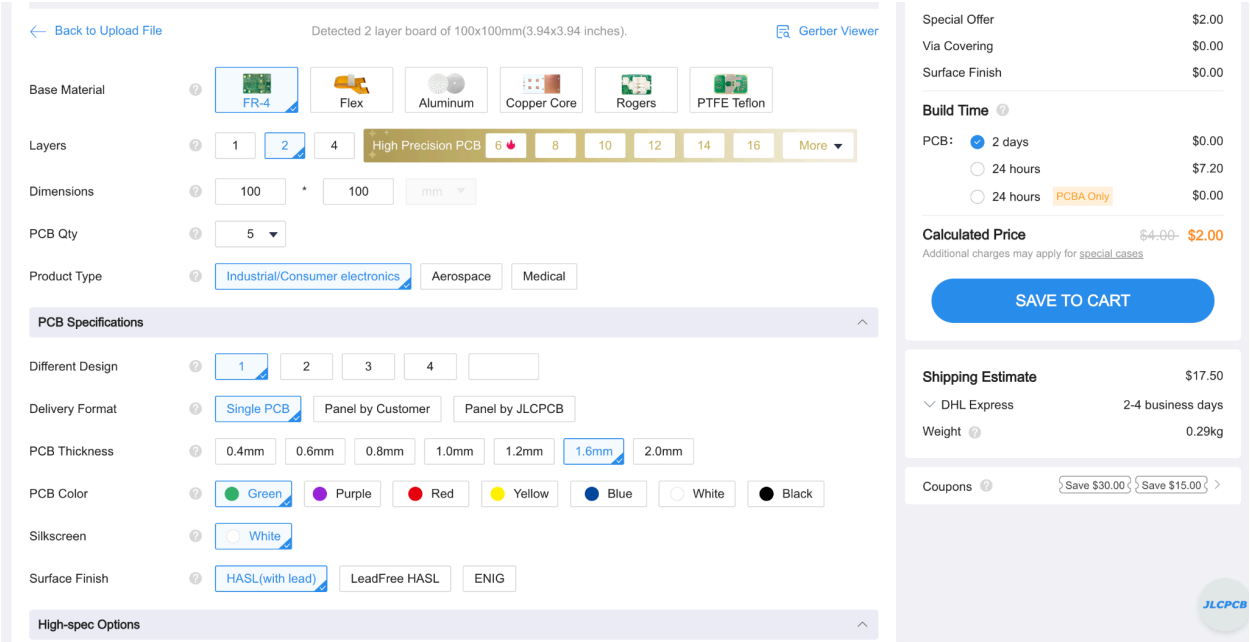

Figure 7 - JLCPCB Quote

In total then, with $65.80 in component costs and $4.30 in the PCB cost, the total cost for our hardware platform is $70.10 in materials.

4.2 Labor

First, we will assume a reasonable salary of $40/hr. The labor cost can be divided into development time and device construction time. For device construction, we conservatively estimate 4 hours of labor for device soldering, programming, and validation per device. For development, we can further divide the labor duration into hardware and software development. For hardware development, we estimate 35 hours of work on the initial design and PCB, 50 hours for evaluation and verification of the first prototype, and 70 hours for support and fixing bugs during integration. This yields a total hardware development time of 135 hours. For software development, we estimate 20 hours of planning and research and 70 hours of development time and fixing bugs. Therefore, we estimate a total development time of 245 hours. Adding a total device construction time of 8 hours for two device prototypes, we estimate the total labor time of this project to be approximately 253 hours, and a total cost of:

$$253[hr] \times 40[\$/hr] \times 2.5 = 25300[\$]$$

5. Conclusion

5.1 Accomplishments

Overall, our platform successfully enables the user to drive a BLDC motor from an application running on a computer connected to the board via USB. This functionality includes the ability to change the desired speed of the motor by entering a target frequency that the motor will attempt to spin up to, and to select between trapezoidal and sinusoidal control algorithms. The application also successfully displays system health information in real time, including any detected abnormal voltage or current values and the power consumption of each rail generated on the board, and the phase voltage and current of the motor phases.

