University of Manitoba
Department of Electrical & Computer Engineering

ECE 4600 Group Design Project

Final Project Report

DESIGN OF A MOTOR CONTROLLER FOR A 60 KW TRACTIVE MOTOR

by
Group 13

Troy Eskilson
Jolene Kozak
Kolby Tober
Ian Sweetland

Academic Supervisor
Dr. Carl Ho

Date of Submission
March 6, 2020

Abstract

Every year, the University of Manitoba's Formula Electric team designs and builds an electric race car to compete in Formula SAE Electric design competitions overseen by SAE International. In the hopes of achieving first place, the team works to improve the car's systems wherever possible. The car's tractive system currently employs an outsourced motor controller that outputs three-phase power to drive an axial flux permanent magnet motor. Besides being expensive, an off the shelf motor controller is not an optimal solution as it has not been specifically designed for the team's car and needs. Available options in the market often do not offer ideal layouts for fitting tightly in a vehicle's envelope. Off the shelf solutions also place arbitrary restrictions on motor selection due to available voltage, current, and power ratings. Further, good vehicle design often necessitates reducing mass, and the mass of an outsourced product is difficult to reduce. This report summarizes the development of a motor controller prototype that will lead to a fully functional model in future iterations.

This prototype only entailed the design of electrical elements needed for the function of the motor controller, and addressed mechanical aspects only when necessary for testing. The designed motor controller features a three-phase half-bridge inverter module and a field-oriented torque-controlled system, contains on-board fault protection, temperature limiting controls, data-logging functionality, and a cross-platform graphical user interface. The team successfully developed and tested a working motor controller, with testing completed for all seven metrics, and five metrics were exceeded.

Contributions

	Troy Eskilson	Jolene Kozak	Kolby Tober	Ian Sweetland
Power Electronics	●		○	
Controls	●			
Embedded Software		●		○
PCB Design	●	○	○	
PCB Manufacturing & Testing		●		
Graphical User Interface				●
Report Editing			●	

Legend: ● Lead task ○ Contributed

Acknowledgements

The team would like to acknowledge:

- Dr. Carl Ho, for his generous financial and technical support.
- The University of Manitoba IEEE Student Branch (UMIEEE), for allowing us to use the McNaughton Learning Resource Centre as a workspace and giving us access to tools, soldering equipment, measurement devices, and consumables.
- The University of Manitoba Student Chapter of SAE International (UMSAE) Formula Electric Team, for allowing us to use their tools and equipment, and providing the motor (along with other necessary hardware).
- Cory Smit, for providing his machining services.
- Michael Rempel Boschman, for designing critical mechanical components when they were required.
- Quinn Sabin, for responding to technical questions related to the project.

UMSAE
Polar Bear Racing Electric

Table of Contents

List of Figures

List of Tables

Nomenclature

ω	Angular frequency
ω_e	Electrical angular frequency of motor
ψ_m	Magnetic flux of motor
BW	Bandwidth of a controller
C	Capacitance
E	Energy
f	Frequency
f_{sw}	Switching frequency of the motor controller
I	Current
I_d	D-axis current in dq-reference frame
I_q	Q-axis current in dq-reference frame
k_i	Integral gain of a PI controller
k_p	Proportional gain of a PI controller
k_{aw}	Anti-windup gain of a PI controller
L	Inductance
L_d	D-axis motor inductance
L_q	q-axis motor inductance
P	Power
pp	Number of pole pairs of motor

Q_{gate}	Gate charge
R	Resistance
R_s	Stator resistance of motor
t	Time
V	Voltage
V_d	D-axis voltage in dq-reference frame
V_q	Q-axis voltage in dq-reference frame
Z	Impedance

Glossary and Acronyms

AC alternating current.

Accumulator This is the lithium-ion battery pack of the Formula Electric vehicle. It contains all of the electrical energy used by the Tractive System.

Back Electromotive Force (Back EMF) This is the electromotive force, or voltage, generated in a motor's coils.

Cascadia Motion PM100DX The PM100DX is the motor controller currently used by UMSAE Formula Electric.

Control PCB The designed motor controller contains main two PCBs. In the context of this report, the control PCB is the PCB mounted on top of the gate driver PCB, and contains the microcontroller and most of its direct interfaces.

Controller Area Network This is a vehicle communication bus standard. Electrical devices use a CAN bus to communicate with each other. It is used on the Formula Electric vehicle.

Data-Driven Documents Library (D3.js) Data-Driven Documents is a JavaScript library for developing dynamic data visualizations.

DC direct current.

Direct Torque Control (DTC) Direct torque control. This is a method of controlling three-phase motors based on calculating an estimate of the motor's magnetic flux and torque. Hysteresis controllers are used to regulate the torque and flux of the motor.

DOM The Document Object Model is a logical tree representing the structure of a web page.

Electromagnetic Interference (EMI) EMI is a significant source of noise in motor controllers, due to the fast switching of voltages by the IGBTs.

EMRAX 188, 208, and 228 motors These are a series of permanent magnet synchronous motors, sold by the company EMRAX. These are quite popular in the FSAE competitions, due to their price and performance. Popular models used in the competition are the 188, 208, and 228 motors, listed in order of ascending physical size. Additionally, these motors are available at three different voltages levels, designated LV (low voltage), MV (medium voltage), and HV (high voltage, not to be confused with the FSAE definition of HV).

Equivalent Series Inductance (ESL) Equivalent series inductance. Capacitors and batteries are not ideal components. An improved approximation includes an effective resistive and inductive component. The inductive portion of this circuit is referred to as an equivalent series inductance.

Equivalent Series Resistance (ESR) Equivalent series resistance. Capacitors and batteries are not ideal components. In reality, a capacitor is not only a capacitance, but a better approximation is a series RLC circuit. The resistive portion of this circuit is referred to as an equivalent series resistance.

Failure Mode and Effects Analysis (FMEA) This is a structured approach to identifying potential failures within a design. The cause of the failures, and the effects of such failures, are analysed.

Field Oriented Control (FOC) Field oriented control. This is a method of controlling three-phase motors based on the magnetic angle of the motor, and utilizing the park transform. PI controllers are used to regulate the currents and voltages applied to the motor.

Formula Electric The Formula Electric team is part of the University of Manitoba Student Chapter of SAE International. They design and manufacture a small Formula-style race car powered by an electric power source and compete against other teams from Universities across the world in the FSAE Electric competitions.

Formula SAE (FSAE) An engineering design competition for undergraduate and graduate students. The competition provides an opportunity for students to enhance their engineering and project management skills.

Gate Driver PCB The designed motor controller contains two main PCBs. In the context of this report,

the gate driver PCB is the PCB mounted directly to the IGBT module, and contains the gate driving circuitry along with the required measurement circuitry.

Grounded Low Voltage (GLV) According to the FSAE 2020 rules, this is defined as every electrical part that is not part of the tractive system.

GUI graphical user interface.

High Voltage (HV) As defined in the FSAE 2020 rules, any circuit with a potential difference where the nominal operation voltage is greater than 60 V DC or 25 V AC RMS.

IGBT insulated gate bipolar junction transistor.

Inverter This refers to any device which converts DC electrical power into AC electrical power.

MOSFET metal oxide semiconductor field effect transistor.

Motor Controller In this report, includes the inverter as well as all the components of the control system used to convert battery energy to usable tractive system power.

Negative Temperature Coefficient Thermistor (NTC) This is a resistor whose resistance varies with temperature. As it heats up, the resistance decrease. This is commonly used to measure temperature.

PCB printed circuit board.

Permanent Magnet Synchronous Motor (PMSM) This is a family of high efficiency and high power density electric motors. These properties make it particularly well suited to electric vehicles.

Powertrain The powertrain of a vehicle is the main mechanical components that generate and transmit power to the road surface.

React A JavaScript library used to build user-interfaces for web-applications.

Resolver A rotary electrical transformer coupled to the motor shaft and used to determine the mechanical angle of the motor.

RPM Revolutions per minute. Used as a common measurement of motor speeds.

Shoot-through In an inverter, a shoot-through fault when the high side and low side switches are closed at the same providing a low resistance path from the high voltage anode to ground..

Space-vector Pulse Width Modulation (SVPWM) Space vector pulse width modulation. This is a common PWM method used to approximate a sinusoid with a digital signal.

The Electron Framework An open-source framework used for developing desktop applications using web-development technologies.

Tractive System (TS) According to the Formula SAE 2020 rules, this is defined as every part that is electrically connected to the motor and accumulator.

TypeScript An open-source programming language and superset of JavaScript that adds static typing.

University of Manitoba Student Chapter of SAE International (UMSAE) An umbrella non profit student organization overseeing a collection of design teams at the University of Manitoba, including the Formula Electric team.

Chapter 1

Introduction

1.1 Overview

The intent of this project was to design and manufacture a first prototype of a motor controller (sometimes referred to as an inverter), for the University of Manitoba Student Chapter of SAE International (UMSAE) Formula Electric team competing in the Formula SAE (FSAE) design competition. The scope of the project included all the electrical design, but excluded certain mechanical aspects of design—specifically the enclosure, the cooling plate, and testing apparatus. In the context of the Formula Electric team, the device converts DC power from the accumulator into three-phase AC power, which is controlled to drive an EMRAX 188 permanent magnet synchronous motor (PMSM). The control system receives torque commands over a controller area network (CAN) channel from a pedal position sensor. A bulk capacitance (also refered to as DC link capacitance), connected across the input terminals to the IGBT module, is used to overcome the accumulator's large series inductance.

This would be replacing an existing off the shelf motor controller. In order for such a design to be accepted by the team, the specifications must be similar.

This motor controller has been designed with the EMRAX 188 high voltage (HV) motor in mind. The UMSAE team is interested in moving to dual motors and this is a promising model. As UMSAE does not currently possess this motor, this prototype had use a spare EMRAX 228 MV motor for testing instead. The relevant ratings are in table 1-1 below, and an image in figure 1-1. The 228 motor possesses significantly higher ratings in most categories, making it an ideal candidate to test the motor controller with, as the motor's ratings would not be pushed if testing at a level suitable to the 188.

Table 1-1: Datasheet values of the EMRAX motors selected, taken from [6]

Parameter	188 HV Rating	228 MV Rating
Peak power	60 kW	100 kW
Continuous power	32 kW	55 kW
Max DC voltage	400 VDC	470 VDC
Peak motor current	200 Arms	340 Arms
Continuous motor current	100 Arms	160 Arms
Max motor RPM	7600 RPM	6500 RPM
Peak motor torque	90 Nm	240 Nm
Weight	7.0 kg	12.3 kg
Pole pairs	10	10

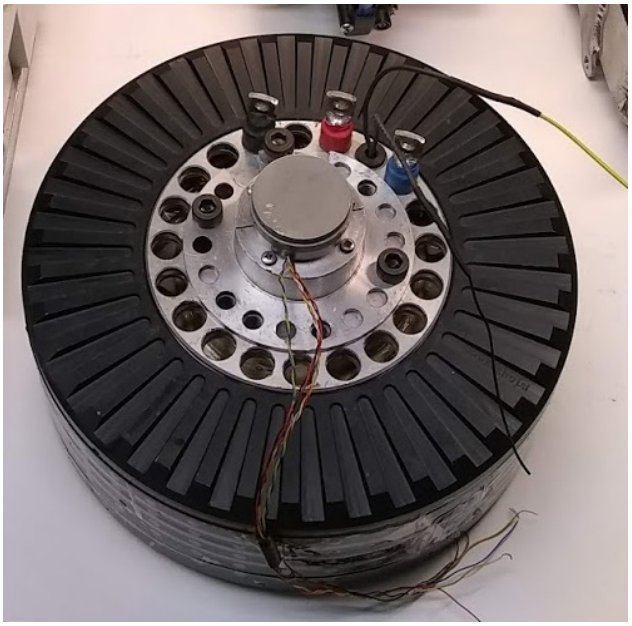


Figure 1-1: Image of EMRAX 228 Motor

1.2 Motivation

There are several major reasons why the Formula Electric team is interested in a team-designed motor controllers:

- Cost savings
- Increased team knowledge of motor controllers
- Additional flexibility of a custom motor controller
- Member learning experience

The team is interested in moving from a single motor powertrain, to a dual motor powertrain in the near future. This would require the purchase of two new motors, costing around $5000 each, and a new motor controller, costing around $9000. The team is unlikely to have the budget to support both purchases in a single year. However, after investigating the costs of material and manufacturing to make a motor controller, the electrical hardware is estimated to only cost $1600. This lower total is feasible for the team.

In the past, the team has had a lack of understanding of motor controllers and, as a result, struggled to debug their off-the-shelf solutions. Successfully designing a motor controller requires an understanding of both the theory and the implementation. Having such knowledge within the team would be a major asset in getting the vehicle running quickly after assembly. In the past, the mechanical assembly has finished around one month before the FSAE competition, and debugging the motor controller can take several weeks of this valuable time. Having an understanding of the operation of these devices would have a significant and noticeable impact on the team's performance, and allow the team to make informed future purchasing decisions.

In motorsports, weight is crucial and every gram matters. This places a great importance on volume and packaging. There is very little freedom of placing an outsourced motor controller in a race car and often the optimal placement of the component cannot be achieved. A custom controller can be designed to fit into optimal and otherwise unused locations. Additionally, having access to the entire source code allows for quick changes to the control system. If, for example, the output phase rotation of the motor controller was different than that of the motor then a single line of code may be changed instead of having to re-

wire the motor itself. Further, having access to all variables is invaluable during debugging with a custom controller. This motor controller will also serve as a platform for testing future control algorithms such as field weakening control, sensorless control, and different thermal limiting algorithms.

Ultimately, FSAE's goal is to provide students with a learning experience. As one of the few remaining purchased electrical components, a custom motor controller offers new design opportunities in power electronics, PCB design, and control system design.

1.3 Requirements

A series of requirements must be met for UMSAE to consider using a custom motor controller over a commercial product. Such a device must be (1) fully FSAE rules compliant, (2) suited to driving the EMRAX 188 motor, and (3) its control characteristics must be comparable or exceed the current motor controller in use: the Cascadia Motion PM100DX. Additionally, there are three conditions requested by the UMSAE Formula Electric that the motor controller must satisfy. A detailed list of these requirements can be found below. These requirements and how they affected design decisions will be explored further in the relevant sections.

