Cost-Effective and Easily Configurable High Voltage Motor Controllers for Automotive Use

DESIGN DOCUMENT

sdmay25-26
PRISUM Solar Car Club and Jonah Frosch
Nathan Neihart, Cheng Huang
Marek Jablonski, Jonah Frosch, Bryce Rega, Gavin Patel, Long Yu
sdmay25-26@iastate.edu
https://sdmay25-26.sd.ece.iastate.edu/

EXECUTIVE SUMMARY

The rise in electric vehicles has grown the market for automotive parts not commonly used prior. High voltage motor controllers use high voltage from an electric vehicle's battery pack and can be digitally controlled to feed the electric motors the correct power. There are many configurations and special features that come with equipment like these. A single high voltage motor controller can be thousands of dollars and difficult to use.

University solar car teams have been a decades-long testing bed for electric vehicles and the technology that has eventually made it to mainstream production vehicles. Iowa State has been a long-time participant in this and therefore making Iowa State's solar vehicles and their motors excellent testing beds for improved high voltage motor controllers. Custom, easily configurable motor controllers will be developed that can be used by the same motors Iowa State's solar car team as well as other vehicles (such as electric bikes) while being more affordable than current comparable motor controllers.

LEARNING SUMMARY

DEVELOPMENT STANDARDS & PRACTICES USED

List all standard circuit, hardware, software practices used in this project. List all the Engineering standards that apply to this project that were considered.

- IEEE 1547 - Standard for Interconnecting Distributed Resources with Electric Power Systems
    - This standard outlines requirements for safely interconnecting distributed energy resources (like solar and wind) with electric power systems, focusing on performance, safety, and reliability.
- IEEE 1149.1 - Standard Test Access Port and Boundary-Scan Architecture
    - This standard defines a boundary-scan architecture for testing and debugging integrated circuits and circuit boards, allowing access to internal signals for easier diagnostics without physical probes.
- IEEE 1554 - Standard for Software Engineering in the Life Cycle of Digital Systems
    - This standard provides guidelines for software engineering throughout the life cycle of digital systems, emphasizing structured practices for development, testing, and maintenance to improve software quality and reliability.

SUMMARY OF REQUIREMENTS

List all requirements as bullet points in brief.

- Technical
    - Electrical
        - Input Voltage range of 48V-135V
        - Capable of continuous 50A Current input transfer
        - Commutation speed from 0rpm to 900 rpm
        - Controlled through 250k CAN bus
    - Mechanical
        - Maximum recommended size of equivalent 200*200*100mm cube
        - Weight no greater than 4kg
        - Resistant against splashed water and dust
        - Able to operate continuously in a 45°C environment
- User Oriented
    - Lower than $5k cost to user
    - USB or UART port for debugging
- Other
    - No Aesthetic Requirements
    - CAN standard

APPLICABLE COURSES FROM IOWA STATE UNIVERSITY CIRRICULUM

List all Iowa State University courses whose contents were applicable to your project.

- EE 330 – Integrated Electronics
- EE 451 – Control Systems
- CPRE 288 – Embedded Systems
- SE 309 – Software Development Practices
- EE 224 – Signals and Systems
- IE 305 – Engineering Economic

SKILLS/KNOWLEDGE AQUIRED OUTSIDE OF CLASS

List all new skills/knowledge that your team acquired which was not part of your Iowa State curriculum in order to complete this project.

- High Voltage Circuit Design
- PCB Design
- CAN Protocol
- Simulation
- Embedded Systems Programming
- Control Feedback Loops
- Thermal Management

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

1.1. PROBLEM STATEMENT

Lightweight electric vehicles, such as electric bikes and solar cars, face a significant challenge when it comes to selecting motor controllers, which are essential for efficient and smooth operation. Currently, available options are limited. Many controllers are sold as mysterious black boxes with little to no ability for users to configure or customize settings to meet specific needs. Some controllers that do offer customizations are extremely expensive, placing them out of reach for many DIY hobbyists and solar car teams. Even when customizations can be made, the configuration and troubleshooting processes are very complex and often missing features that some users desire. This lack of affordable, user-friendly motor controller options hampers the broader adoption and innovation in the lightweight electric vehicles. The problem has been expressed by anyone from Iowa State's own solar car team, PRISUM, to students looking to build electric bikes for transportation. In this project will we be designing the architecture and implementation of a motor controller for lightweight automotive use.

1.2. INTENDED USERS

We have identified three main user groups for our product. We categorized these main user groups as "Solar Vehicle Engineers", "Jonah", and "Old Man McGee". These three categories cover our range of users from collegiate organizations as well as knowledgeable and unknowledgeable individuals.

The generic solar car engineer is a student at your local university looking to build a vehicle for the American Solar Challenge. He is looking for a motor controller that allows reliable operation with good documentation that does not require in depth knowledge of the system to use but the ability to dig into more details later. This engineer needs a way to control motors on a budget because existing solutions are prohibitively expensive. This user group will be aided by our design by having an affordable and reliable motor controller that they can use for their competition vehicles.

Jonah is a bored biker that takes a lot of interest into mountain biking. He enjoys riding down slopes but not working his way back up to the tops of hills, and as such is hoping to motorize his bike in a high torque way that can move his bike under battery power to the top of trails so that he can fully enjoy the experience. Jonah needs a way to use the motor his bike while keeping costs low because existing solutions are either too expensive or not general enough for his use case. This user will benefit from an affordable product that has a relatively high output power.

Old Man McGee is not super tech savvy, but has a basic understand of what's important. He's interested in using this motor controller his work at a Detroit factory or personal transportation. Old Man McGee needs a way to move his custom vehicle that is easy to use as he isn't incredibly knowledgeable on motors. This user will benefit from the good documentation and easy debugging that the controller will have.

2. Requirements, Constraints, And Standards

2.1 REQUIREMENTS & CONSTRAINTS

2.1.1 FUNCTIONAL REQUIREMENTS

The motor controller must be able to spin the motor in either direction under controlled current from 0-3500 electrical rpm.

The controller should be able to support voltage ranges seen by electric bikes and as high as used by solar cars, 48-135VDC (constraint). The controller shall be capable of a continuous 50A current draw (constraint).

The motor controller supports receiving CAN messages for configuring settings on the motor controller as well as real-time use (such as throttle control, cruise control, regenerative braking control). It will also send CAN messages for statuses (such as RPM, voltage and current, and errors). These CAN IDs can be configured so that they better support the specific application they are in and can match the CAN IDs of other motor controllers for ease of swapping controllers. The CAN bus shall be capable of 250k operation.

A secondary objective is to support non-CAN based communication for basic use. For example, an analog voltage input for throttle and regenerative braking, a digital input for direction, and a PWM output of varying frequency for reporting RPM.

2.1.2 RESOURCE REQUIREMENTS

The use of a microcontroller is required to drive at least 6 PWM channels using 3 complex timers designed for motor control (constraint). The microcontroller must also have other timing means not already occupied with PWM output.

If the microcontroller does not have hardware for handling CAN, a separate chip for CAN messages will be used and can communicate with the main microcontroller over an auxiliary communication bus such as SPI or I2C.

Additional communication is also required in order to be easily swappable with other motor controllers. This includes an analog input for throttle and regenerative braking control as well as GPIO for direction control. While these parameters can be specified in CAN messages, not all systems using this motor controller will be using CAN to interface with motor controllers.

The current-carrying switches must be capable of handling a peak current spike four times the steady state average to permit the use of low-inductance motors (constraint). Multiple switches in parallel are permitted given other functional and physical requirements are met.

2.1.3 PHYSICAL REQUIREMENTS

The motor controller shall fit in the footprint of existing motor controllers for solar vehicles (constraint). Those motor controllers act as a maximum since they match the highest power this controller will be used for and are larger than less powerful electric bike controllers. The maximum recommended size is 200mm*200mm*100mm in size.

The high voltage inputs and motor power terminals shall be bolt down lugs protected from environmental contamination with adequate electrical isolation to prevent arcing. Low voltage control and communication shall occur through weather resistant connectors.

The controller shall be able to continuously function with an ambient environment temperature of 0-45 degrees Celsius (constraint).

The weight shall be no greater than 4kg (constraint).

2.1.4 AESTHETIC REQUIREMENTS

The motor controller shall have appropriate labels for safety purposes meaning there must be adequate rooms for warnings such as high-voltage (constraint). Other labels for ports and terminals should be made clear.

2.1.5 USER INTERFACE REQUIREMENTS

The labels included on the motor controller, as mentioned above, shall be written out clearly. This means there is more than just a "+" for a positive terminal, but specifically labelling what positive terminal it is in such a way that it cannot be confused with any other possible input/output to/from the motor controller.

The interface used to troubleshoot and configure the motor controllers from a computer may be done over GUI. Commands will be presented in a professional manner with clear instructions for the user.

2.1.6 USER EXPERIENCE REQUIREMENTS

The user should be able to easily navigate the documentation to mechanically mount their motor controller and electrically connect their motor controller by including all the necessary specifications.

Configuring the motor will be done by utilizing well named macros within the code itself. If implemented, the user will be able to utilize a GUI to update the controller with applicable parameters..

2.1.7 ECONOMIC/MARKET REQUIREMENTS

The final cost of the motor controller should be less than one fifth of what is available for powering solar vehicles and should be comparable to a high-end electric bike motor controller. Our target is less than $2000.

Components shall be accessible through public distributors (Digikey, existing stock, etc..) for order in the event of a replacements.

2.2 ENGINEERING STANDARDS

Engineering standards are important because they ensure safety and guarantee functionality. These are vital to ensuring that the things we use daily are reliable and safe. They also allow for an increase in the simplicity of both product development and use since they can reduce the need for unnecessary duplication. They can also codify best practices to spread the institutional knowledge to all those involved in that field of practice.

- IEEE 1547 - Standard for Interconnecting Distributed Resources with Electric Power Systems

    - This standard outlines requirements for safely interconnecting distributed energy resources (like solar and wind) with electric power systems, focusing on performance, safety, and reliability.
- IEEE 1149.1 - Standard Test Access Port and Boundary-Scan Architecture
    - This standard defines a boundary-scan architecture for testing and debugging integrated circuits and circuit boards, allowing access to internal signals for easier diagnostics without physical probes.
- IEEE 1554 - Standard for Software Engineering in the Life Cycle of Digital Systems
    - This standard provides guidelines for software engineering throughout the life cycle of digital systems, emphasizing structured practices for development, testing, and maintenance to improve software quality and reliability.

All of these are applicable in some way to what we intend to accomplish. The first applies to connecting the battery-powered motors to wall power. The second is directly relevant because the need to test and debug integrated circuits and allow for access to internal signals is essential if someone needs to debug our motor controller or when we need to during product development. The third is also applicable because we are writing the software required for this system to run, so adhering to this standard is essential for the system's reliability.

The most significant modification to adhere to these standards is with IEEE 1554. This is important because it directly applies to what we are doing and outlines how to structure our software delivery process, from development to testing to deployment. This guide outlines a structured approach recommended for the board's development.

3 Project Plan

3.1 PROJECT MANAGEMENT/TRACKING PROCEDURES

Our team is adopting two different management styles for the two sub-teams. The software sub-team is adopting an agile management style to allow for rapid iterations and testing without requiring each element to be complete. The hardware sub-team is adopting a hybrid agile-waterfall method. In this, the overall project is managed in an agile form, with iteration and feedback happening between iterations of the full hardware system. Within an iteration however, the tasks will follow the waterfall management style from a theoretical design, to a schematic, and then a fabricable hardware design.