5.2 Uncertainties and Challenges

At present, the design faces several major uncertainties posed by challenges faced in the final weeks of the course. In the development and evaluation of our most advanced feature, being speed control of a sinusoidal driven motor, we had several gate-driver failures in which the high-side driver output did not respond to a change in the input. Several pieces of debugging evidence point towards a failure inside the gate-driver occurring only after long periods of operation. The design of the gate-drive schematic is at least mostly correct, as the device functions properly for long periods of time, but not as long as it should, suggesting that some lifecycle affecting parameter, perhaps maximum allowed voltages, is being exceeded. More debugging and replication of this failure is needed to determine the root-cause, and is currently the largest uncertainty in the design. Not much else in the system is uncertain currently, as other shortcoming have been well defined, and have clear solutions. Fixing these problems should either present new problems that were not apparent before, continuing the engineering design process of iteration, or would result in a working PCB.

The primary challenge faced is the sensor reading speed for phase voltages and currents. The slow reading speed caused motor speed to be unacquirable, which severely limits the complexity of any control algorithm we implement, prevents the motor from surpassing 1000 rpm, and limits the information presented to the user, or even a metric from which to compare control algorithms, which is vitally important in allowing someone to explore differences between algorithms. Solutions for this problem will be presented in future work, section 5.4.

5.3 Ethical Considerations

In considering the ethics surrounding the development and existence of this project, two main points in the IEEE code of ethics stand out in their relation to this project. The central focus of the project is to be an educational platform for the driving and control of brushless DC motors, an already highly utilized and increasingly important technology. The IEEE code of ethics mentions in point 2 the ethical need to improve the understanding of people in the capabilities of conventional and emerging technologies [4]. It also highlights the necessity of treating all persons equally and with respect, regardless of background in point 7 [4]. We believe the existence of our project is closely linked with these two points, as it will provide a way for a wider array of people to understand brushless DC motors. Additionally, in aiming to make the platform as accessible as possible, we help to lower the barrier of education which can be

frequently apparent in hardware systems, where price of equipment can make learning less accessible. By making an open learning platform using common and cheap components, students from less wealthy backgrounds may be able to study a subject they were previously unable to gain hands-on experience with.

We also followed all aspects of the IEEE code of ethics during the development process of the project, including supporting one another as teammates as outlined in point 10 [4].

5.4 Future work

The most important future work is to solve the issues faced by the system currently. The most important of which is investigating the cause of the gate-driver failure seen twice in the system, and implementing any changes needed to solve this and make the phase outputs reliable.

Secondarily, fixing the sensor reading speed will allow our system to achieve all of its initial requirements, and reach the complexity of operation we were aiming for. To characterize what needs to be done, samples need to be taken on the tens or hundreds of kilohertz timescale, to allow for many samples and the discernment of important waveform features in the commutation waveforms, which are the kilohertz timescale. In order to achieve this speed, and the synchronization with commutation that is required for more advanced control algorithms, the system should switch to collecting phase voltage and current measurements with the microcontroller ADCs. This will increase system complexity, but it is necessary for more advanced operation.

Beyond these improvements, on the software side, implementing things such as field-oriented-control would be very interesting. This algorithm optimizes for maximal torque output, and is used in many applications, and is likely the next logical step for algorithms to implement. On the hardware side, integrating an AC to DC power supply in the PCB would greatly simplify the ease of use and performance for the user. While far too complex of a system to integrate in the scope of this class, adding this power supply would allow the user to plug the PCB directly into the wall, rather than a benchtop power supply. This would further reduce equipment needs, the complexity of the setup, and provide better results. Since our motor bus voltage does not need the configurability or precision of a benchtop power supply, the power output can be majorly increased, likely for less cost than a commercial benchtop power supply. This would allow motors to be driven at full power, as there were some issues with current limits being exceeded in benchtop power supplies during our bringup process, which would cause the motor bus voltage to drop as low as 6 V because the power draw of the motor while spinning exceeded the power output limit on the power supply.

References

[1] "Six step commutation," MATLAB Help Center, https://www.mathworks.com/help/mcb/ref/sixstepcommutation.html.

[2] A. Solovev and A. Petrova, "BLDC Motor Controller: Design Principles & Circuit examples," Integra Sources, https://www.integrasources.com/blog/bldc-motor-controller-design-principles/.

[3] M. Harris, "How to make advanced BLDC Motor Controllers," Altium, https://resources.altium.com/p/build-advanced-brushless-motor-controller.

[4] IEEE, "IEEE Policies - Section 7-8 - IEEE Code of Ethics," [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html.