- Specifications dictated by FSAE:
    - Design must be fully FSAE 2020 rules compliant (see [16] for the full set of rules).
        - The Tractive System (TS) and grounded low voltage (GLV) systems must be galvanically isolated from each other.
- Specifications dictated by EMRAX 188 motor (see [6] for the datasheet):
    - $>$400 VDC operation
    - $>$200 Arms peak
    - $>$100 Arms continuous
    - $>$100 Hz maximum output electrical frequency
- Specifications dictated by PM100DX (see [20] for the datasheet):
    - Torque controlled motor

  - Field oriented control (FOC) system
  - Space-vector pulse width modulation (SVPWM)
  - Switching frequency at least 12 kHz
  - Adjustable switching frequency
  - Temperature monitoring of IGBT and motor
  - Switching fault protection
  - Useful logging and graphical user interface (GUI) for PC
    - $>$ 3 Hz live logging capability
    - $>$2000 sample buffer for detailed, triggered logging.
    - Ability to change control system gains over interface.
  - Resolver feedback for position measurement.

- Specifications dictated by UMSAE usability.
  - Controlled over CAN
  - Resolver calibration capability
  - GUI with cross-platform support

1.4 Metrics

The requirements found in section 1.3 drove the selection of the project metrics. This project is intended to be a first prototype, and the metrics were also limited by the available testing equipment. Specifically, the HV power supply is limited to 300 V and 1.5 kW and the EMRAX 228 motor is rated for a lower speed, at a higher voltage.

Table 1-2: Target performance metrics of this project.

Performance Metric	**Target**
Efficiency	90% at 1 kW
Output Current	200 A
User Interface	$<$ 5 minutes to configure motor controller
Switching Frequency	$>$ 12 kHz
Max Output Mechanical Frequency	$>$ 50 Hz (3000 RPM)
Inner Control Loop Frequency	$>$ 800 Hz
Fault Shutoff Time	$<$ 5 ms

The maximum output frequency has changed from 5000 RPM, indicated in the original proposal, to 3000 RPM. Originally, we were planning to test with a 400 VDC source, but we were only able to test with a 240 VDC source. The original metrics table can be found in appendix section A.1.

The efficiency metric is based on the required efficiency to be considered feasible for use by the Formula Electric team. Information about efficiencies of the off the shelf controllers is unavailable. The output current is based on the current requirements of the EMRAX 188 motor, which is 200 A per phase. The user interface switching time was based on the amount of time deemed acceptable in order to change control parameters. The EMRAX 188 motor is rated for up to 7600 RPM. As the motor has 10 pole pairs, the maximum electrical frequency is $f_{e,max} = 10\frac{RPM_{motor}}{60} = 1.266$ kHz. The switching frequency should be ten times greater than the maximum electrical frequency for the output current waveforms to be reasonable sinusoidal, suggesting a minimum switching frequency 12 kHz. Additionally, higher switching frequency produce output waveforms with less harmonics, which shifts losses away from the motor, to the motor controller. Higher switching frequencies also allow for a lower rating of the DC link capacitance. The maximum output mechanical frequency is proportional to the DC voltage, and how effectively it is being used. The EMRAX 228 is rated for 6500 RPM, with a DC voltage of 470 V. The supply available for testing was rated for 300 V, and capable of supplying 1.5 kW. While the voltage of this supply would be large enough to reach 4000 RPM, the free spinning losses of the motor would exceed 1.5 kW. Therefore, a reasonable speed metric would be 3000 RPM, which would see just over 1 kW of losses in the motor. A

typical value for the inner control loop frequency is one tenth of the switching frequency. The lowest used switching frequency of a motor controller for the team was 8 kHz. One tenth of this is 800 Hz. On the car, the rate at which torque commands are sent to the motor controller is 5 ms. If a fault is detected, the controller should safely shut down in less than one cycle.

1.5 Budget

A comparison of the proposed and actual budgets can be found in table 1-3. The PCB components were more expensive than anticipated, and as a result, spare DC link capacitors courtesy of UMSAE Formula Electric were reused.

Table 1-3: Project budget

Item	**Proposed Cost**	**Actual Cost**
IGBT Module	$506	$509.24
Shunt resistor	$30	$61.62
DC-Link Capacitor	$168	$0
Gate Drivers	$70	$77.68
PCB Manufacturing + Shipping	$100	$0
Microcontroller Development Boards	$64	$42.11
PCB Components	$300	$686.69
Cooling Plate	$0	$0
High Current Busbars, Cable, Lugs	$0	$0
Busbar Manufacturing	$0	$0
Expense Subtotal	$1238	$1377.34
Capstone Funds	($400)	($400)
Support from Dr. Carl Ho	($838)	($977.34)
Total	$0	$0

1.6 Design Summary

A block diagram of the project can be found in figure 1-2. This diagram shows the major components and interfaces of the project. An early CAD model of the prototype can be found in figure 1-3. The project was broken down into four modules: The power electronics, the control system, the embedded systems software, and the graphic user interface.

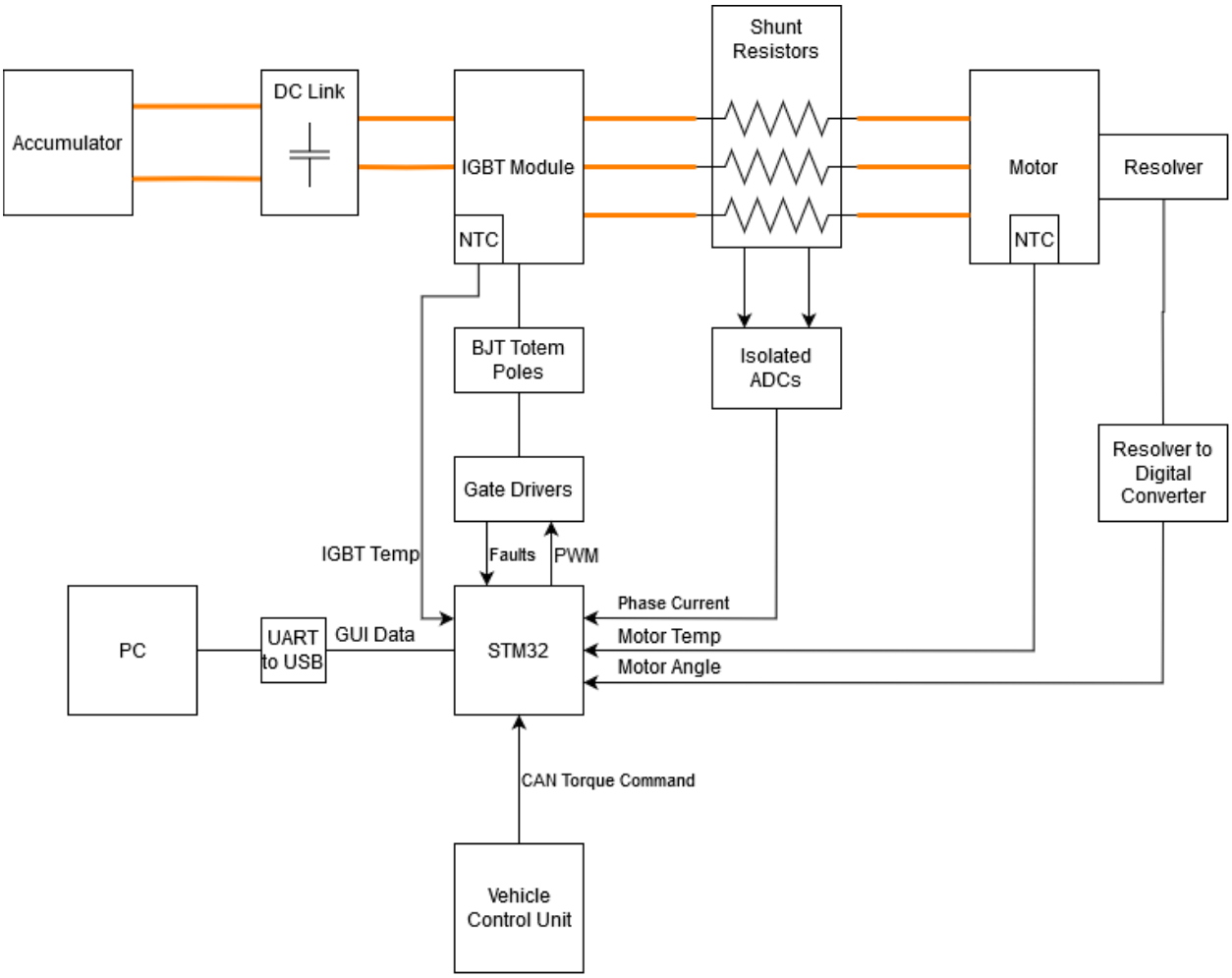
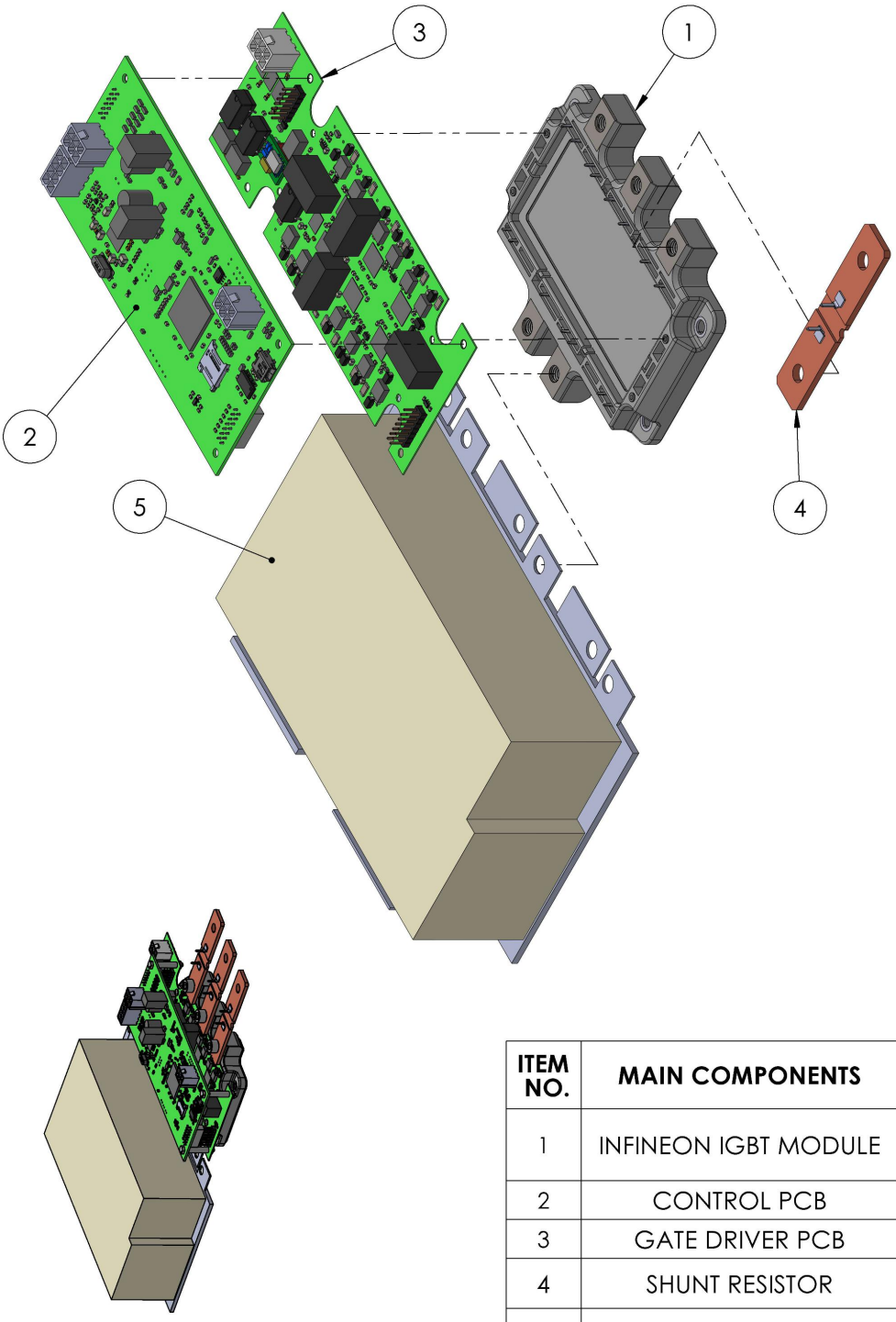


Figure 1-2: Systems architecture

Figure 1-3 below shows the initial components of the motor controller prototype.



ITEM NO.	MAIN COMPONENTS	QTY.
1	INFINEON IGBT MODULE	1
2	CONTROL PCB	1
3	GATE DRIVER PCB	1
4	SHUNT RESISTOR	3
5	DC-LINK CAPACITOR	1

Figure 1-3: Exploded and collapsed model views of the early motor controller prototype

A summary of all major design features can be found below.