Our team will be using GitLab issues based on our generated Gantt chart to keep track of our progress for the remainder of this project. The milestones and filtering ability of this setup will make management easier and allow for greater transparency. It is also built into our existing GitLab repository so we don't have to set up any new pieces of software.

3.2 TASK DECOMPOSITION

The first phase is the research phase. This involved researching components and software libraries to get a better understanding as to what we are either trying to use or emulate in our initial product.

- Research and select a development board
    - Search for boards with software and hardware capabilities for even just barely testing the solar car motors
- Research component categories
- Setup software environment
- Research ST's MCSDK library for controlling motors with PWM
    - Shape software plans with this PWM information in mind

The next phase is prototyping phase. This involves doing the design work to create a PCB and initial software minimum viable product. This is a development phase.

- Test development board using ST's MCSDK library and generated code
    - Generate different versions of code based on different motor and controller configurations
    - Test on different motors
- Complete a schematic for the first board revision
- Complete layout for the first board revision

After the developing a prototype, we can assemble and test a first revision—both in hardware and in software.

- Test generated example code on PCB to determine the hardware functions
- Test MVP software on PCB
- Make adjustments to the software to ensure it is functioning
- Document shortcomings of the PCB design to change for the next revision
    - Being able to spin the motor is different than spinning it well
    - Document user experience shortcomings from a hardware perspective

The final design phase of our second and last revision will involve using lessons learned from the first revision and preparing to make a final product.

- Simulate new components for meeting the specs of the final product
- Complete a schematic for the second board revision
- Compete layout for the second board revision
- Develop additional features and configurations missing in the initial version of the software
    - Build in features to support full range of motor
    - Support different communication types if time permits
    - Develop cruise control
        - Develop portable PID controller

The last phase is the final assembly and testing phase. We will ensure our final product works and investigate how it measures up to what we wanted as well as to what already exists out there.

- Assemble the final PCB revision
- Run previous software on it to ensure basic functionalities work
- Run new software changes to explore the full capabilities of the motor controller
    - Ensure motor can spin up to speed and under load as expected
    - Test cruise control
    - Test different faults, warnings, and other troubleshooting methods
- Measure efficiency of the motor controller setup and compare to expectations
- Make adjustments to software as needed
- Make hardware adjustments as needed
- Develop an enclosure and hardware interface that meets our user experience requirements

3.3 PROJECT PROPOSED MILESTONES, METRICS, AND EVALUATION CRITERIA

Our first milestone is determined by having the first working prototype. This is quantified as spinning the target motor (a Mitsuba M2096-III) at a voltage within some portion of the expected working range (48-135VDC). The speed of the spinning motor will be controlled through an analog input. There is no power output specification for this milestone besides overcoming inertia and static friction.

The second milestone is our first fully custom controller. This iteration must operate along the full input voltage range (48-135VDC) and must be capable of operating at half of the specified maximum amperage output (25Arms). The motor shall be able to be spun from complete stall (0rpm) up to the expected maximum 800rpm (at 135VDC).

Our third major milestone is a controller that meets all critical requirements. It must operate along the full input voltage range (48-135VDC) at the full specified current (50Arms). The motor shall be able to be spun from complete stall (0rpm) up to the expected maximum 800rpm (at 135VDC) in both directions with current-based control. This iteration shall require either no auxiliary power supplies or a single external 12V supply that is not required to referenced to the high voltage ground.

As this project continues, we will be refining our milestones, metrics, and evaluation criteria based on previous outcomes. It is likely we will have more milestones regarding how a final design will meet the critical requirements for this project.

3.4 PROJECT TIMELINE/SCHEDULE

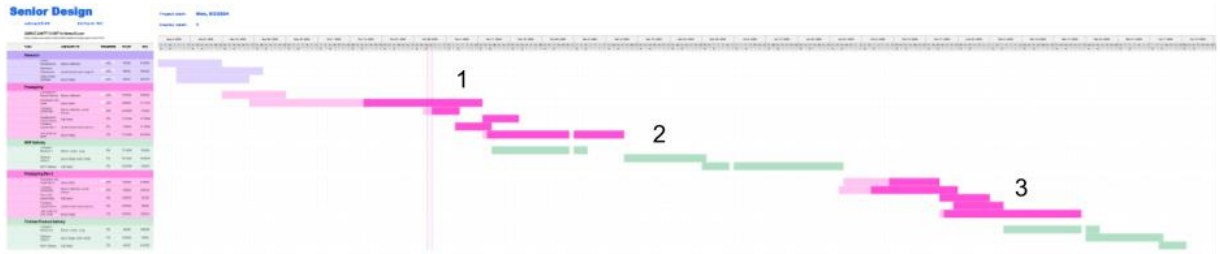

Figure 1. Project Gantt Chart Overview

The Gantt Chart shown above summarizes the total timeline of the project broken down by subtask with the three major milestones annotated on top. The five distinctly colored sections correspond to the five stages identified in our task decomposition. The major milestones are placed at the end of their respective stages.

Milestone 1 corresponds to the completion of our first working prototype. This is placed timing wise at the first week of November. The milestone does not correspond to an exact spot in the combined timeline as it comes at different points in the hardware and software development processes.

Milestone 2 corresponds to the completion of the first custom built controller. This milestone is placed timing wise at the end of the first semester. This is conservative estimate of our capabilities but will act as a “worst case” time constraint this semester.

Milestone 3 corresponds to the completion of our second major controller revision, aimed at meeting all critical requirements. This is placed in mid-March. Planning to complete our project early in the spring semester gives us more time to put finishing touches on the project and prepare documentation and presentation material.

3.5 RISKS AND RISK MANAGEMENT/MITIGATION

- Research - Associated Risks
    - Time Overruns: 90% Probability
        - Mitigation: We are checking in with our advisors and tracking our current timeline against the plan to keep the project on track and move to the next stage
- Prototyping
    - Design Time Overruns: 90% Probability
        - Mitigation: assign set times outside of regular senior design meetings to ensure work is being put into designing the Hardware and software.
    - Test Motor Failure: 10% Probability
        - Mitigation: We already have backup hall sensors ordered and additional motors and components for repairing any potential damage on standby.
    - Development Board Damage: 10% Probability
        - Mitigation: All testing will be performed incrementally upwards with protective limits put in place by the test equipment to minimize avenues for potential damage. In case of damage software is able to pivot to higher level testing until replacement parts arrive.
- First Revision
    - Component Failure: 75% Probability
        - Mitigation: Ensure that all expected and most intensive use cases are properly simulated in LTspice to determine and mitigate likely points of failure. We will be ordering spare components to replace any failed or damaged components.
    - Software Delay: 40% Probability
        - Mitigation: Utilizing different MCUs can cause complexity issues, we plan on focusing our efforts on a single MCU to get it working, then adapting our code to be able to support multiple MCUs.
    - Hardware Shipping Delays: 20% Probability
        - Mitigation: Order hardware as early as possible, allow leeway for shipping to take longer than expected.

- Second Prototype
    - Design Time Overruns: 30% Probability
        - Mitigation: assign set times outside of regular senior design meetings to ensure work is being put into designing the Hardware and software.
    - Overspending of Budget: 43% Probability
        - Mitigation: purchase any components in the first revision in bulk if we feel they will be reused in the second revision.
- Second Revision
    - Component Failure: 55% Probability
        - Mitigation: Ensure that all expected and most intensive use cases are properly simulated in LTSpice to determine and mitigate likely points of failure, including inconsistencies discovered in the first revision. We will be ordering spares to replace any failed or damaged components.
    - Software Delay: 15% Probability

3.6 PERSONNEL EFFORT REQUIREMENTS

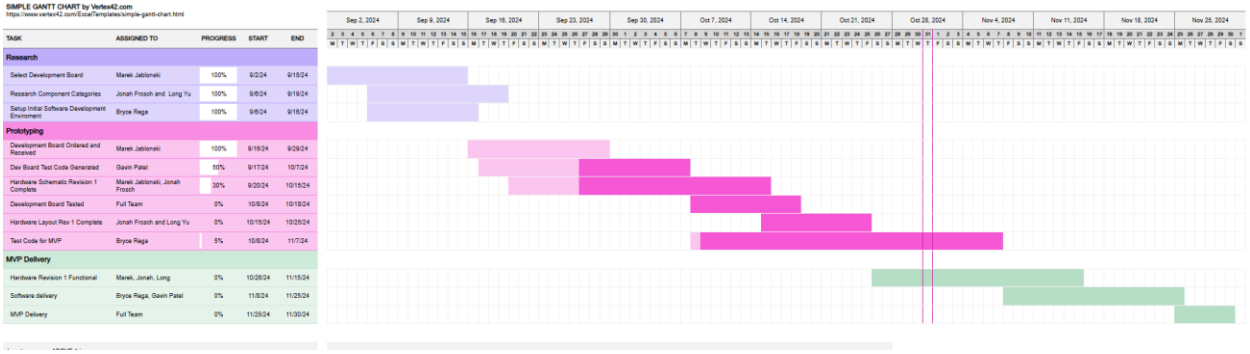

Figure 2. Project Gantt Chart personnel effort

Task	Hours Est.
Select Development Board	8
Research Component Categories	14
Setup Initial Software Development Environment	10
Development Board Ordered and Received from Detroit	5
Dev Board Test Code Generated	22
Hardware Schematic Revision 1 Complete	27
Development Board Tested	15
Hardware Layout Rev 1 Complete	12
Test Code for MVP	32
Hardware Revision 1 Functional	18
Software Delivery	20
MVP Delivery	6

3.7 OTHER RESOURCE REQUIREMENTS

Completing this project requires the use of benchtop multimeters, high voltage power supplies, and Oscilloscopes for testing of the prototypes. The motor we are using for our testing (Mitsuba 2096-III) has been supplied by the Client. Software required will be entirely free to use and includes:

- STM32CubeMX – generating firmware libraries for ST chips
- STM32CubeIDE – IDE and debugger for ST projects
- ST-XCUBE-MCSDK – Software for generating motor controller libraries for our development board and test motor
- VSCode – Code editor
- GitLab – Code repository hosting
- Git – Version control
- ST-Link V3 – the physical programmer and debugger connecting a computer to the micro
- LTspice – Simulator for components to test schematic prototypes before building physically
- KICAD – PCD CAD design software used for designing the schematic and printed circuit board

4 DESIGN

4.1 INTRODUCTION

Our team's project has a wide variety of variables that contribute to the design choices we are making, from cost to efficiency to feasibility, that we are considering at every step. In order to ensure that the final product meets all requirements, we need to ensure that we make the proper choices when designing our controller so that little rework or redesign is needed in the future.

4.2 DESIGN EXPLORATION

4.2.1 Design Decisions

There are several key choices in our design that greatly influence how the final controller will take shape. A few of these design choices are:

- Deciding what kind of phase driving transistors to use: When dealing with power at the scale of a small vehicle, many standard Transistors are not equipped to handle driving such a large load. The amount of current being pushed through the transistor would be enough to fry most transistors, so care will need to be made to ensure the transistor can handle such a high load, and will not cost a fortune to procure.
- Selecting the proper MCU: While a variety of MCUs are equipped enough to handle the same tasks, our project development process will be much smoother if we select one that is optimized towards our needs. Ease of interfacing with components we already know we need, such as the above-mentioned high-power transistors, ease of programming, and extra features commonly seen in vehicles (such as containing a native CAN bus interface).
- Deciding if we want a sensorless mode: If we wanted our controller to be as compatible as possible, we could enact a sensorless mode for running motors off of our controller. This would expand the range of applications this project could be used for, but would require significant software investment and detailed voltage monitoring of individual phases.