[5] WEBENCH Power Designer. Texas Instruments. Available at: https://webench.ti.com/power-designer/.

Appendix A Requirement and Verification Table

Table X System Requirements and Verifications

Requirement	Verification	Verification status (Y or N)
All voltages required on the PCB except the motor bus voltage shall be generated on the PCB	This is verified through design, as if the architecture is correct, the only external power connection will be the motor bus voltage.	Y: Designed as such
All voltages generated on the PCB should have an accuracy of +/-5% around their set-point.	Usage of an oscilloscope to measure switching regulator voltage ripple under light and heavy loads for both the 3.3V and 10V rail at the extremes of the input voltage range will allow us to verify this requirement.	Y: Confirmed on multimeter at 12V and 24V
The PCB should function properly over an input voltage range of 12V to 24V.	As the input voltage only directly interfaces with the motor drive subsystem and power subsystem, if a motor can be spun at a speed of at least 1000 rpm and stopped, at both 12V and 24V input voltage, this requirement will be satisfied as the extremes of operation were validated.	N: Operates between 12-24V but fails to reach 1000rpm
The motor subsystem should be able to generate both trapezoidal and sinusoidal phase waveforms.	An oscilloscope can be used to view the waveform output on any individual phase, which should show a roughly rectangular pulse in trapezoidal mode, and many small PWM pulses of varying duty cycle in sinusoidal mode, or a reasonably sinusoidal waveshape when an inductive motor load is connected.	Y: Both waveforms confirmed on oscilloscope
The motor drive subsystem should have hardware restrictions on shoot through, and should have at least 2ns of time between FET transitions.	Use an oscilloscope to verify at least 2ns between one FET turning off and the other FET turning on for both possible half-bridge transitions.	Y: Shoot-through prevented by gate-driver IC, dead time ~300ns
Each half-bridge should be capable of 100kHz PWM frequency.	Using an oscilloscope, verify that a 100kHz, 50% duty cycle square wave can be generated on any of the three phases at a 24V input voltage.	Y: Square wave implicitly verified through PWM mode operation at 100kHz
All three phases should have voltage and current measured within 150mV and 100mA.	Using an oscilloscope, the actual voltage/current and reported voltage/current for each of the three phases can be measured and evaluated.	Y: Verified on voltage rails and through design (sensor is more accurate than this)
Any voltage rail generated on the PCB should have the voltage, current, and power	Again an oscilloscope and electronic load can be used to measure the actual voltage, current, or power, and can be compared to the measured value	Y: Reporting is verified through design, accuracy is verified through

reported. Measurements are for system health so an accuracy of 5% is acceptable.	from the sensor subsystem.	multimeter and sensor readout cross-reference.
Motor speed should be collected either by an encoder or through measurement of back EMF with +/-10% accuracy.	A tachometer can be used to get a reference RPM of the motor under certain conditions, and either our back EMF calculation, or an external encoder connected to the motor connected to the quadrature inputs should report a speed within 10% of the tachometer's reading.	N: Sensor read-speed limitation prevents speed measurements.
The microcontroller should be programmable and debuggable over serial wire debug (SWD)	Code to output a square wave on a GPIO can be programmed to the microcontroller. An oscilloscope can verify the code was delivered by looking at the GPIO, and the square wave should stop if a breakpoint is set and triggered.	Y: Verified in bringup.
The microcontroller should be programmable over USB C	Similarly, code to output a known signal on a GPIO pin can be uploaded over USB and then the GPIO can be measured to verify this requirement.	N: Unable to fix USB driver issues in time for this.
The microcontroller should report data in real time, at an update rate of greater than 1Hz and a latency of 500ms to the GUI application on the computer	The microcontroller can output a known waveform on the motor drive subsystem, and an oscilloscope can measure when a transition happens, and a timer can be started to measure the delay between the event occurring in real life and the arrival of the data on the computer. A 2Hz square wave output is a good waveform to test these quantitative limits.	Y: ~4Hz and <250ms reporting achieved.
Any failed connection between the computer and PCB should stop the motor from spinning within 2 seconds	Configure the platform to be spinning the motor. Unplug the USB link from the computer side and time how long it takes for the motor to stop spinning. If the data link is severed and the motor stops within two seconds, this requirement is verified.	Y: Verified by removing USB and motor stopping before 2s elapsed.