- Designed for the EMRAX 188 motor
  - 400 VDC
  - 200 Arms
  - 6500 RPM
  - 60 kW
- A 6-pack IGBT module
  - 200 A
  - 650 VDC
  - On-die negative temperature coefficient thermistor (NTC)
- DC link capacitor bank
  - Film capacitors, low equivalent series resistance (ESR) and equivalent series inductance (ESL)
  - 320 µF
- +15/-15 V IGBT gate driver operation
- 16 kHz switching frequency
- Field oriented control system
  - Torque control
  - Regenerative braking capability
  - 16 kHz current loop frequency
  - Resolver position feedback
- Controlled over CAN
- STM32F767ZIT6 microcontroller

  - 144LQFP package
  - Arm Cortex-M7
  - 32 bit MCU
  - 216 MHz clock
  - Double precision FPU
  - 2 MB Flash
  - 512 KB RAM
  - Complimentary PWM with programmable dead time
  - 4 digital filters for sigma delta modulator
- Two Custom 4-layer PCBs
  - Control PCB containing the microcontroller and all its direct interfaces
  - Gate driver PCB containing the gate driver and measurement circuitry
- Fault detection and protection
  - IGBT overtemperature
  - IGBT desaturation
  - Resolver failure
  - CAN failure
- On-board data logging
  - 16 kHz parameter logging using a 6000 point circular memory buffer
  - $>$ 1 kHz low frequency live logging
  - 16 kHz high frequency live logging using a 500 point memory buffer
- GUI for a PC
  - Support for Windows, Macintosh and Linux systems

- Data logs receivable over USB
- 2D graphing of data logs
- Data logs exportable as a CSV file
- Control system parameters configurable from PC

Chapter 2

Power Electronics Design

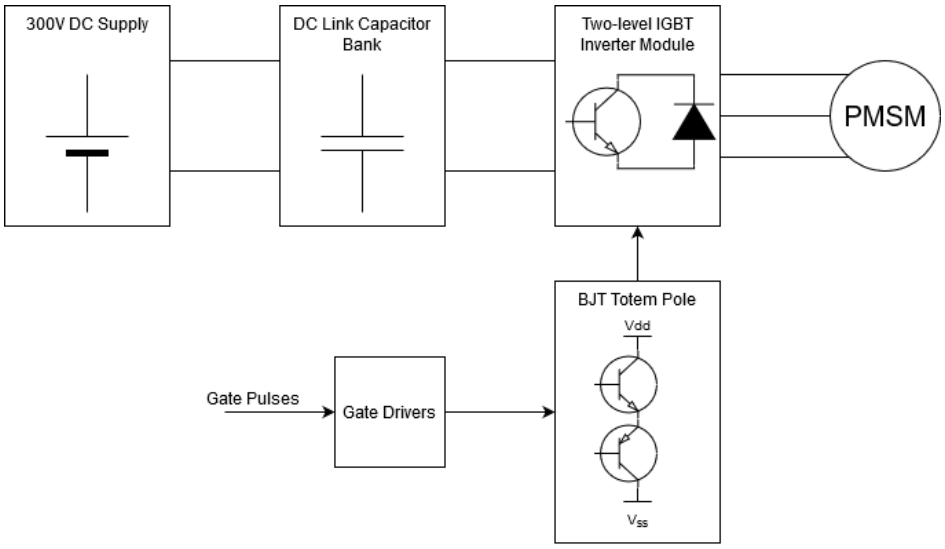


Figure 2-1: Power electronics block diagram

A two-level IGBT inverter topology was used. Film capacitors were used to support the DC bus. A BJT totem pole was used to provide the power required for driving the gates of the IGBTs.

2.1 Switch Selection

2.1.1 Requirements

The requirements for the semiconductor switch were mostly driven by the motor's parameters. In order to achieve full torque, an current rating of 200Arms was required. In order to operate over the full speed range of the motor, the DC voltage should be 400V. For protection of the device, an on-device temperature sensor was required.

Table 2-1: Semiconductor switch requirements

Parameter	Requirement
DC Voltage Rating	$>$ 400 VDC
Peak Current Rating	$>$ 200 Arms
Continuous Current Rating	$>$ 100 Arms
Temperature Sensor	On silicon die

SiC MOSFETs, while offering higher switching frequency capabilities and an ability to operated under higher temperature, were not considered due to cost and availability of single modules at these ratings. As this was our first time experience with motor controller design, we searched for a device with a built in NTC. We were unsure if external sensors would adequately capture the true temperature of the device.

2.1.2 Selection

These factors led to the selection of the Infineon FS200R07PE4 (datasheet available in [10]). This was the most inexpensive and smallest option that met the requirements. An image of this module can be found in figure 2-2. This module contains all six IGBTs in one package, as has been reported to have a sufficient continuous current rating in a similar application in [22]. As useful tool for determining IGBT operation and losses is Infineon's IPOSIM, available at [9]. This tool suggested the selected IGBT was capable of maintaining 200 Arms output, if the switching frequency was reduced to 8 kHz. With this information, design could proceed.

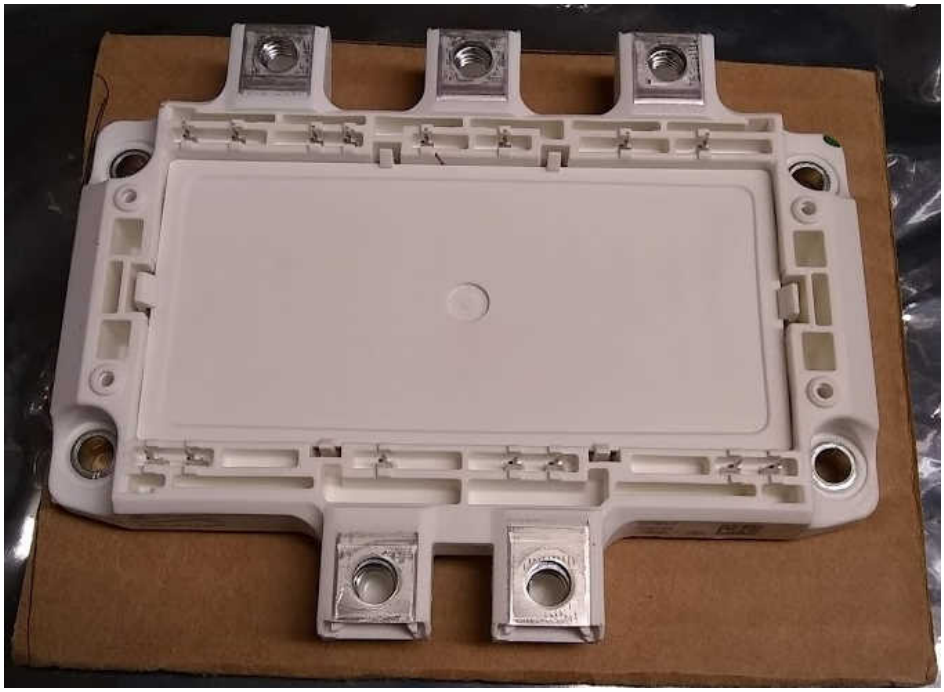


Figure 2-2: Infineon FS200R07PE4 IGBT module

2.2 DC Link Capacitors

The bus capacitors (also called DC link capacitors) are needed to supply the AC portion of the IGBT module input current. The accumulator contains a large equivalent series inductance (on the order of mH), and as such, simply cannot supply the fast transient currents required. This problem is difficult to solve analytically, so we took a numerical approach. The input current waveform was generated from the main Simulink model, and can be found in figure 2-3. This waveform was acquired from a scenario in which the motor controller was supplying 60 kW to the motor.

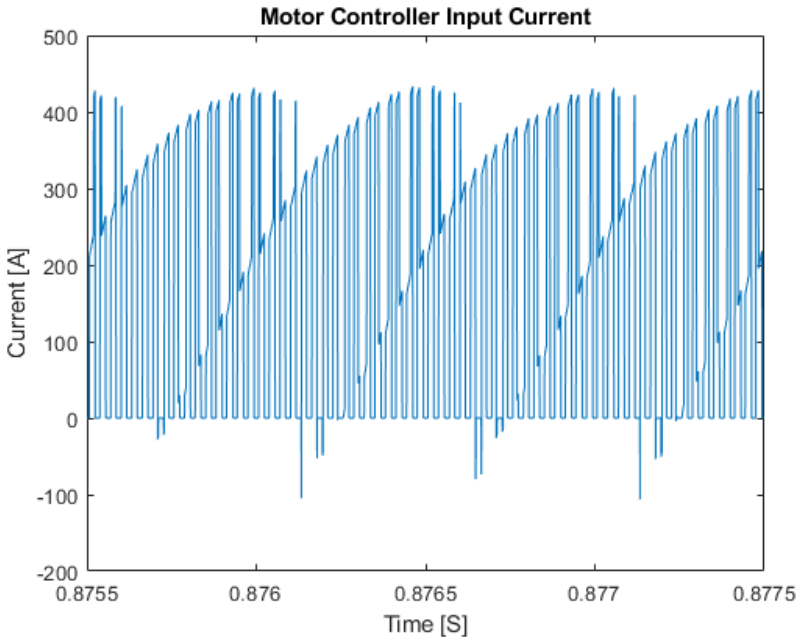


Figure 2-3: IGBT module input current

2.2.1 Determination of Current Rating

In order to analyze this waveform, to determine the DC link capacitor requirements, we assume that the capacitors supply the entire AC portion of this waveform. By subtracting the average current from the waveform in figure 2-3, and calculating the RMS of the resulting current, we are able to obtain the RMS current rating requirement of the DC link. Using equation (2.1), this was determined to be 168 Arms, where I_{cap} is the current leaving the capacitor bank, I_{input} is the current entering the IGBT module, and $I_{avg,input}$ is the average current entering the IGBT module.

$$I_{cap} = I_{input} - I_{Accumulator} = I_{input} - I_{avg,input} \tag{2.1}$$

2.2.2 Required Capacitance

The DC link's capacitance is the primary factor affecting the voltage ripple. Assuming a constant DC voltage, we can obtain the waveform for the instantaneous energy flow from the capacitor bank using equation (2.2). E_{cap} is the instantaneous energy output of the capacitor, P_{cap} is the instantaneous power supplied by the capacitor, V_{input} is the voltage across the capacitor, and I_{cap} is the current leaving the capacitor. The

resulting waveform can be seen in figure 2-4.

$$E_{cap} = \int P_{cap} dt = \int V_{input} I_{cap} dt \tag{2.2}$$

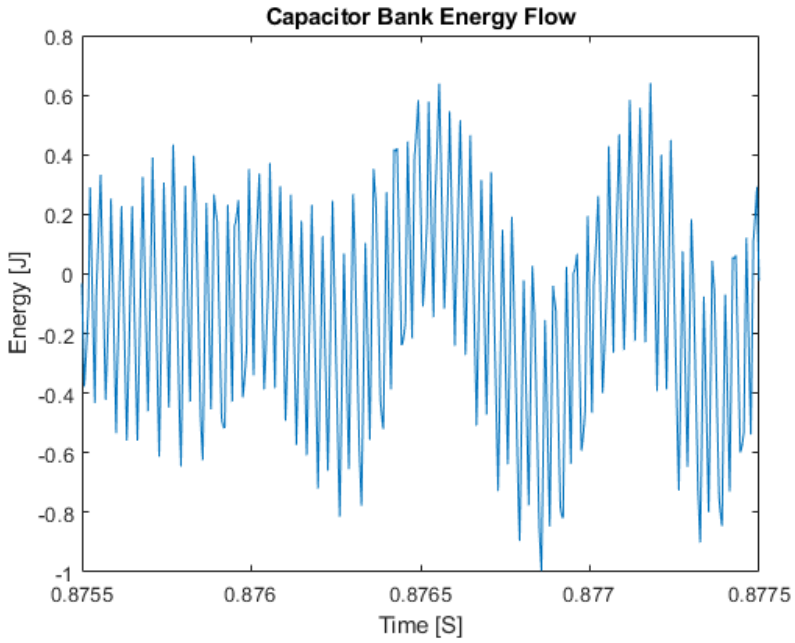


Figure 2-4: DC link capacitor energy flow

We can then come up with an expression for the required capacitance, as a function of allowable voltage ripple, energy ripple in the capacitor bank, and nominal DC link voltage. These are represented by the variables k, ΔE, and V_{nom} respectively. In these expressions, the voltage ripple has been defined as $k = \frac{V_{min}}{V_{nom}}$. The final expression can be found in equation (2.3). C_{min} is the minimum required capacitance, ΔE is the peak to peak energy waveform ripple, and V_{nom} is the input DC voltage. A detailed derivation of this can be found in appendix section A.2.

$$C_{min} = \frac{2\Delta E}{V_{nom}^2(1-k^2)} \tag{2.3}$$

Setting $k = 0.95$, $\Delta E = 1J$ (based on the worst case ripple found in figure 2-4), and $V_{nom} = 300$ V, we obtain $C_{min} = 165$ µF.

2.2.3 Acceptable values of ESR & ESL

Every capacitor has a finite ESR and ESL. This is another factor that affects voltage ripple. In an effort to ensure optimal performance, we decided to set a maximum voltage ripple threshold of 0.1% of the nominal voltage (this works out to 0.3 V), at the fundamental switching frequency (16 kHz). Therefore, we can calculated the combined effect using equation (2.4). V_{ripple} is the observed voltage ripple, I_{rms} is the rms of the current leaving the capacitor, R_{ESR} and ωL_{ESL} are the ESR and ESL of the capacitor, and Z_{EQ} is combined impedance of these.

$$|V_{ripple}| = |I_{rms}| \cdot |(R_{ESR} + j\omega L_{ESL})| = |I_{rms}| \cdot |Z_{EQ}| \tag{2.4}$$

Given a current of $I_{rms} = 168$, find that the maximum acceptable value for the equivalent impedance is $|Z_{eq}| = 1.79m\Omega$.

2.2.4 Summary of Requirements

A summary of the above sections can be found in table 2-2. Once the requirements had been determined, the search for an adequate component began.

Table 2-2: Summary of DC link capacitor requirements

Parameter	Requirement
Voltage Rating	>400 VDC
Current Rating	>168 Arms
Capacitance	>165µF
ESR & ESL (16 kHz)	<1.79 mΩ

2.2.5 Capacitor Type Considerations

As discussed in [22] and [21], the capacitor materials and construction are the main contributing factors of the characteristics we are concerned with. While electrolytic capacitors are able to provide more than sufficient capacitance and voltage ratings, they have significantly higher ESR and ESL. In this application, film capacitors tend to offer better characteristics, and as an added bonus, are non-polarized.

2.2.6 Capacitor Selection

The above requirements led to the selection of the 944U101K801AAI film capacitor from Cornell Dubilier. While a single capacitor alone would not meet the current rating, three of these capacitors in parallel would meet the ratings. Selected properties of this capacitor can be found in table 2-3.