4.2.2 Ideation

For deciding what kind of phase driving transistors to use, we utilized the electronics distributor DigiKey to assist in narrowing down our options. The site has filters users can use to sort through thousands of components to find ones that fit their specific needs. In addition to that, we also examined an existing motor controller to take into account the transistor it used. Major considerations we gave when selecting the components were if they could hold up to the expected loads, if they were affordable, and how efficient they could be (in other words, what was the expected power loss across the transistor). Some of our top choices were:

- STGW30H60DFB. This component is classified as an IGBT, which is a special kind of transistor highly capable of dealing with high power loads. This specific component fits that description, capable of supporting up to 60 amps and up to 600 volts, well above our 150 volt 50 amp limits. Additionally, this component is relatively cheap, at only $3.40 per component. One dramatic downside that this component has is that it has a significant inefficiency, with a 2V drop from the collector to the emitter, which would

equivalate to nearly 100 watts lost at this component alone when under full throttle. This inefficiency quality is widespread throughout IGBTs, so we will be utilizing a high power MOSFET.

- IRFP4227PBF. This component holds a much higher efficiency, with a measured 25 mOhms of resistance (an equivalent 1.25 volt drop at 50 amps, but only more efficient at lower currents). Fitting within all of our limits and costing $5.16 per piece, this component isn't a bad choice.
- SQM90142E_GE3. Moving from the last choice, this component is much more efficient, at just 15.3 mOhms of resistance. It also has a much higher current limit, at 95 amps max, and while both fit under our requirements, having that margin of safety is a factor worth considering. Costing just $3.91 per, it would be an ideal choice, save for the fact that it is entirely SMD. There is no viable way to mount a heatsink to this component capable of dissipating potentially dozens of watts, as this component was designed to dissipate heat into a board.
- IRFP90N20DPBF. This component fits all required specifications, has a resistance of 23 mOhms, has a mechanical screw mounting hole for heatsinks, and is through hole mountable to the board, meaning we have more flexibility in how we place it relative to heat sinks. It has reasonable limits to voltage and current, at 200V and 94A, but its most major drawback is the climb in cost we start to see, at $7.15.
- IRFP4668. This component cost by far the most, at $7.75. It also has the most padding to its limits, at 200V and 130 amp, and a staggeringly low 9.7mOhms of resistance from its drain to source. It also retains the through hole mounting to board and screw mountable area for heat sinks and heat dissipation.

4.2.3 Decision-Making and Trade-Off

The process we use for identifying pros and cons between our choices involves identifying, if any, the numerical gains or losses from one option to another and comparing the design effort required to make up the losses elsewhere. For instance, the transistor we chose is the IRFP4668, due to its tremendous efficiencies compared to its alternatives, a runner up example being the IRFP90N20DPBF. Dropping from around 23mOhm to 9.7mOhm yields up to an additional 33 watts saved at our most power-hungry point, at the cost of only $0.60 more per component (from $7.15 to $7.75). Saving that much power elsewhere both in and outside of the scope of our project for the same cost is virtually infeasible, so the con of choosing the higher cost component far outweighs the cost of inefficiency from not using the part in the first place.

4.3 PROPOSED DESIGN

4.3.1 Overview

The current design consists of two major blocks that connect together the input connections, the power source, and the driven motor. The software processing blocks takes the

acceleration inputs and the information from the motor to determine how to drive the motor in the desired manner. This is communicated to the hardware power stage that amplifies the control signals using power from the battery or power supply to drive the motor.

4.3.2 Detailed Design and Visual(s)

The full system consists of the motor controller, the control input, a power source, and the attached motor. The control input consists of a selectable GPIO or CAN bus input according to user specification. This connects to the motor controller and feeds into input signal software processing. This input “command” along with the current speed and output power are fed into the acceleration control software module. This module consists of PID controllers attempting to match torque, speed, and flux output based on desired values and current outcomes. The calculated target magnitude is then passed into the sinusoid generator software module. This complex module takes in the current rotor position, speed, target amplitude and generates the required magnitudes of the three output phases (including target polarities). The next software the stage is the PWM converters, taking the target phase amplitudes, deciding which gate of the driver will be locked (for power efficiency) and the required PWM outputs of each stage. This stage is also responsible for accounting for physical non-idealities such as keeping the high-side bootstrap capacitors charged. Current control would require detailed physical parameters of the motor itself or adaptive feedback, but would allow for a control method more similar to the one currently used in existing controllers.

The PWM Converters control the first physical hardware, the triple-half bridge. This stage consists of the Physical Gate Driver, the Power Transistors, and their supporting circuitry (power supplies and filtering). The Triple Half-Bridge stage receives its large power input from the vehicle’s traction battery or a testbench power supply and outputs to the three motor phases. The Motor has three embedded Hall Sensors for position sensing positioned 120 electrical degrees apart. This allows the controller to know the position of the rotor within 6 distinct positions and allowing interpolation for increased fidelity. The raw input from the sensors is fed into the onboard microcontroller and processed to determine current rotor position and speed. These processed values are fed back into the output stages described previously.

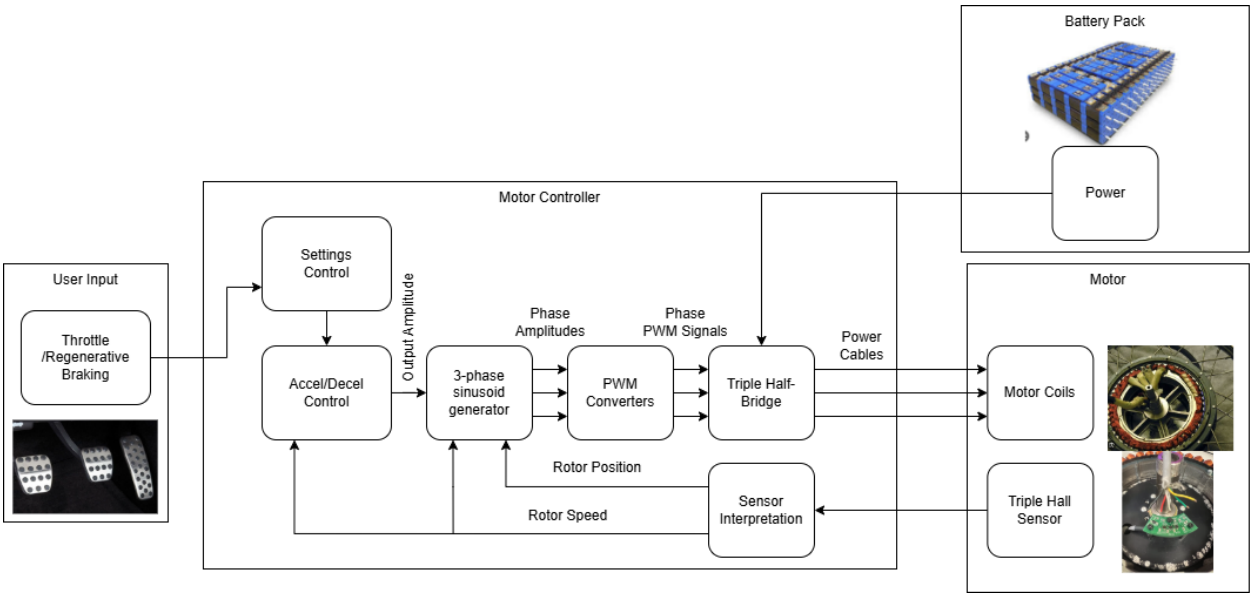

Figure 3. System design diagram

4.3.3 Functionality

There are two main stages of operation a user will have with product, setup and runtime. In setup a user will input the parameters of their motor and system into the prebuilt software and update the parameters of the controller. In runtime use the user will communicate the acceleration/regenerative braking commands to the controller though the vehicle's controls, reaching the controller as either a CAN bus message or a GPIO signal. The controller then responds by accelerating or braking the vehicle accordingly. If the user does not think about the controller while it is in use it is doing the job well.

4.3.4 Areas of Concern and Development

Our current design meets all hardware specifications, but not all user-experience requirements, as we currently are incapable of doing a "quick setup" with a motor. This means that the user has to manually get different measurements from the motor and input them into the code for the controller to work effectively. This requires expensive equipment and some knowledge of how the inner system works. Based on our current design, our primary concerns for delivering a product are our condensed time table. Large setbacks towards our project can have very drastic impacts on our ability to get the project complete because of the lack of slack in our calendar. Our immediate plans for mitigating this issue involve assigning time outside of our designated meeting times to also get more work done on the project to keep it moving steadily. One question we have for our faculty advisers are what sort of safety precautions they would recommend we implement when it comes to full testing the motor with load on it.

4.4 TECHNOLOGY CONSIDERATIONS

A few of the distinct technologies that we are using for the development of this design are as follows: benchtop multimeters, high-voltage power supplies, and Oscilloscopes for testing the prototype. For the development of software specifically we will be using

STM32CubeMX for generating firmware libraries, STM32CubeIDE as the IDE and debugger for our development board, ST-XCUBE-MCSDK for generating motor controller libraries for the development board, VSCode as a code editor, GitLab for code repository hosting, ST-Link V3 for the physical programmer and debugger, LTSpice for schematic simulation and testing, and KICAD for PCB CAD design. Some of the strengths of these products in general are that they are free to use for us as either Iowa State Students or they are just free to use in general. This allowed for us to have some familiarity on how they work prior to us starting this project. Specifically, when it comes to the ST libraries and code generation, the UI gives you a good amount of customization and allows for you to have lots of examples online on how to use and read the code generated. Some of the weaknesses come from the fact that because we ultimately plan to use a microcontroller that can support CAN, the ST programs are very helpful for skeleton code, but ultimately all of it will have to be rewritten to fit the new microcontroller that we plan to use. Along with this, a weakness of LTSpice is that you can either have a detailed simulation or a lengthy simulation, but achieving both of those is quite difficult. A possible solution to this issue is for the short-term making a more simplistic model so the simulation can run faster, then once the cruder architectural design is finished, to input the more complicated model to get a more accurate simulation.

4.5 DESIGN ANALYSIS

So far, we have built a testing rig with a real application motor (Mitsuba M2096-III) so that the hardware and software can be evaluated. We have also simulated various components of our high-level circuit using LTSpice. The simulations confirmed that our current schematic design will in theory work well with the components we have selected. Our proposed design from 4.3 has not been implemented yet, but based on the test hardware that we have, there have been important lessons learned such as how voltage adjustments impact motor at the startup. We are able to control the torque and speed of the motor, but we are facing semiregular faulting and issues when trying to control the actual speed of the motor. These issues seem related to high frequency noise in signal feedback issue or the calculation in the code, and will need further investigation. Some of our next goals are finishing our design so we can order rev 1 to begin testing of our hardware and software together, along with ironing out the issues we are currently facing when it comes to the faulting and speed control. We need to troubleshoot this to ensure that it does not impact our future implementation. Another thing we want to implement soon is the “fast setup” of a motor to the controller, so a user doesn’t have to do the calculations themselves. After our first revision we learned that the pinout that we selected limited the effectiveness of our controller due to limitations of the MCU itself. This was a key change we made to our second revision.

5 TESTING

5.1 UNIT TESTING

Most software functions will be subject to unit tests to ensure that key functions work as expected. Testing these functions with expected inputs will be done as well as possible edge cases that ensure the function doesn't cause unexpected issues. Using GitLab's built in CI pipeline will allow us to run unit tests upon each code push. This way our code is continuously checked and will fail tests if an update inadvertently breaks previously working code.

Before hardware revisions are designed or ordered unit simulation tests must be run on the designed circuit. These simulations are primarily completed using LTSpice and manufacturer-supplied spice models. These simulations are run at expected maximum stress conditions, ensuring all device parameters remain within tolerances. Current hardware unit tests include the following:

- Gate Driver Unit Test
    - Ensure gate current remains within specification
    - Ensure bypass supply remains stable
    - Record Gate Rise-Fall Times for other simulation tests
- Input Stability Unit Test
    - Ensure acceptable input stability
        - Minimal Ringing
        - Acceptable Droop
    - Helpful to determine minimum expected capacitance

These hardware simulations are helpful as we are able to perform the same tests on our physical hardware. This allows us to not only validate correct performance but the validity of our simulations. We were also able to perform physical tests and verify that the Gate Driver worked as we expected. The rise and fall times were within an acceptable margin of error to the simulation results.

5.2 INTERFACE TESTING

A basic command line interface will be developed and can be used to set configurations and test the motor controller. Unit tests will be run on the command line interface. Nominal values can be used to ensure basic operation works, but edge cases or other misinputs can be tested to ensure that no input from the user could damage the system. For example, inputs of the wrong type or inputs that are out of the expected bounds will be tested for. If a non-negative integer is expected, but the user enters a negative float as an argument, the interface should safely handle it be converting the float to an integer and exiting by prompting the user with an "invalid argument" option or "parameter out of bounds" before continuing.

If a GUI is developed, similar tests will be run to the command line interface. In addition, tests will be run to prevent graphical bugs or confusing situations for the user. Unit tests are more difficult to run in this situation, but the same potential issues will be tested for. The example of entering an incorrect data type and data that is out of bounds can be used here by

entering the erroneous data into its respective input box and ensuring that a similar error is thrown.

5.3 INTEGRATION TESTING

With an assembled board, software integration testing can begin. This involves spoofing inputs and measuring outputs (such as pins, waveforms, signals) to make sure the program as a whole works as expected. For example, a spoofed CAN message is sent to increase the throttle, and the PWM signals across each of the three phases should react accordingly. Of course, without a motor connected, some features would need to be disabled or spoofed (such as calculations or sensors related to the position of the motor.

If a signal does not react as expected, the first step in troubleshooting will be to determine if the problem is hardware related, software related, or both. The first revision will involve a number of debug pins to measure what might be occurring at intermediate steps. For example, a debug pin could be used to investigate a signal coming from the microcontroller before it reaches more of the circuitry. Here a digital logic analyzer or oscilloscope could be used. If the intermediate step is acting as expected, but the final output after the circuitry is not, then there may be a hardware issue.

Software issues can also be investigated with a debugger. PRISUM's software environment includes a debugging setup for the microcontroller we are using. ST also has a debugger of their own for their chips. In each case, variables concerning state machines can be watched to troubleshoot logic and program flow issues, and the exact values of a peripheral's register can be examined to ensure the firmware layer and any setup is working as expected.

Hardware Integration testing following as an extension of the hardware unit testing. Connecting together the power stages (input, triple-half-bridge, motor, and gate driver) is first done in LTSpice simulation, built off the results of the unit test simulations. This full test allows a full integrated hardware test to be performed, allowing for accurate simulations of the entire powertrain under a maximum expected load. The stimuli for these tests are programmatically generated from unit test and mathematical results, allowing for quick, minute scale simulation runs where a fully detailed simulation would take days on end. As it is impractical to run physical integration testing by itself, this testing instead falls under system testing.

5.4 SYSTEM TESTING

In order to reach the system testing stage, all portions of a design must pass their applicable unit and integration tests. For this project, system testing will be performed on an idle test stand and in a functional vehicle. All specific electrical and mechanical requirements are able to be tested from these system tests.

On an idle test stand the motor controller will be connected to an AC-DC High Voltage power supply acting as a power source and a Mitsuba 2096C-III motor mounted to a test stand. The Controller will be connected to a computer for configuration and requirement-defined user inputs. These will allow basic control functionality (forward/idle/reverse) to be tested with the full voltage input range and output speed range. This test is incapable of testing constant power output but it capable of testing acceleration torque and efficiency.

On-vehicle testing involves connecting the controller to the battery, control systems, and motor of an existing vehicle. This testbed allows for much better endurance and long-term high-power testing but makes measurement precision and test repeatability much more difficult. This test environment is an important portion of acceptance testing.

5.5 REGRESSION TESTING

Software testing will be done with larger commits and will involve motor testing to ensure that the operation is still correct. Before being tested on the motor itself, waveforms will be analyzed to ensure that it is still operating as expected.

Code that oversees managing current levels as well as managing other safety features (such as other limits or safe states) must be unit tested and added to the pipeline. This part of the code has the potential to damage the system as a whole.

Regression testing is not applicable for hardware elements and would fall under unit and integration testing.

5.6 ACCEPTANCE TESTING

Several system tests will be run to ensure the requirements expressed by the client were met. User inputs into the system (CAN messages, analog signals, and any basic GPIO input) will be used to demonstrate control of the output of the system as expected (the power drawn matching the throttle, the direction of the motor matching the input, and any additional features being commanded by the different inputs that affect the motors). Reliability tests will be done with extended use of the deliverable.

The motor controller will be integrated with test vehicles as specified in the System tests to prove product suitability and reliability. Running environmental vibration and temperature tests are used to confirm compliance with other non-functional requirements.

5.7 RESULTS

The first stage in our project was using a purchased development board. It wasn't powerful enough to get anywhere close to meeting the requirements of the user, but tests on efficiency and software operation were still able to be run. After debugging and running tests on the motor itself to measure more accurate motor parameters, the motor was able to spin in either direction and can be throttle controlled. However, when testing for reliability, the system was not so successful. The longest period the motor was powered for before throwing an error was 23 seconds. It was worth noting that reliability was better when the motor was powered at a lower voltage (90V) as opposed to higher ones that were tested (105V and 130V).

The first revision is still in progress, but preliminary simulations are promising. The full power-stage is capable of relaxed power and full input filtering continuously with expected efficiency of >90%. The successful results have passed the simulation unit and integration tests and is in the final hardware design stages.

Testing the second revision with finalized hardware and near-finalized software gave promising results. The maximum speed range that was able to be achieved was 1050rpm, a full 17% above the target speed. Due to delayed timelines and equipment accessibility, full load

tests have been unavailable, but the performance metrics are inline with our predictive simulations that estimated >90% efficiency under load.

6 IMPLEMENTATION

6.1 DEVELOPMENT BOARD

An ST development board was ordered and integrated into our motor testing rig which involved a high voltage power supply, the controller, hardware and software for editing firmware, a GUI for controlling the controller, and a Mitsuba 2096-III motor attached to a cart.

With the setup, code was generated and flashed onto the microcontroller on the motor controller and the GUI was used to start testing spinning up the motor. These tests yielded issues with pinning down motor parameters and also revealed some errors in the generated code. With these issues being partially fixed, the motor was able to be controlled rudimentarily.

The generated software from this prototype is a trusted reference for developing custom software on the first fully custom revision.

6.2 CUSTOM REVISION – SOFTWARE

The frame of the first fully custom revision's software has begun development. A modular design was developed classifying code functionality into their own blocks of files and then classifying those blocks into layers. Those layers were designed with the intent of keeping everything tidy– meaning that a given layer can be used by the layer above it or use functions from the layer below it, but should not skip steps. This has been reflected by the code being written so far.

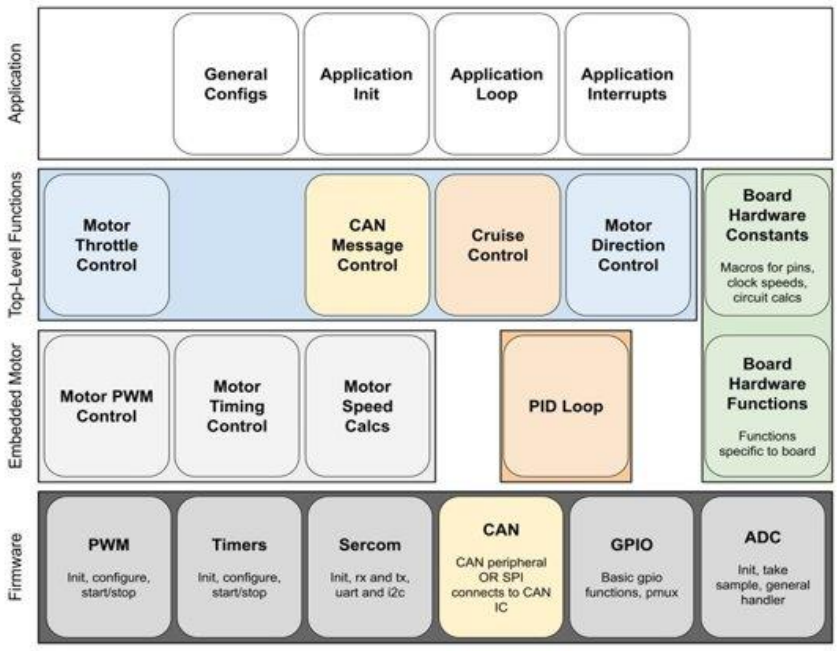

Figure 4. Software module implementation

When starting from scratch on software side of the project, it was important to start with what was immediately known. As shown in the figure above, the layers towards the very top and bottom have some level of completion (indicated by the check marks) whereas the middle layers tend to be barer.

The top layer is the “application layer” which involves high level function calls and generally describe how the program will behave. This isn’t too complicated and includes little technical details, so this layer has begun development with empty functions being written out.

The bottom layer is the “firmware layer” which directly interfaces with the peripherals on the microcontroller. These are very hardware specific so the layer is designed to be swappable depending on the microcontroller used. Empty functions were first written out to set a standard. These functions were initially left empty and were tweaked until a comfortable standard was set. This standard being the functions that could be called by the layer above no matter what microcontroller was being used. What makes this different from the top layer is that there is technical detail and that this technical detail is known. Development has started on “filling in” these skeleton functions with lines of code from the CMSIS library to write to the microcontroller’s peripherals’ registers. Functions related to controlling GPIO, ADC, CAN, I2C, clocks, etc... are included.

The middle layers involve the “nitty gritty” functionality of the motor controller. This involves timing phases, estimating motor position, and measuring current among other controller logic. This level of development involves a deeper understanding of the workings of a motor controller when compared to the top layer. By calling upon the firmware layer (as opposed to direct register writes), this logic can be ported from one MCU to another. This is a similar technique as was observed on ST’s development board software.

6.3 CUSTOM REVISION – HARDWARE

We have completed multiple revisions of our hardware. Our first revision has two different layouts that we designated as revision 1.0 and 1.1. Our 1.0 Revision involved only 1 board but due to the easier design process and assembly two boards would have, revision 1.1 involved two boards instead of one. This would have the signal board, consisting of the MCU and signal and debugging pins, placed on top of the power board, which would have the gate drivers, power mosfets, and capacitors.

After the hardware layout is finalized and reviewed, it will be ordered and assembled before entering unit and integration hardware testing.

The top-level schematic for revision 1.0 can be seen in the figure below. This schematic integrates the microcontroller, phase driver, and capacitor bank subsystems with each other as well as providing input filtering and basic interface tasks.