Table 2-3: Selected datasheet values for 944U101K801AAI, taken from [5]

Parameter	Rating
Voltage Rating	800 VDC
Current Rating	74 Arms
Capacitance	100 µF
ESR	0.5 mΩ
ESL	20 nH
Equivalent impedance (16kHz)	0.5003 mΩ

This would have been the selected capacitor. However, an existing capacitor bank was re-used from an old motor controller owned by the Formula Electric team, due to budgetary constraints. At the time of purchase, there was enough budget for either the capacitors, or to replace the IGBT module if it were to be damaged. As a result, the capacitors as seen in figure 2-5.

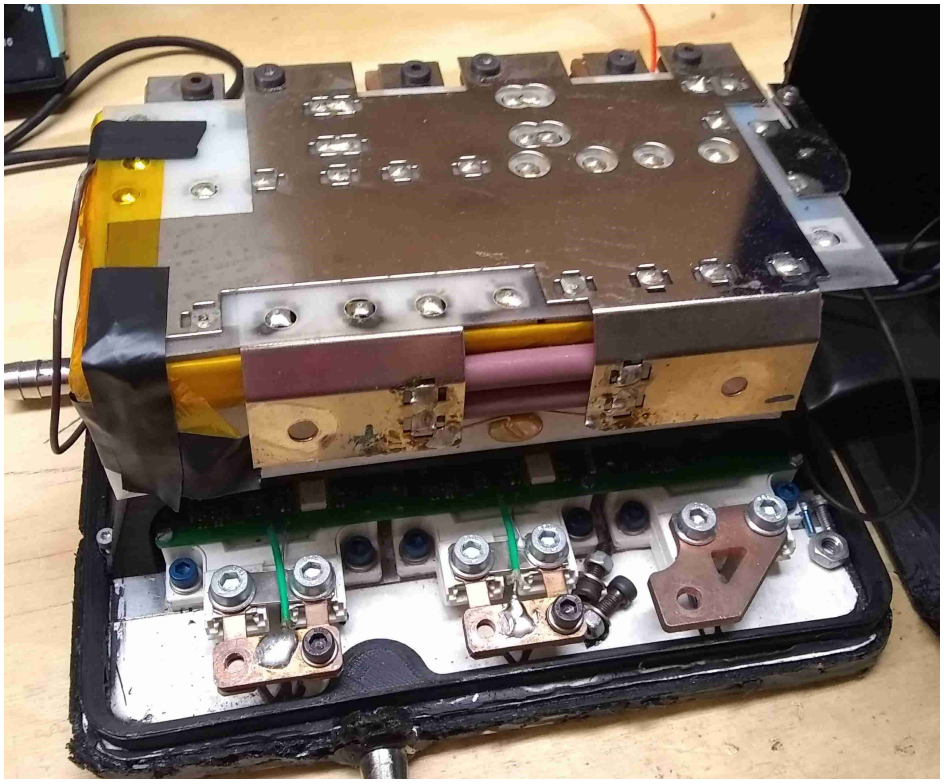


Figure 2-5: Salvaged bus capacitors used in motor controller

2.3 Gate Driver Circuit

2.3.1 Requirements

In order for the IGBTs to start conducting current, their gates must be charged beyond a threshold voltage. In order to stop conducting current, these gates must be charged below a threshold voltage. Additionally, these six gates require individual isolated power supplies. The relevant IGBT parameters can be found in table 2-4.

Table 2-4: Selected IGBT parameters, taken from [10]

IGBT Parameter	Value
Gate-emitter peak voltage	+/-20 V
Typical gate threshold voltage	5.8 V
Internal gate resistance	2 Ω
Input capacitance	12 nF
Reverse transfer capacitance	0.38 nF
Typical turn-on delay	0.14 µs
Typical turn-off delay time	0.42 µs

The requirements for the gate driver circuit was driven by the selected IGBT, and the requirements of the team. These requirements are as follows:

- Galvanic seperation between GLV and TS systems
- Operation for bipolar supply
- +15/-15 V operation
- 15 A source and sink current rating
- Short circuit detection
- Undervoltage lockout on TS side
- Enable or disable functionality
- Adjustable PWM deadtime

The galvanic separation is required by rules, and are required for the gate drivers to function properly. A bipolar gate driver power supply was pursued due to performance improvements over a unipolar supply. +15/-15 V was the maximum gate voltages the team felt comfortable operating with, and 15 V power supplies were readily available. The 15 A current rating is based on the maximum voltage difference (30 V

with a +15/-15 V supply), and the minimum gate resistance ($R_{gate,min} = R_{internal} = 2\ \Omega$). This gives us a maximum current of $I_{max} = \frac{V_{max}}{R_{gate}} = 15\ A$. The minimum recommended gate resistance of 3 Ω was suggested in [22]. In order minimize the chance of IGBT damage, short circuit detection was required. In order to ensure the controller did not start switching until all power supplies are ready, an undervoltage lockout was required. In order to shut down the gate drivers in the event of a fault, there needed to be an enable or disable functionality. Finally, there needed to be a way to set an appropriate deadtime. The final circuit used can be found in figure 2-6.

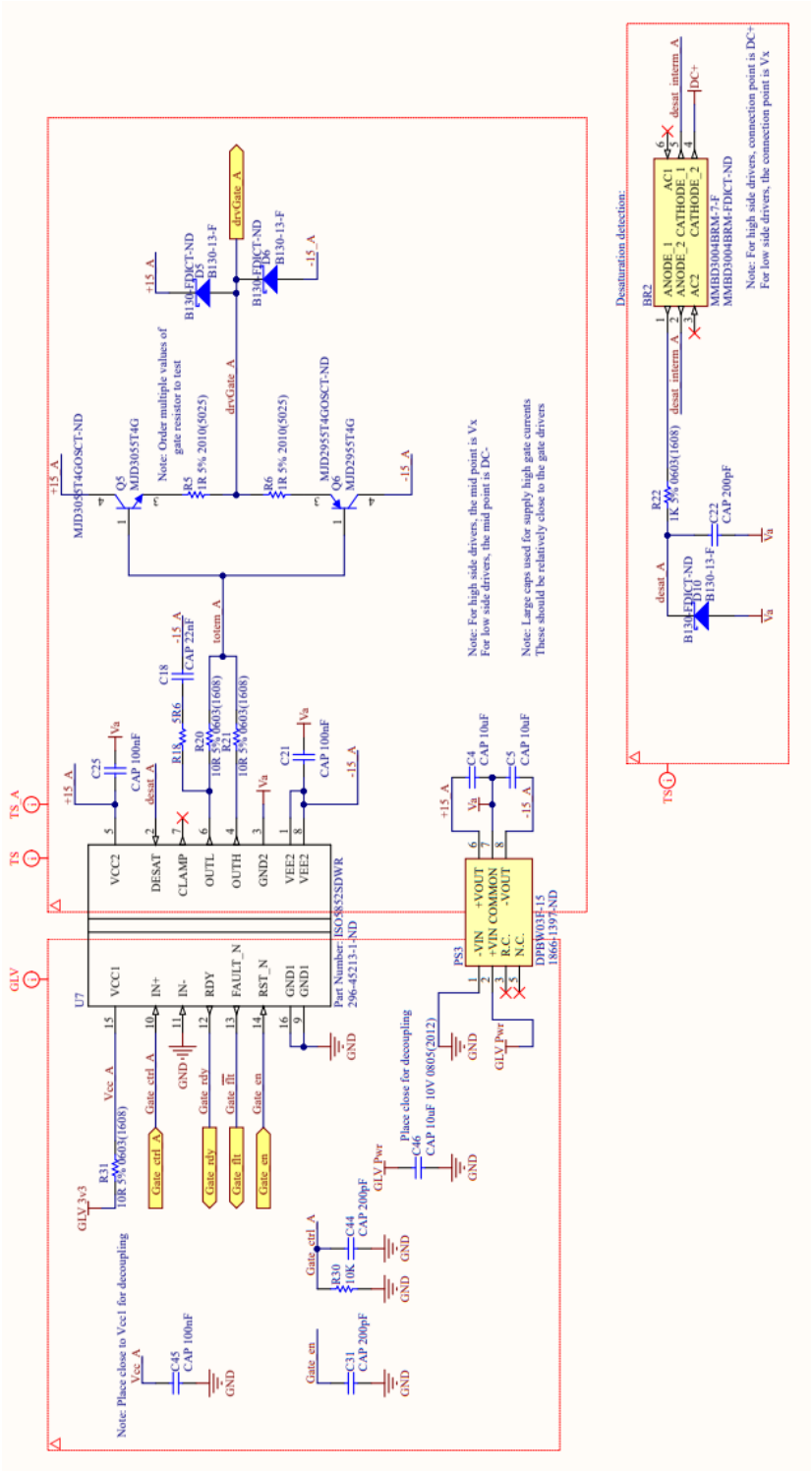


Figure 2-6: Schematic for gate driver circuit

2.3.2 Gate Driver IC

We selected the ISO5852SDWR gate driver IC from TI (datasheet available at [14]). We choose to use a dedicated IC for the gate driver, as it took care of signal isolation, undervoltage lockout, enable control, and short circuit detection in a single package.

Note about desaturation detection.

2.3.3 Gate Driver Supply

All three high-side drivers require their own power supply, however all three low side drivers can share a supply. These supply are referenced to the gate they are driving (and therefore need isolation between the inputs and outputs. This supply needed to work over the full range of our GLV systems' battery voltage (12-16.8 V). The equation used to calculate the required gate driver power is found in equation (2.5), where P_{gate} is the power required to switch the gate, Q_{gate} is the charge of the gate, f_{sw} is the inverter's switching frequency, ΔV_{gate} is the differential voltage used to switch the gates, and C_{ge} is the IGBT's gate to emitter capacitance. It has been taken from [3].

$$P_{gate} = Q_{gate} \cdot f_{sw} \cdot \Delta V_{gate} + C_{ge} \cdot f_{sw} \cdot \Delta V_{gate}^2 \tag{2.5}$$

In our case $Q_{gate} = 2.15$ µC, $f_{sw} = 16$ kHz, $\Delta V_{gate} = 30$ V, and $C_{ge} = 12$ nF. This indicates a requirement of $P_{gate} = 1.4$ W. This led to the selection of the DPBW03F-15 supply for the high side drivers, and the DPBW06F-15 supply for the low side drivers. The datasheet can be found in [28].

These power supplies were rated for only 100 mA. Therefore, in order to achieve the peak 15 A, bulk capacitors were used. To size these capacitors, we first calculated the energy required to switch the gate: $E_{switch} = P_{gate}/f_{sw} = 70$ µJ. We then selected capacitors that would hold 10 times as much energy as this: $C_{bulk} = \frac{10 \cdot E_{switch} \cdot 2}{\Delta V_{gate}^2} = 6.2$ µF. This was rounded up to 10 µF.

2.3.4 BJT Totem Pole

The gate driver IC is only able to source 2.5 A and sink 5 A. Ideally, the gate drivers needed to source and sink 15 A (assuming no external gate resistance) at collector-emitter voltage of 30 VDC. External circuitry was required to meet this parameter. A list of requirements can be found in table 2-5.

Table 2-5: Current buffer requirements

Parameter	Requirement
Current rating (at 30 V)	15 A
Transition Frequency	2 MHz
Package	Surface mount
-	Complimentary pair available

As space on gate driver PCB was particularly limited, a surface mount device was desired. MOSFETs can easily meet the voltage and current requirements as well as the timing requirements in any package, however they would require additional circuitry to operate. Instead, BJTs were chosen. In order to minimize a propagation delay, a device with a corner (transition) frequency of 100 times the maximum theoretical switching frequency (20 kHz) was required. We were not able to find a device which met all these requirements, and settled on the MJD2955/MJD3055 complimentary pair (datasheet found in [24]). This device is capable of supplying 10 A at 30 V for a brief period of time.

2.3.5 Gate Resistance

The IGBT gate and gate driver circuit is effectively an RLC circuit. The IGBT gate has a capacitance and internal resistance, the PCB trace an inductance, and an additional external resistance may be included. A smaller gate resistance will allow the gate to charge/discharge faster, and therefore reduce IGBT turn on and turn off times, reducing switching losses. Therefore, it is desirable to use as small a gate resistance as possible. However, a small resistance could lead to an underdamped RLC circuit, causing hazardous voltage oscillations at the IGBT gate. In order to prevent oscillations and maintain a small gate resistance, the trace inductance should be minimized. During PCB design, these traces were made as short and wide as possible.

While the IGBT gate capacitance is a known quantity, the loop inductance is not. TI's recommended practice is to start with the smallest external gate resistance, and then increasing it if necessary [13]. Our smallest external gate resistance of 1 Ω was determined by the current rating of the BJT totem pole. Once the gate driver PCB was assembled, the circuit was tested on the bench. The results of the test can be found in figure 2-7. As the gate did not display any oscillatory ringing, the 1 Ω external gate resistance was sufficient.