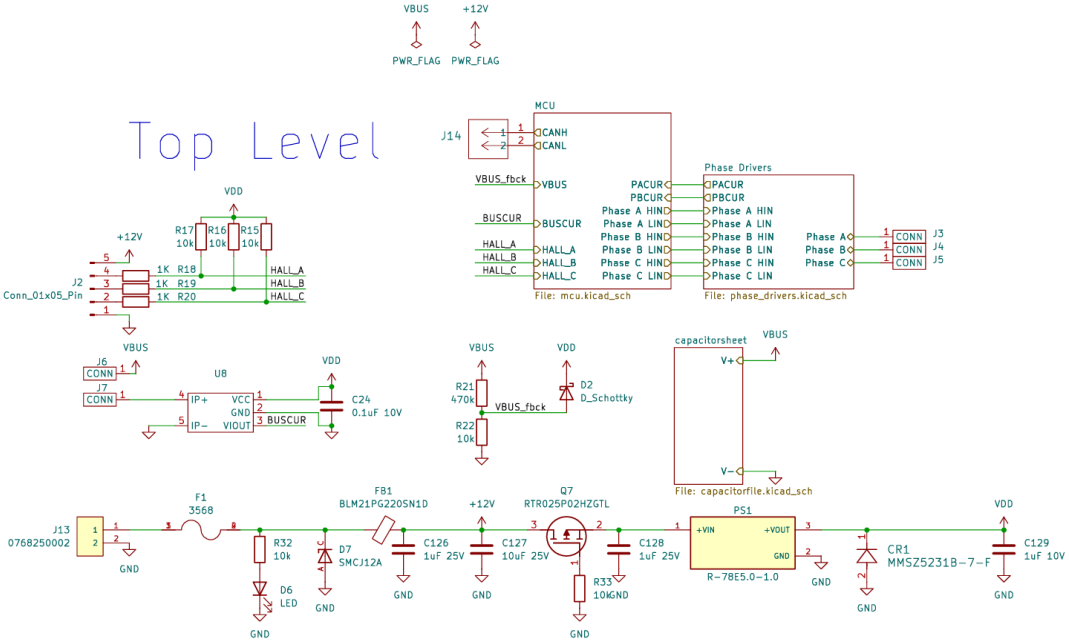

Figure 5. Rev 1 Top Level Schematic

The phase driver schematic, seen below, details the circuit that converts the drive signals from the microcontroller to high power phase actions from the triple-half-bridge. This circuit was based datasheet recommendations and existing products and verified through simulation.

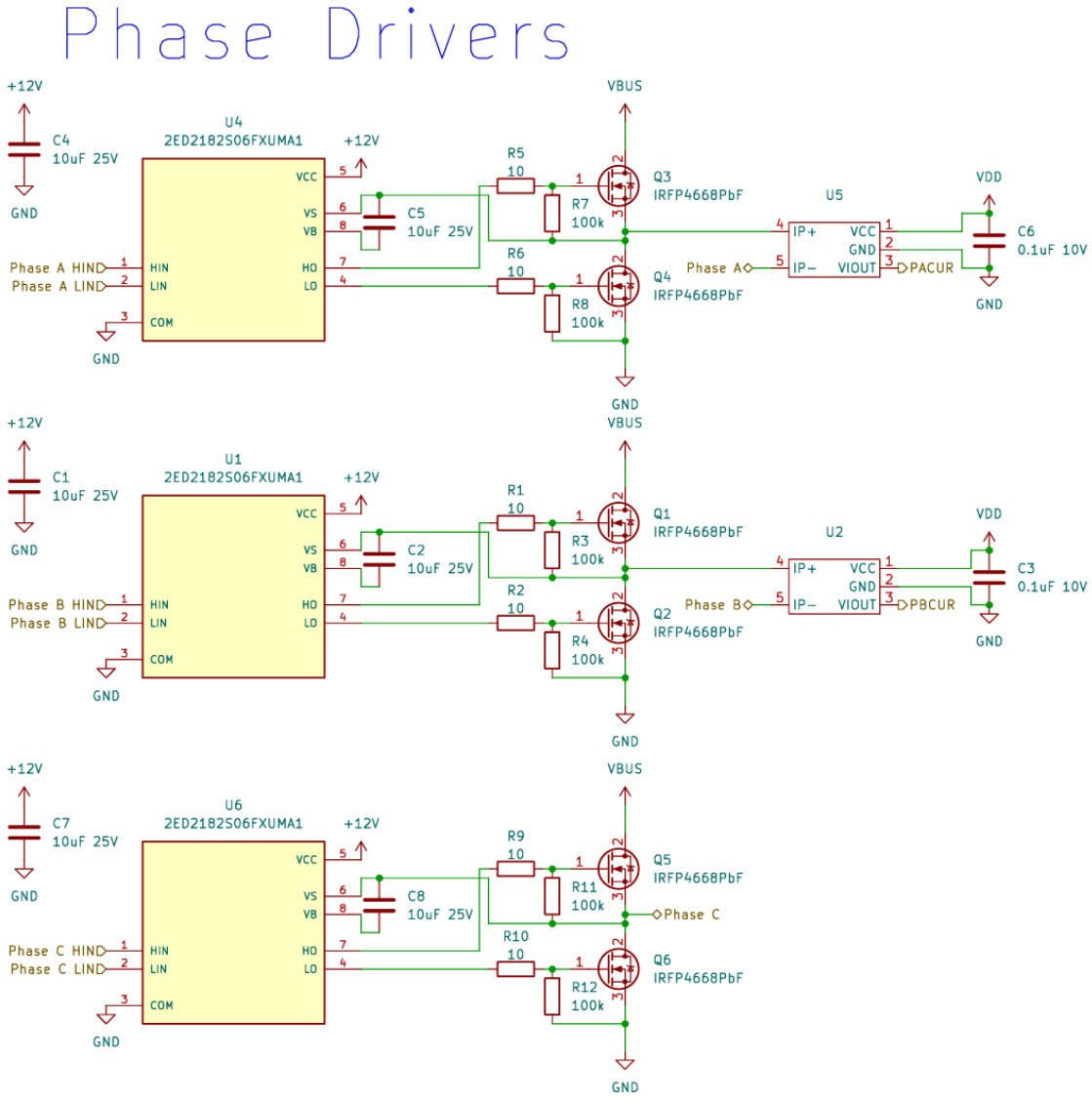

Figure 6. Rev 1 Phase Driver Schematic

The next section of the schematic is the microcontroller supporting circuitry. This encompasses the boilerplate circuitry needed for the microcontroller to run and be programmed as well as application specific circuitry. The application specific circuitry is the many fanned-out IO connections as well as debug connectors and CAN communication.

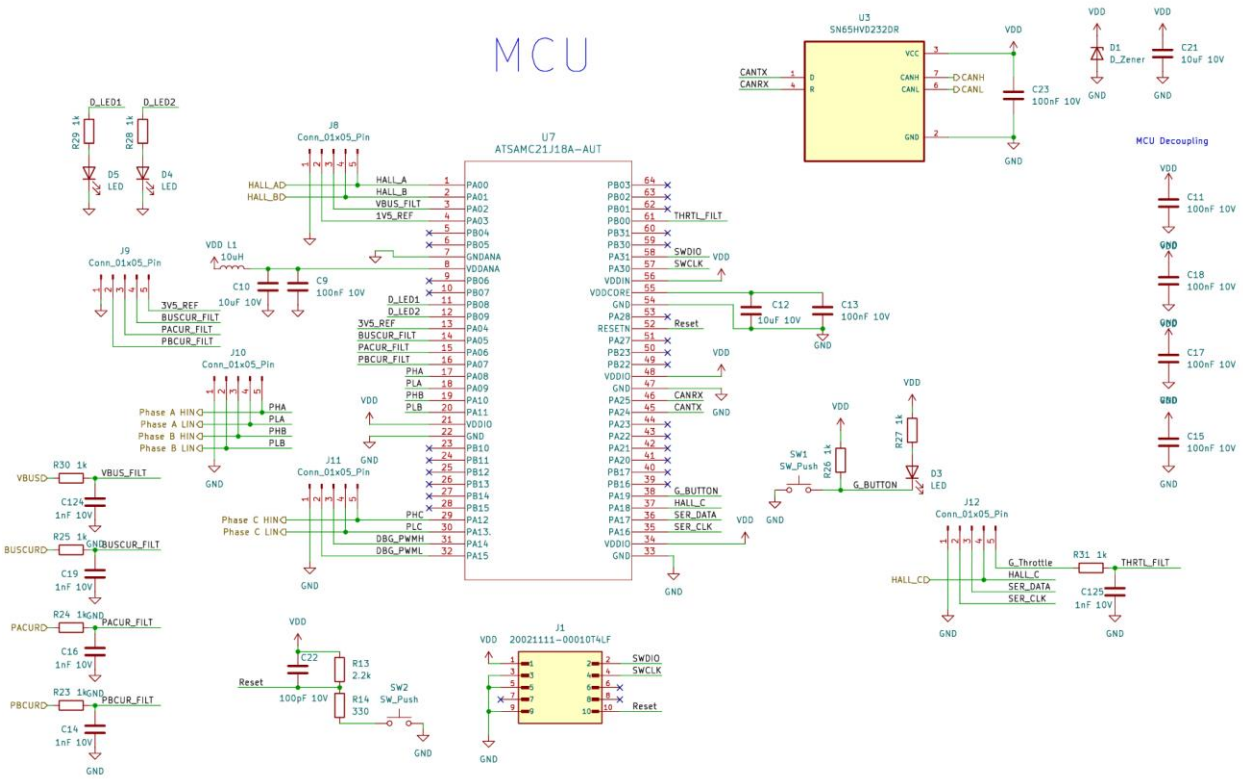

Figure 7. Rev 1 Microcontroller Schematic

The last schematic section (not pictured) simply contains the large MLCC capacitor bank used to stabilize the voltage input of the controller. This was separated as an organization optimization to make the schematic easier to read.

After this design came revision 1.1, which featured the exact same operation, just split into two boards. Below is the design of this revision

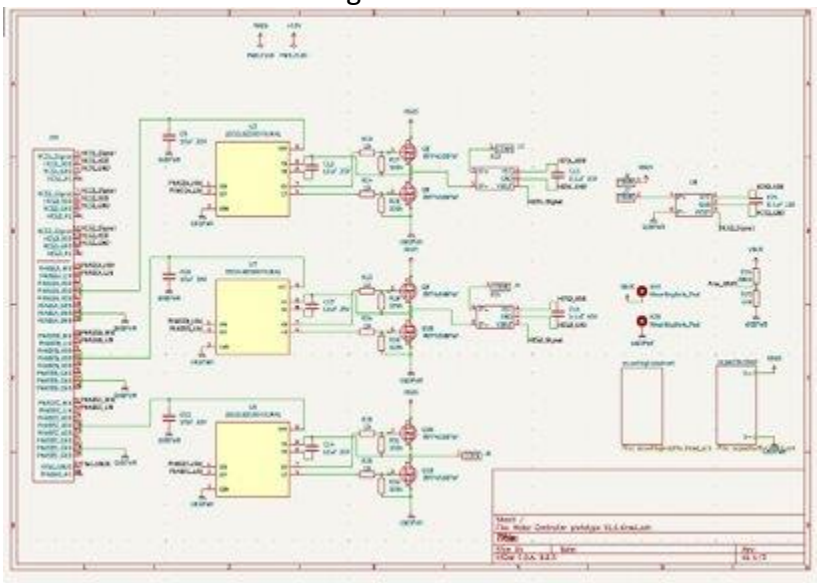

Figure 8. Revision 1.1 Power Board top level

Figure 8 shows the top level of the power board of this revision. This would have the gate drivers and the high voltage capacitor bank located on it, as well as signal traces leading to the signal board shown below.