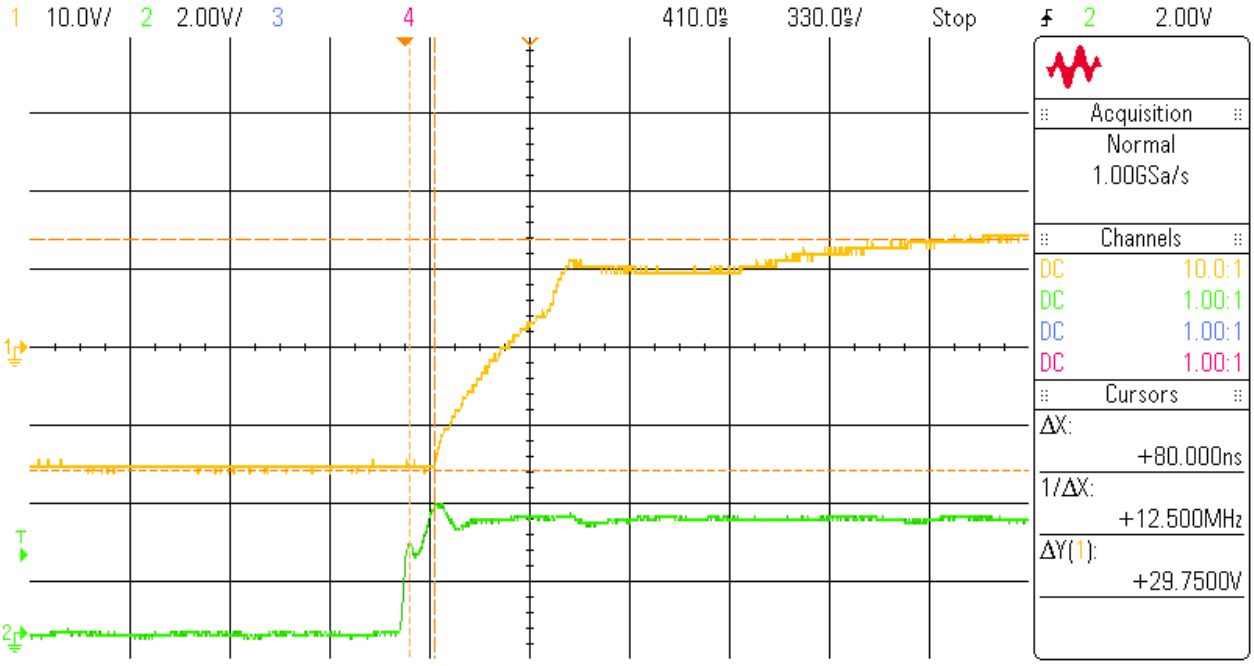


Figure 2-7: IGBT gate charging

2.3.6 IGBT Dead Time Calculation

Dead time is used to prevent shoot-through. In our design, a programmable dead time was implemented on the microcontroller. Infineon provides an application note on calculating appropriate dead times, found in [8]. The suggested equation can be found in equation (2.6), where $t_{d,off,max}$ is the maximum turn off delay, $t_{d,on,min}$ is the minimum turn on time delay, $t_{pd,max}$ is the maximum propagation delay of the driver, $t_{pd,min}$ is the minimum propagation delay of the driver, 1.2 is a safety factor, and t_{dead} is a suitable dead time.

$$t_{dead} = (t_{d,off,max} - t_{d,on,min} + t_{pd,max} - t_{pd,min}) \cdot 1.2 \tag{2.6}$$

Unfortunately the IGBT and gate driver datasheets only provide typical values for turn on, turn off, and propagation delays. The IGBT turn on and turn off time are dependent on external factors such as the controller stability and slew rate. The typical procedure for tuning this is to start with a conservative dead time of a couple µs, and then reduce it and observe that the controller is still stable.

Chapter 3

Controls

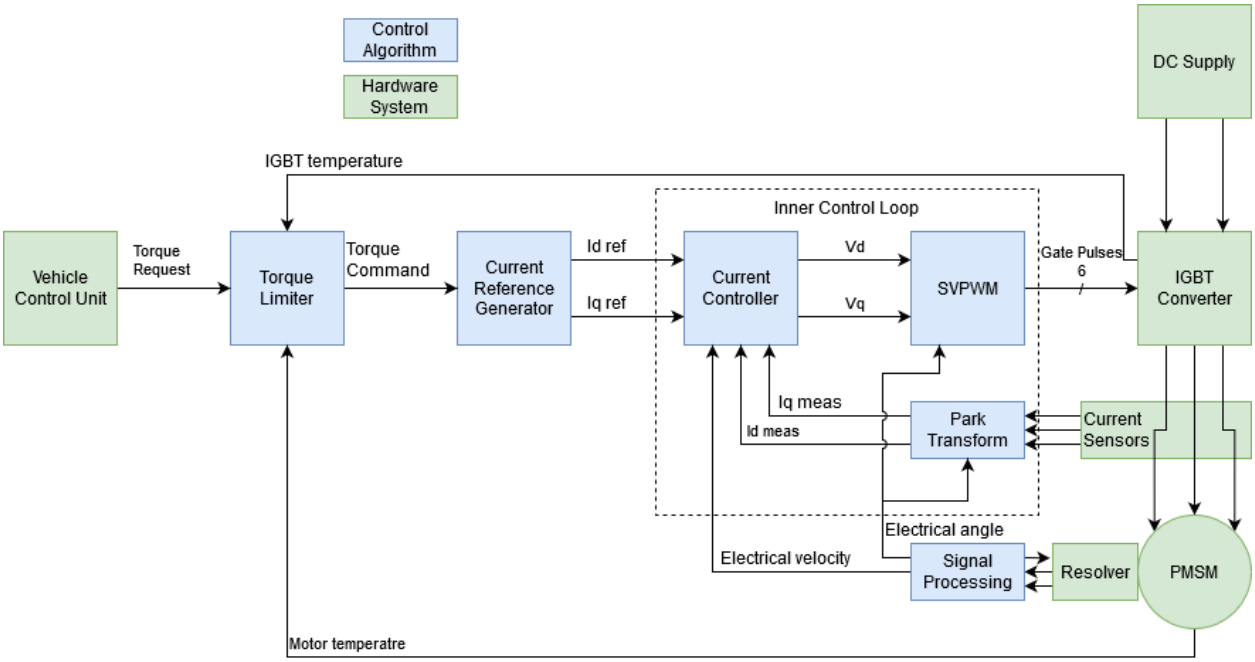


Figure 3-1: Control systems block diagram

A bandwidth-tunable, field oriented control system for torque control of a motor was designed and tested in Simulink. A block diagram of this control system can be found in figure 3-1. This model was then used to test the precision requirements of the angle and current sensors. A resolver is used for angle feedback, and hall effect sensors for current measurements. The control system is run on a 216 MHz STM32F7 microcontroller. An algorithm to determine the resolver angle offset was also developed and tested in Simulink.

3.1 Requirements

The requirements for the control system were as follows:

- Torque control of motor
- Capable of regenerative braking
- Operate with motor speed ranging from 0-6400 RPM
- Vector Control
- Space vector PWM
- Resolver based position feedback
- Programmable torque ramp

These requirements were defined by the performance requirements of the Formula Electric team, and the comparable motor controller alternatives. As the device is operating in a vehicle, torque control of the motor was required. While the team does not currently implement regenerative braking, it intends to in the future, and so any motor control system designed must support this functionality. The control system must also support the full operating speed of the EMRAX 188 motor. As alternative controllers implement vector control, along with space vector modulation, this was also a requirement. Resolver position feedback is the suggested method of the motor. Finally, as the car implements a chain drive, a torque ramp is used to prevent damage to the powertrain on startup.

These requirements led to the selection of field oriented control. There are two main types of vector control: direct torque control (DTC) and FOC. A comparison can be found in [18]. In this application, the main performance benefit of DTC, the improved dynamic torque performance, would not provide a significant benefit as the torque is slowly ramped up to the requested value, instead of a step response. This drove the decision to pursue FOC instead.

3.2 Field Oriented Control

The field oriented control system can be found in figure 3-1. The methodology for designing a bandwidth tunable motor control system is described in [30] and [29]. A torque controller generates the current values required to produce the requested amount of torque. A current controller uses these values as references, and adjusts the output voltage references to regulate these currents. These voltage references are realized

using Space-vector Pulse Width Modulation (SVPWM).

The implemented current controller can be found in figure 3-2. The control system is derived below. Several additions are present: a limit on current magnitude, and a limit on the output voltage. The current limit ensures that the control system will not request more current than the system is rated for, and the output voltage enforces the limit on output voltage from the DC link voltage.

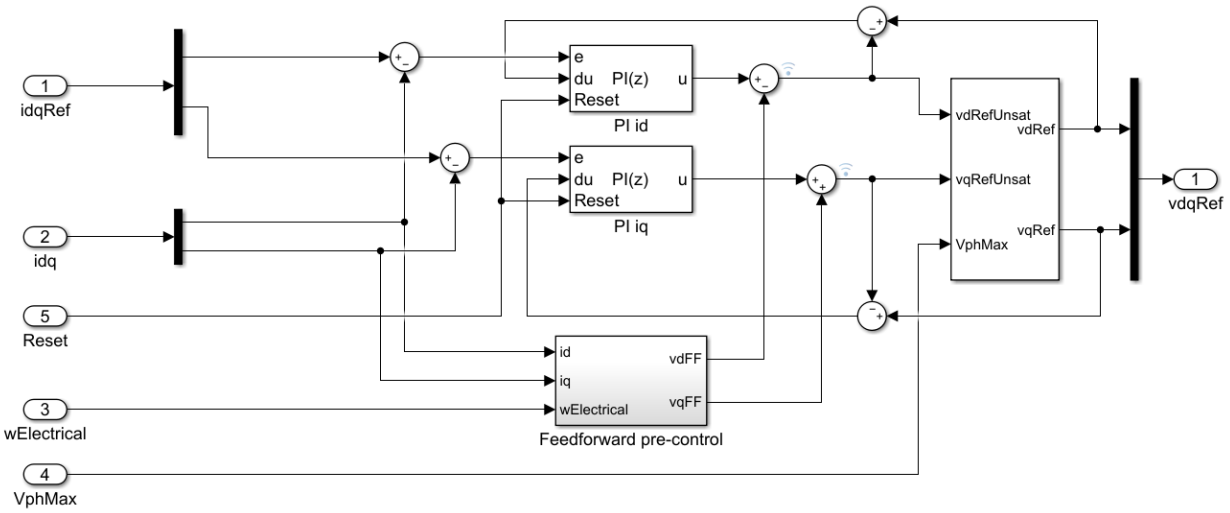


Figure 3-2: Simulink control system diagram of the current controller, based on [19]

3.2.1 Motor Model

The design of the control system started with a model of the system. The approximation used can be found in figure 3-3, and is described in [2]. The system is represented by a resistance, inductance, and back EMF in the dq-reference frame, allowing the motor to be controlled in a similar manner as DC motors. R_s is the stator resistance, L_{dq} is a vector of the d-axis and q-axis inductances, V_{dq} is the d-axis and q-axis voltages at the motor terminals, I_{dq} is the d-axis and q-axis currents, and $V_{EMF,dq}$ is the generated d-axis and q-axis back EMFs. From this circuit model the transfer function in the Laplace domain can be derived, as found in equation (3.2). As the motor torque is a function of current, the input to the system is voltage, and the output is the current.

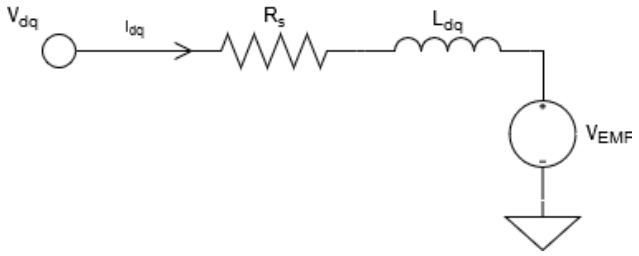


Figure 3-3: Circuit model of a permanent magnet motor in the dq reference frame

$$V_{dq}(s) = I_{dq}(s) \cdot (R_s + sL_{dq}) + V_{EMF,dq} \tag{3.1}$$

$$G(s) = \frac{I_{dq}(s)}{V_{dq}(s) - V_{EMF,dq}} = \frac{1}{R_s + sL_{dq}} = \frac{\frac{1}{L_{dq}}}{(s + \frac{R_s}{L_{dq}})} \tag{3.2}$$

Where the equations for the back EMFs are as follows, where ψ_m is the magnetic flux of the motor, and ω_e is the electrical frequency of the motor:

$$V_{EMF,d} = -\omega_e L_q I_q \tag{3.3}$$

$$V_{EMF,q} = \omega_e (L_d I_d + \psi_m) \tag{3.4}$$

3.2.2 Current Controller

The transfer function of a PI controller can be written as equation (3.5). In this equation, $e(s)$ is the d-axis and q-axis current error. From equation (3.5), we can see that the pole of the motor's transfer function can be cancelled out by appropriately selecting appropriate PI controller gains.

$$G_c(s) = \frac{V_{dq} - V_{EMF,dq}}{I_{e,dq}(s)} = k_p + \frac{k_i}{s} = \frac{1}{s}(k_i + sk_p) = \frac{1}{s}k_a(s + k_b) \tag{3.5}$$

While the controller equation (3.5) would cancel out the pole of the motor, we are not able to directly control the term $V_{dq} - V_{EMF,dq}$, only the term V_{dq}. Instead, the back EMF term $V_{EMF,dq}$ was estimated using equation (3.3) and equation (3.3), then added to the PI controller's output. This is realisable through equation (3.6). $I_{ref,dq}$ is the reference current sent to the PI controller, $I_{meas,dq}$ is the measured current,

$\widehat{V_{EMF,dq}}$ is the estimated back EMF, and $V_{ref,dq}$ is the voltage reference sent to the Space-vector Pulse Width Modulation (SVPWM) module.

$$V_{ref,dq} = (I_{ref,dq} - I_{meas,dq})\frac{1}{s}k_a(s+k_b) + \widehat{V_{EMF,dq}} \tag{3.6}$$

Selecting appropriate PI controller gains results in a first order open loop transfer function, found in equation (3.7). This means that the entire system can effectively be tuned with one parameter, k_a, which will determine the bandwidth of the controller.

$$G_{open}(s) = G(s)G_c(s) = \frac{1}{s} \cdot \frac{k_a}{L_{dq}} \tag{3.7}$$

$$BW = \frac{k_a}{L_{dq}} \tag{3.8}$$

In order to cancel out the poles of the motor, and set the bandwidth of the system, k_a and k_b should be set as follows:

$$k_a = L_{dq} \cdot BW \tag{3.9}$$

$$k_b = \frac{R_s}{L_{dq}} \tag{3.10}$$

From k_a and k_b, we can find the more traditional gains of the PI controller, k_p and k_i, as found below.

$$k_p = k_a \tag{3.11}$$

$$k_i = k_a k_b \tag{3.12}$$

3.2.3 Torque Controller

The input torque request must be converted into current references. The equation for torque in a PMSM is shown in equation (3.13), as found in [2]. In this equation, T_e is the total electrical torque applied to

the motor, pp is the number of pole pairs of the motor, ψ_m is the magnetic flux of the motor, I_s is the stator current amplitude, β is the angle of the stator current vector, and L_d and L_q are the d-axis and q-axis inductances respectively.

$$T_e = \frac{3pp}{2}\psi_m I_s cos\beta + \frac{3pp}{4}(L_q - L_d)I_s^2 sin2\beta \tag{3.13}$$

In equation (3.13), $\frac{3pp}{2}\psi_m I_s cos\beta$ corresponds to the magnetic torque of the motor, and $\frac{3pp}{4}(L_q - L_d)I_s^2 sin2\beta$ corresponds to the saliency torque. As the EMRAX 228 has a relatively low saliency, the saliency torque was assumed to be zero. The above equation is then solved for $I_s = I_q$.