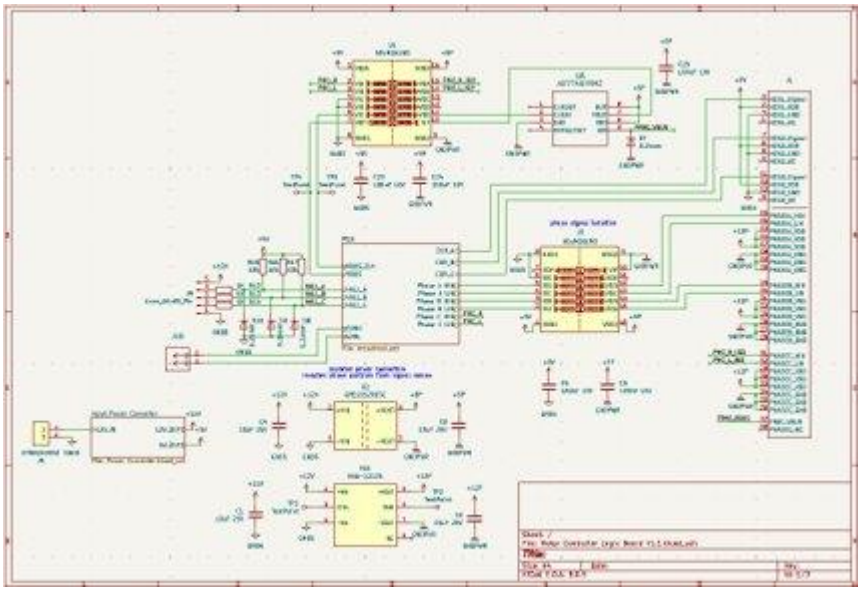

Figure 9. Signal Board Top level

Figure 9 shows the top level of the signal board, the board that would lie on top of the power board. It has the MCU and all associated debugging pins.

After being able to run some tests on revision 1.1, the second began. No changes were made to the phase drivers, but more debugging options were added to the MCU itself, and some of the pinout was changed to allow for better operation.

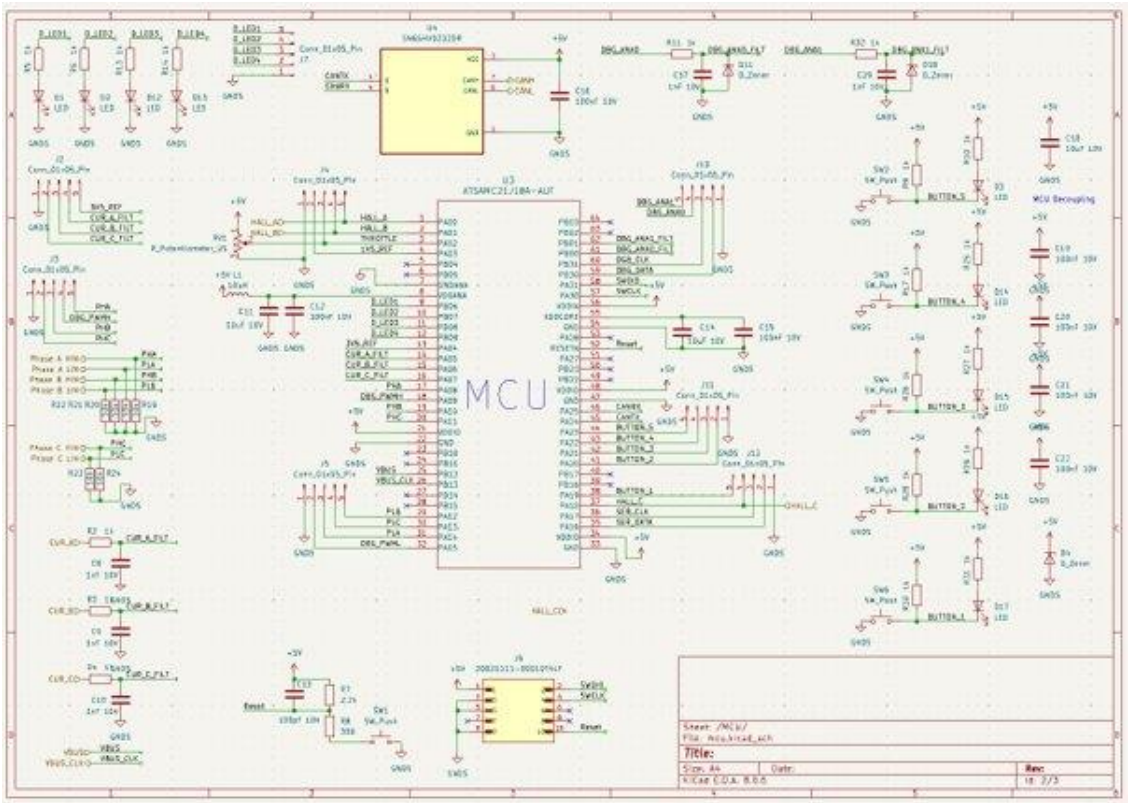

Figure 10. Power Converter

Figure 10 shows the second revision of our MCU schematic. As mentioned, the main changes were a large increase in debugging pinouts.

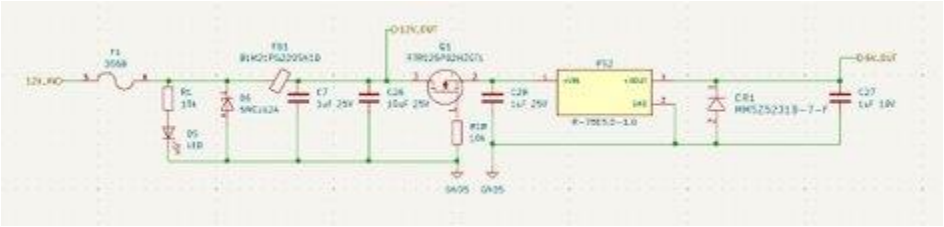

Figure 11. Power Converter

Figure 11 shows our power converter, which allows for 12 V logic for the gate drivers to be made from the 5 V logic output from the MCU.

7 ETHICS AND PROFESSIONAL RESPONSIBILITY

Ethics is always an important consideration when it comes to developing a project, and in the context of our project, that is no different. Engineering ethics, to us, refers to the application of moral principles to the decision-making processes in engineering, ensuring that our actions are both technically correct and socially and environmentally responsible. In our case, when designing our motor controller, we have to ensure that the user's safety is our highest priority and when adding in customizability, we must ensure that there are safe limits in place. Professional responsibility emphasizes the duty to uphold the trust placed in engineers to act in the best interest of society, ensuring that safety, fairness, and transparency are all prioritized. We carefully consider potential impacts, seeking to avoid harm or misuse of our project. Additionally, we ensure compliance with industry standards, legal frameworks, and regulatory guidelines throughout this project by being informed on the current requirements. We maintain an overarching commitment to honesty, accountability, and fairness; always striving to act in a manner that reflects positively on both our professional and personal integrity.

7.1 AREAS OF PROFESSIONAL RESPONSIBILITY/CODES OF ETHICS

Area of Responsibility	Definition	Relevant Item from Code of Ethics	Interaction/Adherence with this area in our project
Work Competence	Ensuring that the work we perform is of high quality, integrity, timeless, and within our professional expertise	“To avoid any conflict of interest and to avoid situations where financial interests or personal gain compromise judgement.” (IEEE Section 5)	We have previous experience working with the solar car’s previous motor controllers, and we will seek guidance when necessary to deliver quality work. We also regularly assess our progress and quality of work.
Financial Responsibility	Delivering products and services that offer value at a reasonable cost, without sacrificing quality	“To avoid any conflict of interest and to avoid situations where financial interests or personal gain compromise judgement.” (IEEE Section 6)	We have a goal of making this project as cheap as possible. This will be achieved by making the code open source, allowing for the only end cost being the components and board itself.
Communication Honesty	Reporting work truthfully, without deception, and	“To be honest and realistic in stating claims or	We ensure that all updates and deliverables are communicated truthfully to both each other and

	ensuring that the information is understandable to stakeholders.	estimates based on available data." (IEEE Section 2)	our other stakeholders. We also plan to develop clear documentation and transparency regarding the progress and challenges faced.
Health, Safety, Well-Being	Minimizing risks to the safety, health, and well-being of stakeholders	"To hold paramount the safety, health, and welfare of the public." (IEEE Section 1)	Our project has safety at the forefront of our decisions, to mitigate any potential injury to our users. This will be achieved with extensive testing and safety protocols within both hardware and software upon deployment.
Property Ownership	Respecting the intellectual property and confidential information of clients and others	"To respect the proprietary rights of others and avoid using any materials or information without permission" (IEEE Section 3)	We are careful to only take credit of work that is actually ours, while also properly crediting other sources and elements. Because our project will be open source, it will be available to all who want to use it, meaning that we won't have to limit who has access.
Sustainability	Protecting the environment and natural resources locally and globally.	"To contribute to the welfare of the environment and avoid actions that contribute to environmental damage."(IEEE Section 7)	Our project considers environmental factors by having its main application be focused on electric vehicles, promoting the use of more sustainable forms of energy.
Social Responsibility	Producing products and services that benefit society and communities.	"To strive to improve the understanding of technology, its use, and its potential impact on society." (IEEE Section 8)	We aim to ensure that our project will be to benefit of the broader community by addressing the need of a cheap motor controller that has a high amount of customizability for lightweight automotive use.

Our team has very good communication honesty. We are very transparent with our understandings and expectations of each other. We also are very quick to say when we may not know something or will have discussions when we disagree to arrive to a point that we all agree upon. This signifies strong performance because it means that we always keep a strong understanding of what position every other member is in. One area our team needs to improve is on the health, safety, and well-being ideal. Currently our team doesn't have well defined

testing procedures which could cause safety issues when testing. Along with this, we need to make sure that safety is paramount while writing our code and designing our board to make sure that our project is safe for others to use.

7.2 FOUR PRINCIPLES

	Beneficence	Non-malificence	Respect for autonomy	Justice
Public health, safety, and welfare	Enhancing public health outcomes through the design of safe products or services that benefit society.	Preventing harm by ensuring that the product is safe and does not cause injuries or negative health effects.	Ensuring that users are fully informed and can make their own choices regarding the use of the product.	Ensuring equitable access to the benefits of the product or service for all social groups.
Global, cultural and social	Promoting global health and wellbeing, respecting diverse cultures in design and approach.	Avoiding practices or products that may be harmful or culturally insensitive.	Respecting the rights of people from diverse cultural backgrounds to make informed decisions	Ensuring that the product or service is available to diverse social groups without bias.
Environmental	Designing the product in a way that contributes positively to environmental sustainability.	Minimizing harm to the environment, such as reducing waste or emissions.	Allowing consumers to choose more sustainable options and understand the environmental impact of their choices.	Ensuring that all communities, regardless of wealth or location, have access to our product.
Economic	Ensuring that the product contributes to economic growth or prosperity.	Preventing economic harm, such as ensuring that the product does not exploit workers or customers.	Allowing customers to make their own decisions without undue economic pressure.	Ensuring that the economic cost of the project is equal to all its potential users.