In order to protect the motor and IGBTs, a temperature-based torque limit was enforced. This function can be seen in figure 3-4. To protect the mechanical powertrain, a rate of change limiter was added. This was selected such that the controller would take 50 ms to ramp up to full torque from zero torque.

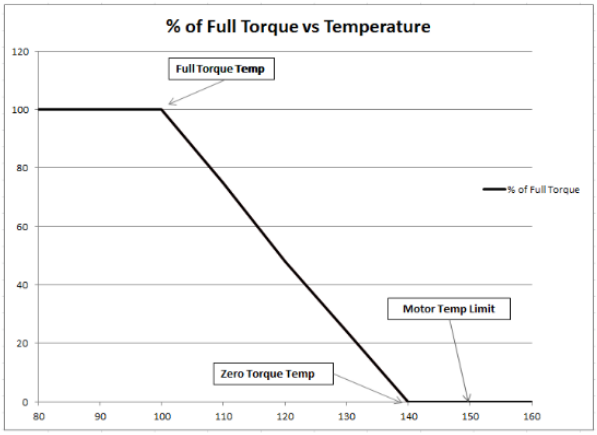


Figure 3-4: Control system temperature-based torque limit, taken from [20]

3.2.4 Bandwidth Tuning

To test the control system and tuning method, a model was developed in Simulink. The control system was tested across a range of motor speeds and torque values. The torque command and set motor speed waveforms can be found in figure 3-5. The results of this test can be found in figure 3-6. The results of testing the control system bandwidth tuned to 1600 Hz and 160 Hz can be found in and figure 3-7 respectively. As the control loop is running at 16 kHz, the maximum recommended bandwidth is 1.6 kHz,

which is one-tenth of the sample rate.

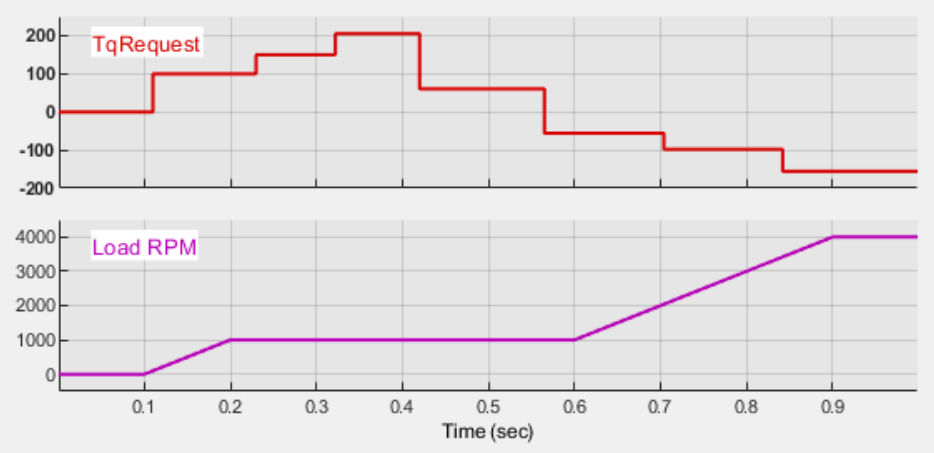


Figure 3-5: Input torque request and motor speeds used to test the control system

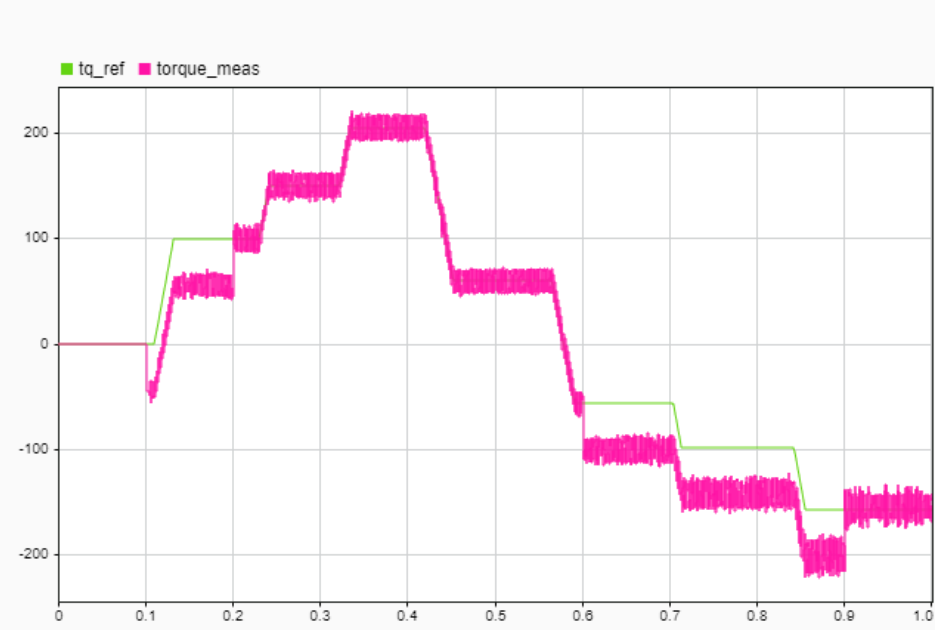


Figure 3-6: Control system dynamic torque response, tuned to a bandwidth of 1600

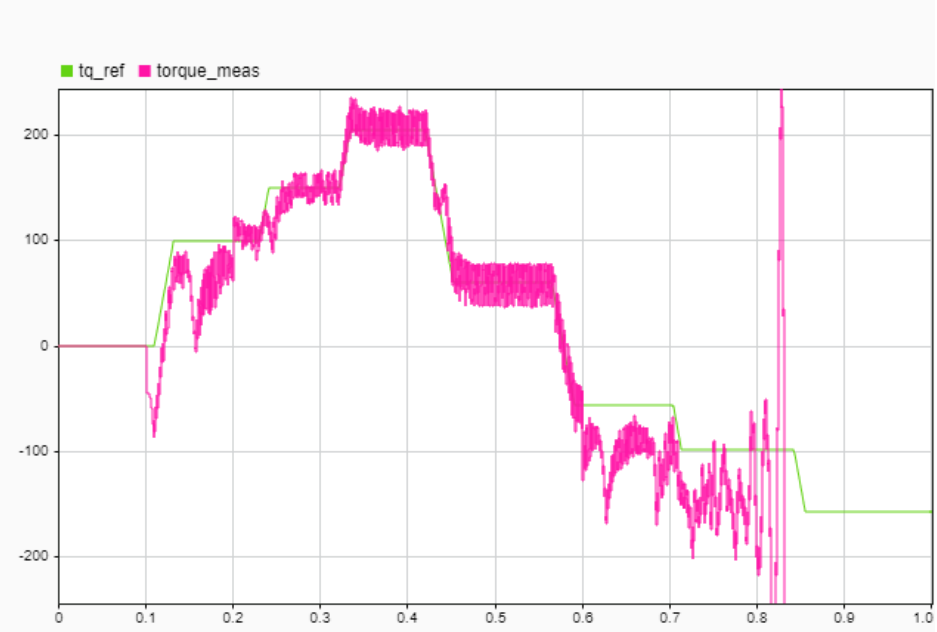


Figure 3-7: Control system dynamic torque response, tuned to a bandwidth of 160

In general, higher gains lead to a lower system stability but allow the control system to track higher frequency signals. We can see the effect of this in figure 3-6 and figure 3-7, where in figure 3-6 the systems is stable across the operable range of the motor, but in figure 3-7 the system becomes unstable at higher motor speeds. This is despite results suggested by the bode plots of these systems (found in figure 3-8 and figure 3-9) where both systems show large gain and phase margins. This is due to the saturating effect of the Space-vector Pulse Width Modulation (SVPWM) module, and the output phase voltage of the controller. In the simulations above the motor was being driven at a speed externally, which would not be the case in a real test. In order to verify that this would not be a problem encountered during testing addition simulations were run. The results can found in figure 3-10, where the motor is driving a small load and a speed-dependent torque is applied, representative of the wind resistance experienced by the UMSAE Formula Electric vehicle. Anti-windup was also added and tuned to reduce the effect of the integral windup.

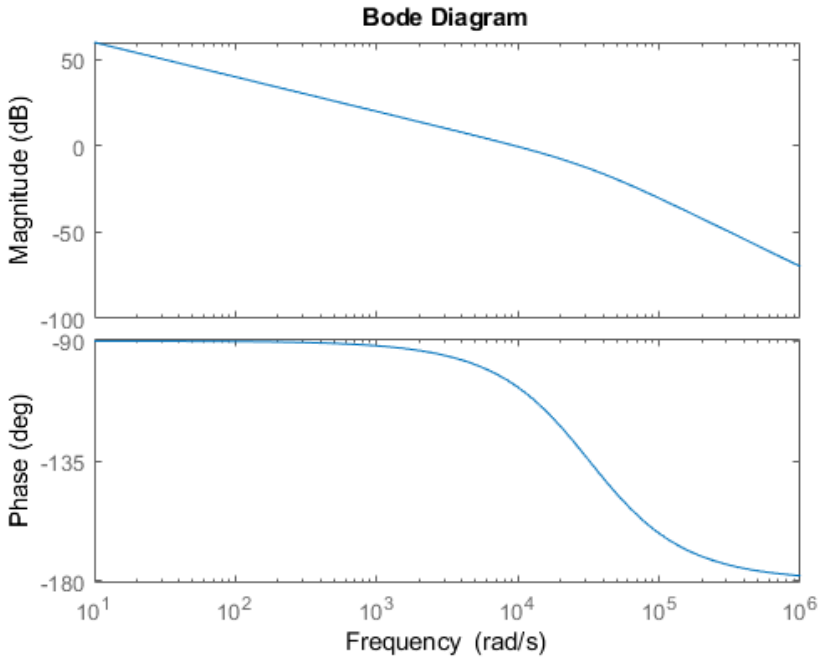


Figure 3-8: Bode plot, with system tuned to 1600 Hz

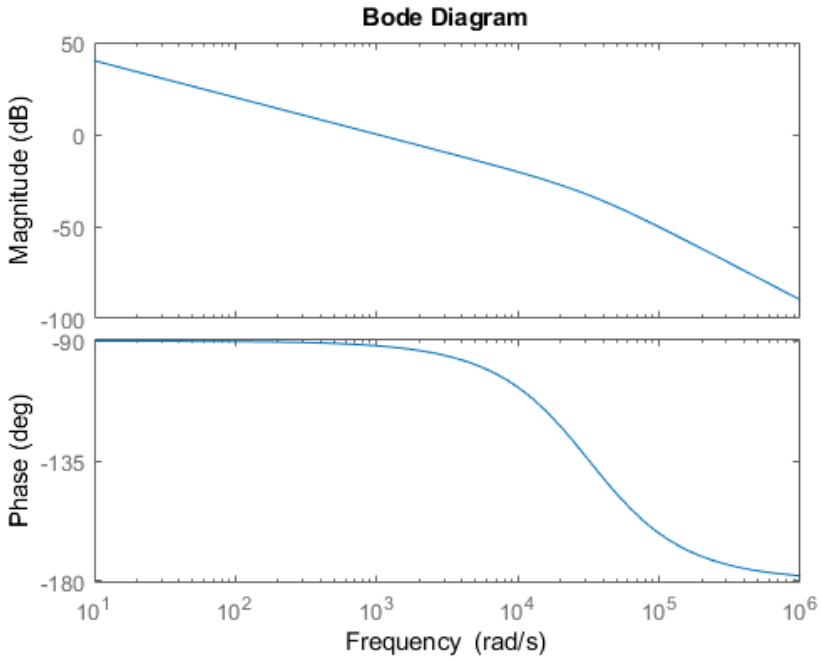


Figure 3-9: Bode plot, with system tuned to 160 Hz

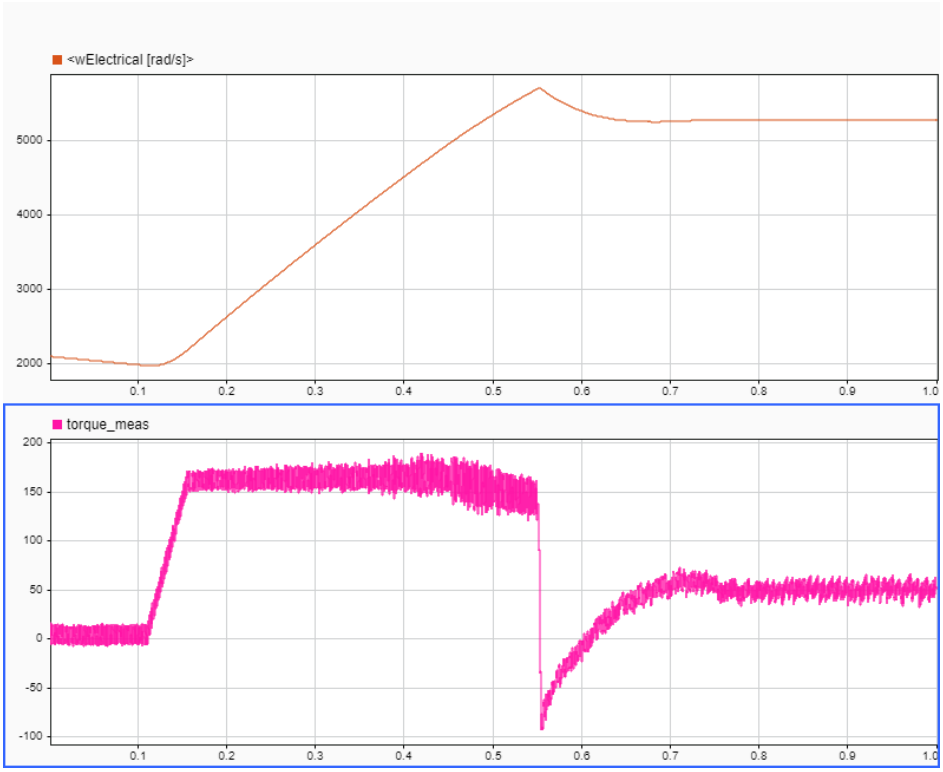


Figure 3-10: Simulation results

3.3 Resolver Interface

The resolver used was the TS2620N21E11. This is the single pole pair resolver suggested by EMRAX. Relevant datasheet parameters can be found in table 3-1. This device is used to measure the mechanical angle of the motor, and used by the control system to determine motor speed, use dq-transforms, and perform Space-vector Pulse Width Modulation (SVPWM).