One broader context-principal pair important to our project is public health, safety, and welfare with nonmaleficence. This pair is important because a large portion of our time will be spent trying to improve the safety of our project for all its potential users. We plan to have multiple fail-safes built into our board design and within our software to address potential issues that may arise. One area we are currently lacking is in the Environmental and Respect for Autonomy pairs. Because we plan to have complete customization available to our users, it would be up to our users to adjust their settings to be better for the environment. Along with this, our use case is electric vehicles which are inherently better for the environment than other forms of transportation that may use fossil fuels.

7.3 VIRTUES

There are a few key virtues that are important to our team. The first is Commitment to quality. This is a key virtue to our team because we have a high self interest in our project. After all, it is planned to be used with the solar car, and our client is one of our group members. We plan to continue supporting this virtue through peer reviews and iterative testing. The second virtue vital to our team is Openness to Correction. Admitting mistakes and acknowledging oversight are essential for personal and professional growth. Our team has created an open environment where members can admit errors without fear of judgment. We believe acknowledging mistakes allows us to learn and grow as individuals and as a team. Handling these corrections constructively and focusing on solutions and improvements is key to our production as a team. The third virtue key to our team is a Habit of Documenting Work Thoroughly and Clearly. Clear and thorough documentation is one of the key deliverables for our project in the requirements, and we are constantly striving to produce. Along with this, making sure we have a thorough understanding of the work we have done allows us to have a better understanding of where we need to go in the future. We plan to instill a habit of documenting each aspect of our work so that we can create a referenceable history.

7.3.1 Individual Accounts

"One virtue I have demonstrated is seeing the "big picture" and the details of smaller domains. This is important to me because understanding whichever part of the project I am working on and its place in the whole is paramount to my success as a part of this team. I have demonstrated this by learning of FOC or Field Oriented Control as our key control mechanism for the motor. I need to learn about key parts like Space Vector PWM and the physics explanation of why there are multiple PID controllers were individual components that helped shape together to form my understanding of our eventual adaptation."

– Gavin Patel

"One virtue I feel I have shown throughout this project has been "openness to correction", whether it be from teammates or advisors. There is always room for improvement, and working with peers and educators gives many views beyond your own, and can often find flaws that you would not see on your own. Being open to any comments and suggestions others have is an easy way to have quick improvements to any designs or other work you may have. I do need to work on being more imaginative, as most of the work I do tends to be replicas of previous work by myself or others, or does not explore new paths that could be taken."

– Jonah Frosch

"I feel I have demonstrated the virtue of commitment to quality over the course of this project. The quality of my work weighs heavily on me. I prefer to go to sleep at night feeling like I have made something beautiful for people to use. Being on a group amplifies this since my group expect me to come through on my work and to finish it with quality. The software I've been working on has been planned and is primed for quality tests. I'm setting standards early and being consistent to make bullet-proof code that works efficiently. Perhaps all this quality could negatively impact my ability to meet timelines. Despite all this qualitative work, I have yet to work on demonstrating a habit of documenting work thoroughly. Other than a progress diagram, there is little of this code documented. Writing documentation is not something I shy away from; it tends to be something I hold to be very important in all my code. Reading undocumented code is something I dread, and I aim to not be the cause of this. As development progresses, there will be more code to document so I can begin to demonstrate this virtue."

– Bryce Rega

"*One virtue that I have not only demonstrated but strive to improve would be competence. This is important to me as competence not only allows me to work and learn effectively, but to analyze and continually improve me work and knowledge. Competence often relates to understanding the system as a whole and being able to give feedback on portions of a project that may not be your expertise. A virtue I need to improve on would be communicating clearly and informatively. This informative communication is important not only to ensure others outside our project group understand our presentation, but within our group itself. We had several miscommunications early-on stemming from assumptions that were made that hindered progress, and could have been cleared up much sooner with clear and informative communication."*

– Marek Jablonski

"*I have demonstrated the virtue of commitment to quality in this project in coordinating our team's meetings and deadlines. To ensure smooth team operations and every session was productive, I prioritized scheduling meetings that accommodated everyone's availability. Everyone was kept informed of upcoming milestones. This helped the team stay aligned. However, I realize that I can improve the habit of documenting work thoroughly, such as maintaining detailed records of meeting discussions and task assignments."*

– Long Yu

8 Conclusion

8.1 Summary of Progress

In the span of the first semester our team has been able to get an operational prototype and complete much of the design work for our first fully custom revision. This includes the hardware schematic and much of the layout as well as the software architecture.

Our major goals this first semester were to get an operational prototype and develop hardware and software such that we could assemble and test our first revision beginning immediately next semester. Next semester we are looking to assemble and test that first revision as well as iterate and produce a second revision that meets all requirements. As of now we are on track to achieve our goals if we are able to order and assemble the circuit board and develop prototype software over winter break.

Despite our current on track successes, we had a major setback that nearly prevented us from reaching these goals. At the beginning of the semester, we were not serious enough about our deadlines and did not appreciate the amount of effort these goals would take. This set us back many weeks. This is something that we have greatly improved on and must not fall into next semester to meet our goal.

Now at the conclusion of our second semester, we have a working second revision that is able to control the motor using the potentiometer located on the signal board for debugging as well as easy access buttons for starting and stopping the motor. We also have developed a GUI that can send and receive CAN messages from the motor detailing status and desired throttle input.

8.2 Value Provided

We were overall able to accomplish our goal of providing a much cheaper alternative to the current market of EV motor controllers while also not having them act as a black box to its users. There is still more work to be done for this project in order for it to be seamlessly transitioned into the solar car, but our progress has enabled us to be able to provide a very good jumping off point for whoever may continue this project.

8.3 Next Steps

There is still much more that could be done for this project that we were unable to accomplish

8.3.1 Hardware Next Steps

More load testing should be done on the board, along with analysis on different types of power supplies performance on our board. Due to our time limitations, no test dyno was ever used in our testing so we don't have accurate information on how the controller performs under a load greater than just the motor itself. We also mainly only used power supplies during our testing, so seeing how the performance of the board changes when hooked up to a battery is also important future work.

8.3.2 Software Next Steps

There is still more work that needs to be done on the software side of things as well. Currently, we do not have any working current control, which is the main way that the solar car

currently works to control its motors. Connecting an existing PID controller and tuning it for the measured current would need to be done. We also plan to move away from using 6-step waveform generation for control, and instead, would plan to have a control methodology that better simulates a sinusoidal waveform by utilizing field-oriented control. Finally, we would also like to more fully flesh out our GUI to allow users to have more features than just adjusting the throttle control of the motor itself.

9 REFERENCES

[1] "Controlled Regenerative Braking using Real Time Speed Sensing," *RoboteQ*, [Online]. Available: https://www.roboteq.com/applications/all-blogs/24-controlled-regenerative-braking-using-real-time-speed-sensing [Accessed Nov. 10, 2024].

[2] "Field-Oriented Control," *MathWorks,* [Online]. *www.mathworks.com*. https://www.mathworks.com/discovery/field-oriented-control.html [Accessed Oct. 30, 2024].

[3] "STM32MotorControl: Introduction to Motor Control with STM32 stm32mcu," *Stmicroelectronics.cn*, 2024 [Online]. Available: https://wiki.stmicroelectronics.cn/stm32mcu/wiki/STM32MotorControl:Introduction_to_Motor_Control_with_STM32 [Accessed Oct. 08, 2024].

[4] "WAVESCULPTOR22 MOTOR CONTROLLER," *Prohelion,* [Online]. Available: https://prohelion.com/shop/wavesculptor-motor-controllers/wavesculptor22-motor-controller/ [Accessed Nov. 07, 2024].

[5] "Solar Car Products, Micro Mobility Products," *MITSUBA,* [Online]. Available: https://www.mitsuba.co.jp/en/vpep/products/solar_car_products.html [Accessed Aug. 17, 2024].

[6] "48-72V Brushless Motor Speed Controller DC 1500W Scooter Controller Replacement for E-Bike Scooter Motor Controller," *Amazon,* [Online]. Available: https://www.amazon.com/48-72V-Brushless-Controller-Scooter-Replacement/dp/B0BTYJK2J8/ref=asc_df_B0BTYJK2J8 [Accessed Nov. 07, 2024].

[7] "Beyond PID: Exploring Alternative Control Strategies for Field-Oriented Controllers," *MathWorks,* [Online]. Available: https://www.mathworks.com/campaigns/offers/next/field-oriented-control-techniques-white-paper.html [Accessed Oct. 30, 2024].

[8] "Understanding BLDC Motor Control Algorithms," *MathWorks,* [Online]. Available: https://www.mathworks.com/campaigns/offers/next/understanding-bldc-motor-control-algorithms.html [Accessed Oct. 30, 2024].

[9] "Field-Weakening Control," *MathWorks,* [Online]. Available: https://www.mathworks.com/discovery/field-weakening-control.html [Accessed Oct. 30, 2024].

[10] F. Niessen, "Winding Scheme Calculator," *Homebuilt Electric Motors,* [Online]. Available: https://www.bavaria-direct.co.za/scheme/calculator/ [Accessed Dec. 07, 2024].

APPENDIX

Empathy Map

Hears	Sees
• Does it work with my battery pack? • We need this to work all the time • Applause when your vehicle is successfully powered by the motor controller	• The wheel(s) spinning • Readable documentation • Easy to configure hardware interface
Says and Does	**Thinks and Feels**
• Compatible with my different scenarios • Easy to use and configure! • Connects the motor to other systems	• Pride that the project works • Not frustrated with configuration Don t need to worry about the product breaking their hardware

Figure 8. Empathy Map

AT SAM C21 J18	Primary		Secondary		Tertiary	
Function	Port & Pin	Peripheral	Port & Pin	Peripheral	Port & Pin	Peripheral
Phase H	PA08	TCC0 [0]	PA16	TCC0/2	PB30	TCC1 [2]
Phase L	PA09	TCC0 [1]	PA17	TCC0/2	PB31	TCC1 [3]
Phase H	PA10	TCC1 [0]	PA30	TCC1 [0]	PB30	TCC1 [2]
Phase L	PA11	TCC1 [1]	PA31	TCC1 [1]	PB31	TCC1 [3]
Phase H	PA12	TCC2 [0]	PA16	TCC0/2	PB30	TCC1 [2]
Phase L	PA13	TCC2 [1]	PA17	TCC0/2	PB31	TCC1 [3]
Bus Voltage	PA02	AIN2	PB00	AIN0		
Phase Current	PA05	AIN5	PB01	AIN1		
Phase Current	PA06	AIN6	PB02	AIN2		
Phase Current	PA07	AIN7	PB03	AIN3		
1.5V Reference	PA03	VREFA				
3.5V Reference	PA04	VREFB				
HALL Sensor	PA00	EXTINT[0]	PA14	EXTINT[14]		
HALL Sensor	PA01	EXTINT[1]	PA15	EXTINT[15]		
HALL Sensor	PA18	EXTINT[2]	PA02	EXTINT[2]		
CAN Tx	PA24	CAN0 TX	PB22	CAN0 TX	PB14	CAN1 TX
CAN Rx	PA25	CAN0 RX	PB23	CAN0 RX	PB15	CAN1 RX
DAC Analog Out	DNC		PA02	VOUT		
Sercom Data	PA16	SERCOM 1/3	PA22	SERCOM 3/5		
Sercom Clock	PA17	SERCOM 1/3	PA23	SERCOM 3/5		
Throttle ADC	PB00	AIN0	PA08	AIN8	PB08	AIN2/4
Timer or PWM H	PA14	TC4 [0]	PA18	TC4 [0]	PB08	TC0 [0]
Timer or PWM L	PA15	TC4 [1]	PA19	TC4 [1]	PB09	TC0 [1]
Extra Button	PA19	EXTINT[3]	PA14	EXTINT[14]	PA15	EXTINT[15]
Extra LED	PA14	EXTINT[14]	PA15	EXTINT[15]	PA19	EXTINT[3]
Extra LED	PA15	EXTINT[15]	PA14	EXTINT[14]	PA19	EXTINT[3]
Extra GPIO	PA22	EXTINT[6]	PB22	EXTINT[6]	PB12	EXTINT[12]
Extra GPIO	PA23	EXTINT[7]	PB23	EXTINT[7]	PB13	EXTINT[13]