Table 3-1: Selected resolver parameters, taken from [26]

Parameter	Value
Input voltage	7 Vrms
Input frequency	10 kHz
Transformation ratio	0.5 +/- 10%
Input impedance	70 + j100 Ω

3.3.1 Resolver to Digital Converter

In an attempt to limit the scope of the project, a dedicated resolver to digital converter (RDC) IC was used. The RDC used was the PGA411-Q1 (datasheet can be found in [15]). The requirements the led to this selection can be found in table 3-2. The resolution requirement was determined by testing the control system in Simulink, and the read rate is equal to the maximum desired switching frequency. The maximum RPM is the maximum motor RPM, and the excitation voltage and frequency were determined by the resolver's recommended operating point.

Table 3-2: Resolver interface requirements

Parameter	Requirement
Resolution	10 bit
Sample rate	$>$ 20 kHz
Max RPM	$>$ 6400 RPM
Excitation frequency	10 kHz
Excitation voltage	7 Vrms

3.3.2 Resolver Resolution Determination

In order to get a reasonably accurate angle measurement, there should be 100 points in one electrical period. This means that there needs to be at least 1000 resolver points in one mechanical rotation. This suggested a resolution of 10 bits. In order to verify this estimate, the Simulink control system model was used. The results of the 16, 8, and 6 bit resolver tests can be found in figure 3-11, figure 3-12, and figure 3-13 respectively. These graphs are a comparison of the requested and measured torque. Note: there is an offset between the measured and request torque from 0.1 s to 0.2 s, as well as from 0.6 s to 0.9 s. This is due to the way Simulink could measure the torque, while the motor was being accelerated by an external source.

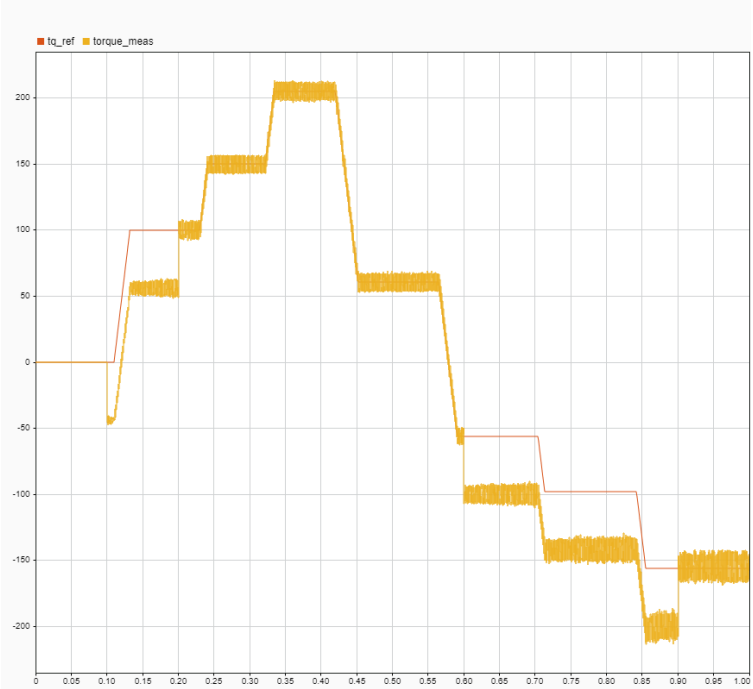


Figure 3-11: 16 bit resolver Simulink test

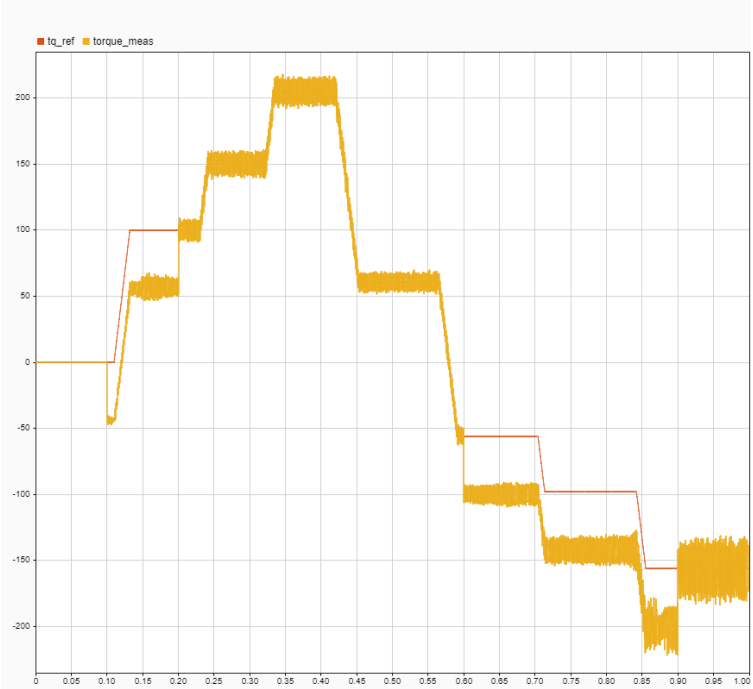


Figure 3-12: 8 bit resolver Simulink test

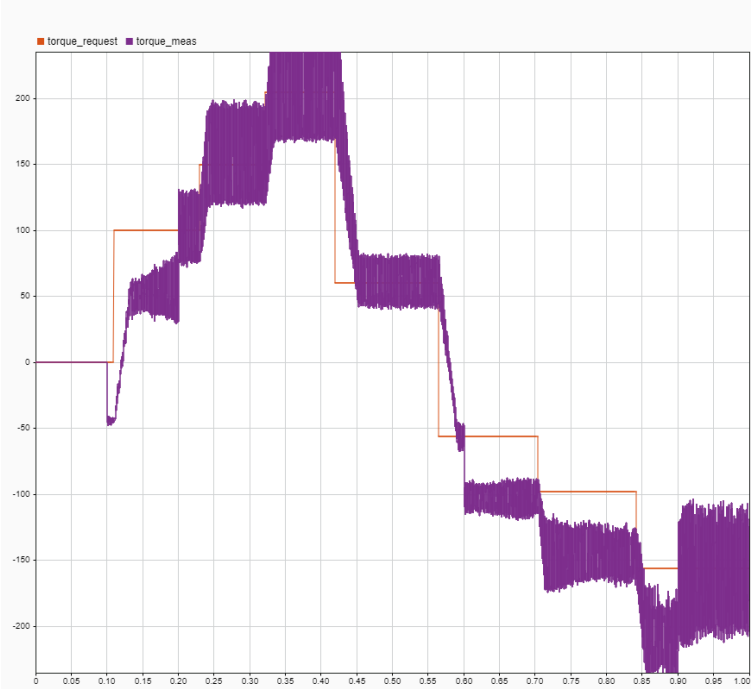


Figure 3-13: 6 bit resolver Simulink test

We can see that the system is stable and performing well with both 16 and 8 bit RDCs, but there is a noticeable increase in the torque ripple between the 16 and 8 bit tests. This is particularly noticeable toward the end of the tests. There is a dramatic increase in the torque ripple between the 8 bit and 6 bit tests, suggesting that an 8 bit resolver is the minimum viable resolution for this application. As a safety factor, the selected RDC should have a resolution at least 2 bits larger, leading to the requirement of a 10 bit RDC.

3.4 Current Measurements

Initially, shunt resistors with sigma-delta modulators were used in order to measure the phase current. During testing, there proved to be too much noise on these lines. There was at least +/-60 mV of noise, while the linear range of the sigma-delta modulator was only +/-50 mV. Instead, the current measurement was changed to use hall effect sensors. These sensors were moved further away from the noise source, and the output signal would be over the range +/-800 mV, significantly improving the signal to noise ratio. The selected hall effect sensor was the LEM HO-250S (datasheet available in [17]).

3.4.1 Current Measurement Requirements

In addition to the resolver, the phase currents are used as feedback by the control system. The requirements for the current measurements are found in table 3-3. The current must be measured every time the current control loop runs, which would be at a maximum rate of 20 kHz. It must also be able to measure the entire nominal current range, which is 200 Vrms (corresponding to +/-280 A peak). Over this range, there must be 10 bits of resolution, as discussed in section 3.4.3. Finally, the measurement must be isolated.

Table 3-3: Current sensing requirements

Parameter	Requirement
Measurement rate	20 kHz
Measurement range	+/- 280 A
Measurement resolution	10 bit over range +/- 280 A
Signal isolation	The GLV and TS systems must remain galvanically isolated
Conversion start	Must allow for simultaneous current readings

The two most common ways to measure current are hall effect sensors and shunt resistors [22], [21], [1]. A hall effect sensor measures the magnetic field, while a shunt resistor is just a known resistance that produces a measureable voltage when a current passes through it. Hall effect sensors are an inherently isolated measurement, while a shunt resistor would require additional circuitry to achieve this. Both methods would be able to achieve the performance requirements, however the shunt resistor measurements are cheaper and potentially smaller.

3.4.2 Shunt Resistor & ADC Selection

The shunt resistor need to be able to sustain a current of 200 Arms. The tolerance was not a concern, as we could calibrate them using a 4 wire measurement. This led to the selection of the WSBS8518L1000JK20. Key datasheet parameters can be found in table 3-4, and an image can be found in figure 3-14.

Table 3-4: Selected shunt resistor parameters, taken from [4]

Parameter	Value
Resistance	100 μΩ
Power rating	36 W
Inferred current rating	600 Arms

Figure 3-14: Image of the WSBS8518L1000JK20 shunt resistors

To measure the voltage across the shunt resistor, the AMC1106M05DWVR sigma-delta modulator (SDM) was used. A more traditional approach is to either use an isolated amplifier, or a regular adc and isolated the digital output. A sigma-delta modulator takes care of measurement, amplification, and isolation in a single IC, but requires additional filtering. This is achieved through hardware digital filters on the microcontroller. Key datasheet parameters can be found in table 3-5.

Table 3-5: Selected datasheet parameters of the AMC1106M05DWVR, taken from [12]

Parameter	Value
Input range	+/-50 mA
Voltage logic level	3.3 V or 5 V
Resolution	16 bit
Input clock	Up to 20 MHz

3.4.3 Current Sensor Resolution

In order to have a reasonable current measurement, there should be at least 1000 points over the full measured current range, from -280 A to 280 A. This suggested a required resolution of 12 bits. In order to verify this estimate, the Simulink control system model was used. The results of the 16, 10, and 4 bit resolver tests can be found in figure 3-15, figure 3-16, and figure 3-17 respectively. These graphs are a comparison between the requested and measured torque. Note: there is an offset between the measured and request torque from 0.1s to 0.2s, as well as from 0.6s to 0.9s. This is due to the way Simulink could measure the torque, while the motor was being accelerated by an external source.