Figure 12. SAMC21 Full Pinout

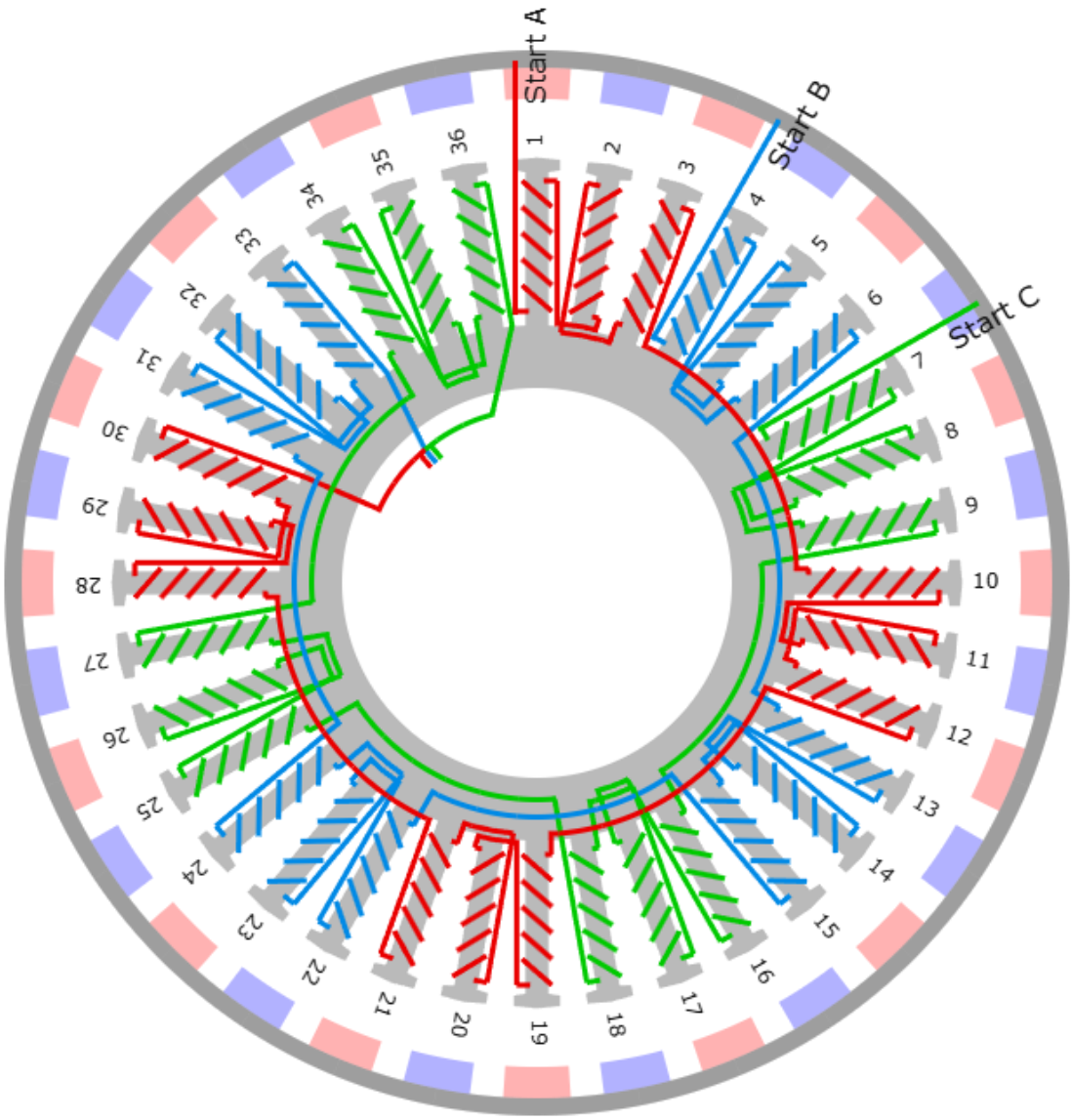

Figure 13. Visualized Motor Winding Diagram [10]

9 TEAM

9.1 TEAM MEMBERS

- Marek Jablonski
- Bryce Rega
- Jonah Frosch
- Gavin Patel
- Long Yu

9.2 REQUIRED SKILL SETS FOR YOUR PROJECT

- LTSPICE Circuit Simulation
- Power Systems Circuit Development
- PCB Design
- Embedded Software
- Soldering
- Feedback Control Systems
- C (Programming Language)

9.3 SKILL SETS COVERED BY THE TEAM

- LTSPICE Circuit Simulation – Marek Jablonski
- Power Systems Circuit Development – Marek Jablonski, Jonah Frosch
- PCB Design – Jonah Frosch, Marek Jablonski
- Embedded Software – Bryce Rega, Gavin Patel
- Soldering – Jonah Frosh, Marek Jablonski, Long Yu
- Feedback Control Systems – Gavin Patel
- C (Programming Language) – Bryce Rega, Gavin Patel

9.4 PROJECT MANAGEMENT STYLE ADOPTED BY THE TEAM

The team will meet with advisors on a weekly basis to keep them in the loop and seek advice on progress, as well as a minimum of 1 weekly meeting time to focus solely on project work as a group. On a work basis:

- Software Side: Agile method
    - Allows for rapid iterations
    - Can test without a completed project
- Hardware side: Agile-Waterfall Hybrid method
    - Agile allows for iteration feedback management to take place
    - Within each iteration waterfall is used to keep a logical flow of work from concept to ordered product

9.5 INITIAL PROJECT MANAGEMENT ROLES

- Marek Jablonski – Testing & Systems Compliance
- Bryce Rega – Software Design & Adherence
- Jonah Frosch – Hardware Design
- Gavin Patel – Docs & software writing
- Long Yu – Team Organization

9.6 TEAM CONTRACT

Team Name: sdmay25-26

Team Members:
1) Bryce Rega 2) Jonah Frosch
3) Gavin Patel 4) Marek Jablonski
5) Long Yu

Team Procedures

1. Day, time, and location (face-to-face or virtual) for regular team meetings:
    a. Face-to-face meeting weekly in the TLA, 1:30pm Thursdays. Additional Meetings or workdays scheduled as needed.
2. Preferred method of communication updates, reminders, issues, and scheduling (e.g., e-mail, phone, app, face-to-face):
    a. Text messaging for communication updates and reminders, as well as discussions during weekly in person meetings.
3. Decision-making policy (e.g., consensus, majority vote):
    a. Attempt for a consensus, 4 out of 5 members need to approve a decision.
4. Procedures for record keeping (i.e., who will keep meeting minutes, how will minutes be shared/archived):
    a. Records will be kept in a meeting records folder in CyBox
    b. Record keeper will be Gavin Patel

Participation Expectations

1. Expected individual attendance, punctuality, and participation at all team meetings:
    a. Attend all meetings, communicate reasonably beforehand (24 hours, 30 minutes for unplanned interruptions) if unable to attend or in Detroit.
    b. Be on time (within 5 mins of nominal start time).
        i. Communicate any late arrivals before nominal start time
    c. Participate in meeting discussions and work.
2. Expected level of responsibility for fulfilling team assignments, timelines, and deadlines:
    a. Each member should have a significant contribution effort towards team assignments, and should adhere to timelines and deadlines as much as possible.
3. Expected level of communication with other team members:
    a. Each member should respond to messages within 12 business hours and should be involved in conversations related to what they are working on.
4. Expected level of commitment to team decisions and tasks:
    a. We expect each member to come to our meetings focused and committed to getting work done. Each member should be actively involved in the decision-making process to ensure a successful outcome.

Leadership

1. Leadership roles for each team member (e.g., team organization, client interaction, individual component design, testing, etc.):
    a. Team Organization - Meeting Scheduling, Deadline Coordination
        i. Long Yu
    b. Software Design & Adherence – Ensuring software stays organized and modular
        i. Bryce Rega
    c. Hardware – Designing Boards and Selecting proper components
        i. Jonah Frosch
    d. Testing and Systems Compliance
        i. Marek Jablonski
    e. Documentation and software writing
        i. Gavin Patel
2. Strategies for supporting and guiding the work of all team members:
    a. This team will largely be split into electrical and software so group members can seek support from those who are in their respective groups. Guiding of work can also be decided in large group meetings and then split among sub-teams.
3. Strategies for recognizing the contributions of all team members:
    a. Recognizing the contributions of team members can be recognized upon successful completion of their task(s) by verbal praise/recognition on reports.

Collaboration and Inclusion

1. Describe the skills, expertise, and unique perspectives each team member brings to the team.
    a. Long Yu – Knows about electricity
    b. Marek Jablonski – Experience with the primary motor we are using and motor control operation and debugging
    c. Jonah Frosch – Knows about motors & electricity
    d. Bryce Rega – Experience with embedded software and firmware as well as specifically using STMicroelectronics and their software (dev-board)
    e. Gavin Patel – Experience with embedded software and firmware
2. Strategies for encouraging and supporting contributions and ideas from all team members:
    a. Keeping a positive team atmosphere by using positive language and an encouraging and open environment
3. Procedures for identifying and resolving collaboration or inclusion issues
    a. Part of keeping an encouraging and open environment is allowing for people to voice their opinions on the current direction and atmosphere of the team. Making the communication clear and directed towards the betterment of the group is key to these sorts of conversations.

Goal-Setting, Planning, and Execution

1. Team goals for this semester:
    a. Design the first draft of the cost effective and configurable motor controller
        i. Major component operation simulated successfully

   b. Successfully test the Development Board & Software with the motor.
      i. “Successfully test” defined as able to control speed and direction of the motor on command.
2. Strategies for planning and assigning individual and team work:
   a. Work will be separated based on interest and work type based on people's experience.
3. Strategies for keeping on task: Plan ahead, attend every meeting, and keep the record
   a. We will create a tentative calendar to try and keep to it as much as possible. Every meeting attendance is expected unless otherwise stated. Keeping records will be done by Gavin Patel for each meeting and technical documents will be worked on by those who worked on the components as required

Consequences for Not Adhering to Team Contract

1. How will you handle infractions of any of the obligations of this team contract??
   a. On the first and second instances of an issue the team will bring it up and attempt to correct future action
   b. In the case of a severe infraction (conduct detrimental to the work of others or the project as a whole) the team member will be temporarily excluded from team work until a resolution is decided with the course organizers.
2. What will your team do if the infractions continue?
   a. On further instances of an issue the team will reach out to the course organizers for advice on how to continue.

****

a) *I participated in formulating the standards, roles, and procedures as stated in this contract.*
b) *I understand that I am obligated to abide by these terms and conditions.*
c) *I understand that if I do not abide by these terms and conditions, I will suffer the consequences as stated in this contract.*

1) *Bryce Rega* _____ DATE 9/12/2024
2) *Jonah Frosch* _____ DATE 9/12/2024
3) *Gavin Patel* _____ DATE 9/12/2024
4) *Marek Jablonski* _____ DATE 9/12/2024
5) *Long Yu* _____ DATE 9/12/2024