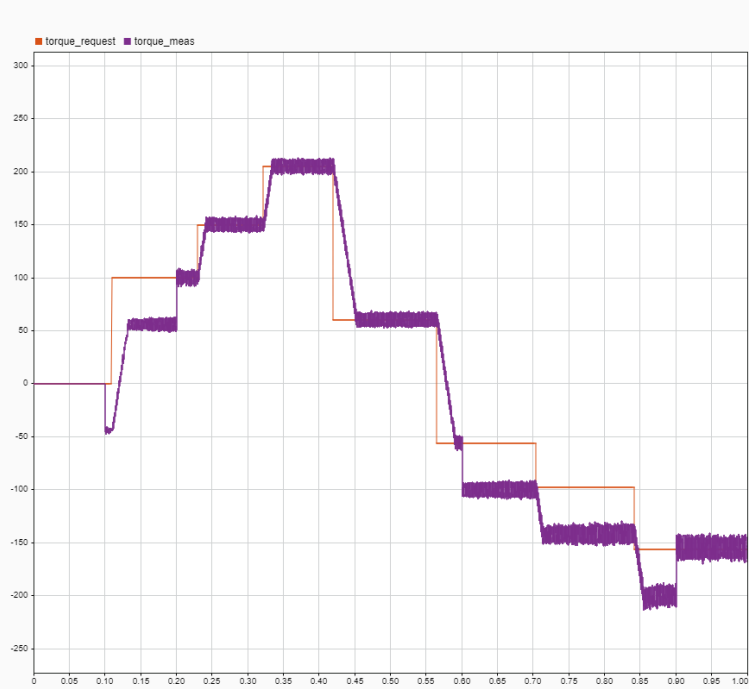


Figure 3-15: 16 bit current ADC Simulink test

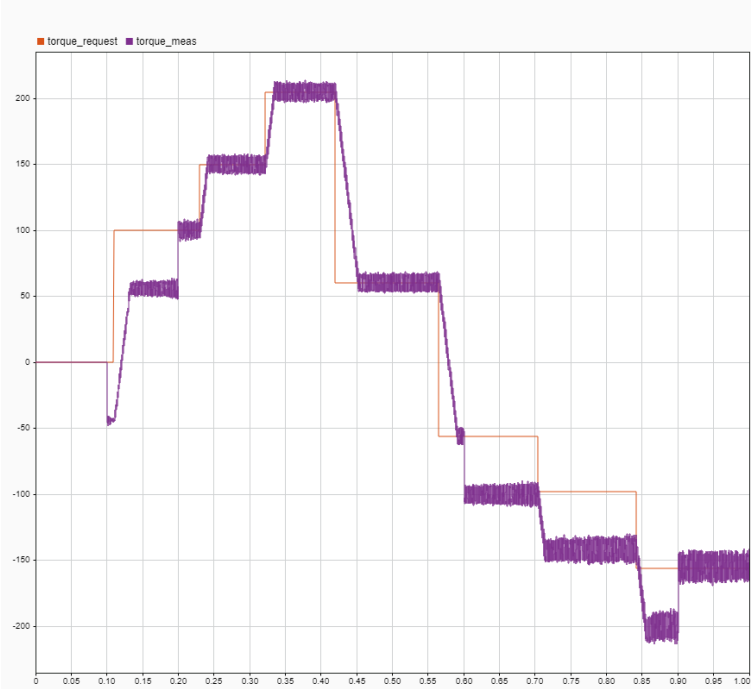


Figure 3-16: 10 bit current ADC Simulink test

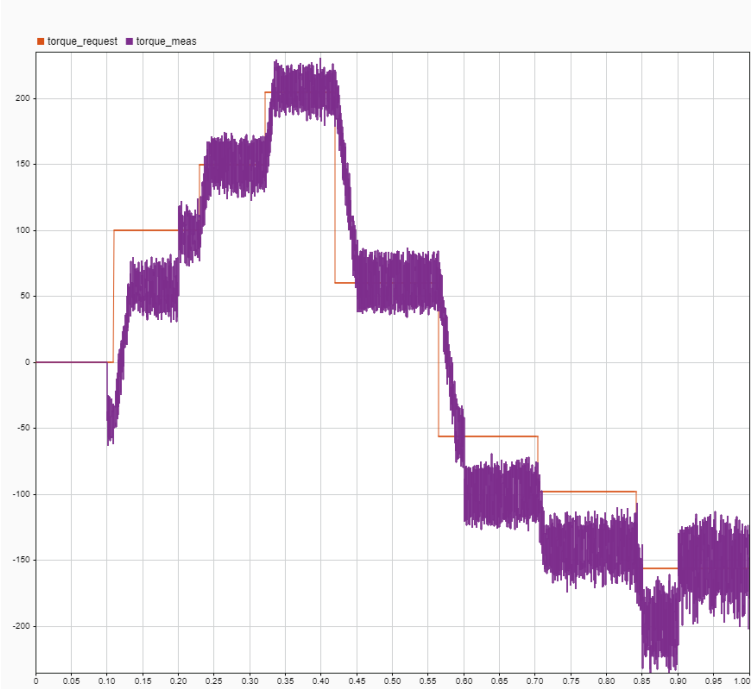


Figure 3-17: 4 bit current ADC Simulink test

We can see a considerable torque ripple in figure 3-17 compared to figure 3-12. As we can see, in every case the control system was still stable.

3.5 Controller Selection

3.5.1 Requirements

The microcontroller selection requirements were driven by the control system and the required interfaces. At the time of selection, the requirements were as follows:

- Controller processing speed
    - The controller must be able to run the control loop at 16 kHz
    - Floating point unit
- Controller peripherals
    - $>$ 3 sigma delta filters
    - $>$ 1 CAN interface
    - $>$ 2 SPI interface
    - $>$ 1 UART interface
    - $>$ 5 Analog inputs
    - $>$ 3 complimentary PWM ports with deadtime

Processors that met the above requirements were available from both TI and ST. However the ST provided two main benefits: the team was more familiar with the STM32 series, and is ARM based, which is industry standard, while the TI C2000 series is not.

3.6 Resolver Offset Determination

The resolver is mechanically mounted to the motor. The signal it returns corresponds to the angle between the resolver's rotor and start, however the angle required for the control system is the angle of the magnetic axis of the motor. These two angles are separated by a constant offset due to the mounting of the resolver.

3.6.1 Back EMF Observer

There are two main ways to determine the magnetic angle of the motor. The first is based on the saliency of the motor, and the second is based on the back EMF. As part of a strech goal, a back EMF observer was designed and tested in Simulink.

Saliency is a measure of how magnetically round a motor is. The inductance of a motor will vary through a complete rotation. High frequency injection takes advantage of this fact by estimating the impedance of the motor as it completes a rotation by adding a high frequency signal on top of the low frequency fundamental signal. The perfomance of this method depends on the saliency ratio of the motor. However, the saliency ratio of the EMRAX 228 motor is only 0.04, which would make such a measurement difficult.

The more direct method of finding the magnetic angle of the motor is using the back EMF. This requires the motor to be spinning at a sufficient speed in order to generate a sufficient voltage (1000 rpm for the EMRAX 228). If the motor is not being driven with the controller, this can be directly measured. However, if there is current flowing in the phases between the motor and controller, then it is not directly measured and must be estimated.

The basic equation describing the system can be found in equation (3.14). V_{emf} is the back EMF of the motor, V_{ph} is the phase voltage at the terminals of the motor, I_{ph} is the phase currents, R_s is the stator resistance, and L_s is the average stator inductance, where $L_s = \frac{L_d+L_q}{2}$. By using the phase voltage (known from the control system) and the phase currents (measured), the back EMF can be directly calculated. However, this requires the numerical calculation of the derivative of the current, which is problematic as such a calculation would amplify the noise of an already noisy signal. Instead, equation (3.15) is used. A PI controller is used to drive the steady state error to zero.

$$V_{emf} = V_{ph} - I_{ph}R_s + L_s\frac{dI_{ph}}{dt} \tag{3.14}$$

$$\frac{V_{emf} - V_{ph}}{(R_s + sL_s)} + I_{ph} = 0 \tag{3.15}$$

For faster computation, the controller is implemented in the $\alpha\beta$-reference frame. This requires only computing two controllers, instead of one for each phase. The equations for the α and β back EMFs can be

found in equation (3.16) and equation (3.17) respectively. The control system block diagram for this method can be found in figure 3-18. Finally, the simulation results can be found in figure 3-19.

$$V_{EMF,\alpha} = \psi_m \omega_e(-cos\theta) \tag{3.16}$$

$$V_{EMF,\beta} = \psi_m \omega_e(-sin\theta) \tag{3.17}$$

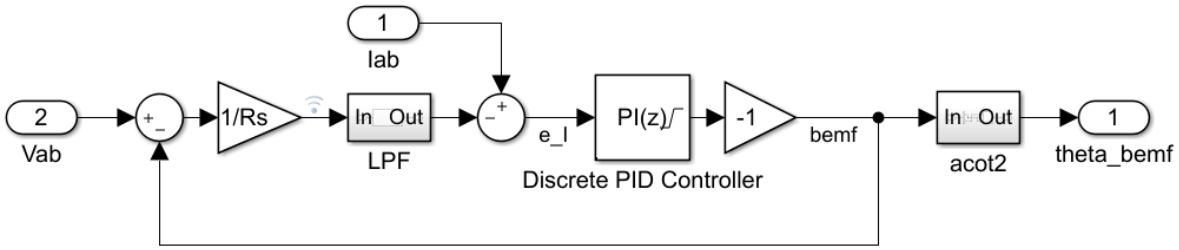


Figure 3-18: back EMF observer control system diagram

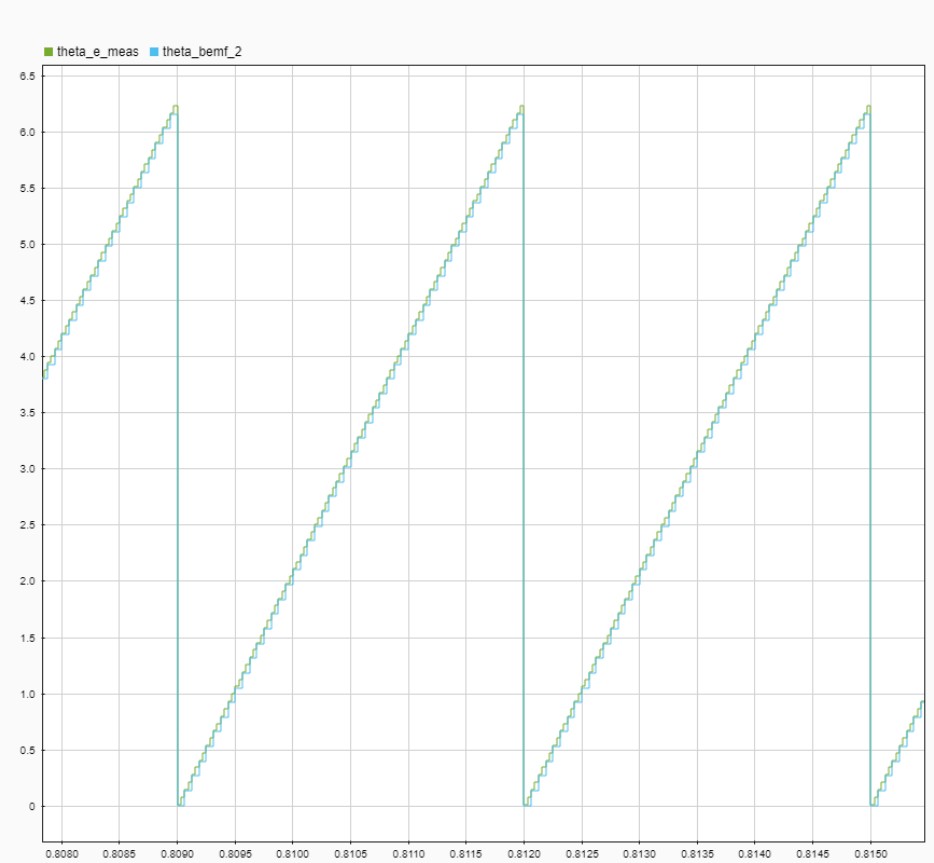


Figure 3-19: Comparison of actual angle and estimated angle from back EMF observer

Chapter 4

Hardware Design

The motor controller has two custom four-layer PCBs (the control PCB and the gate driver PCB), a two layer PCB, hall effect current sensors, an IGBT module, and a DC link capacitor bank.

While the project focused on the electrical design of a motor controller, it was impossible to ignore the mechanical constraints. The PCBs, IGBT module, and bus capacitors had to fit together. An image of the prototype can be found in figure 4-1. As can be seen in figure 4-1, the two PCBs are stacked on top of each other. This layout was chosen as it was the most efficient use of space, and keep the signals between the two PCBs short. The bus capacitors were also located as close as possible to the IGBT DC inputs, in order to minimize line inductance.

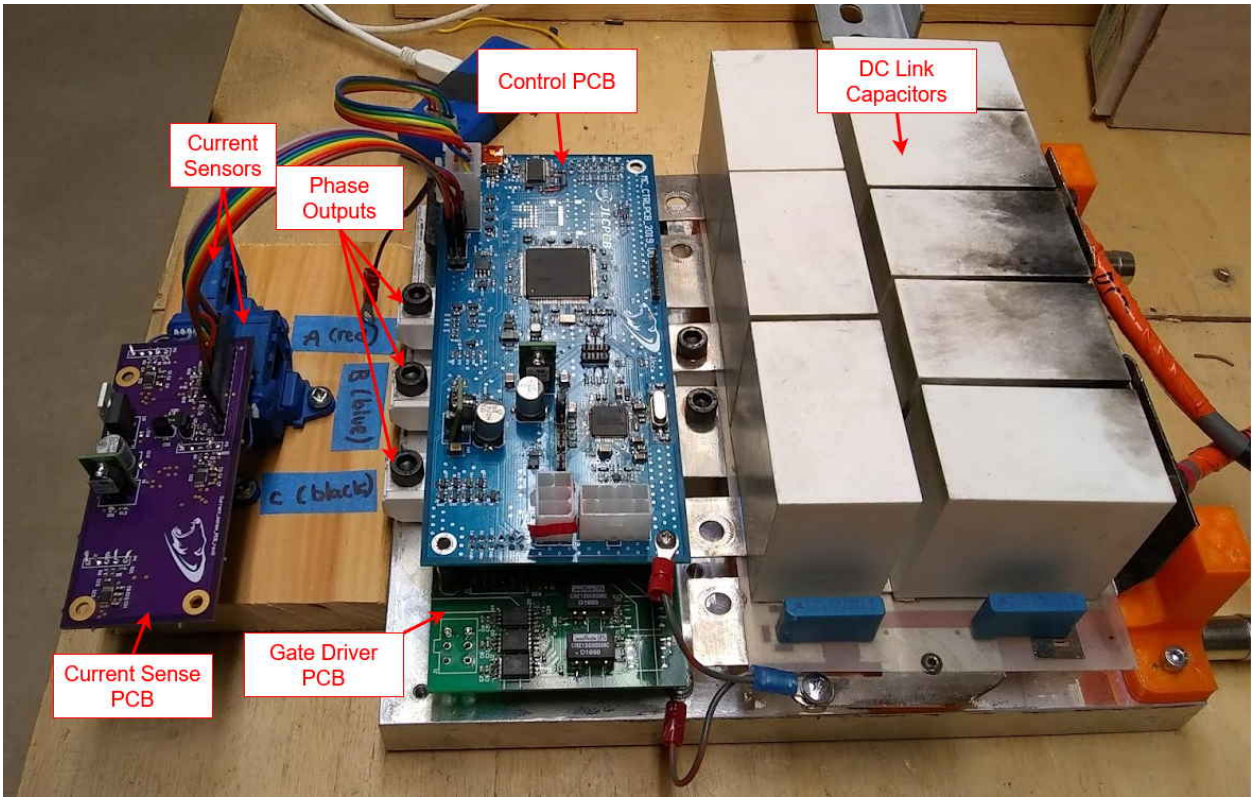


Figure 4-1: Assembled motor controller prototype complete with water cooled heatsink and 3D printed plastic supports

4.1 Control PCB

4.1.1 Overview

The Control PCB housed an STM32F767ZE microcontroller that runs the control system of the motor controller as well as communicate with the vehicle control unit and GUI. A block diagram of the Control PCB can be seen in figure 4-2.

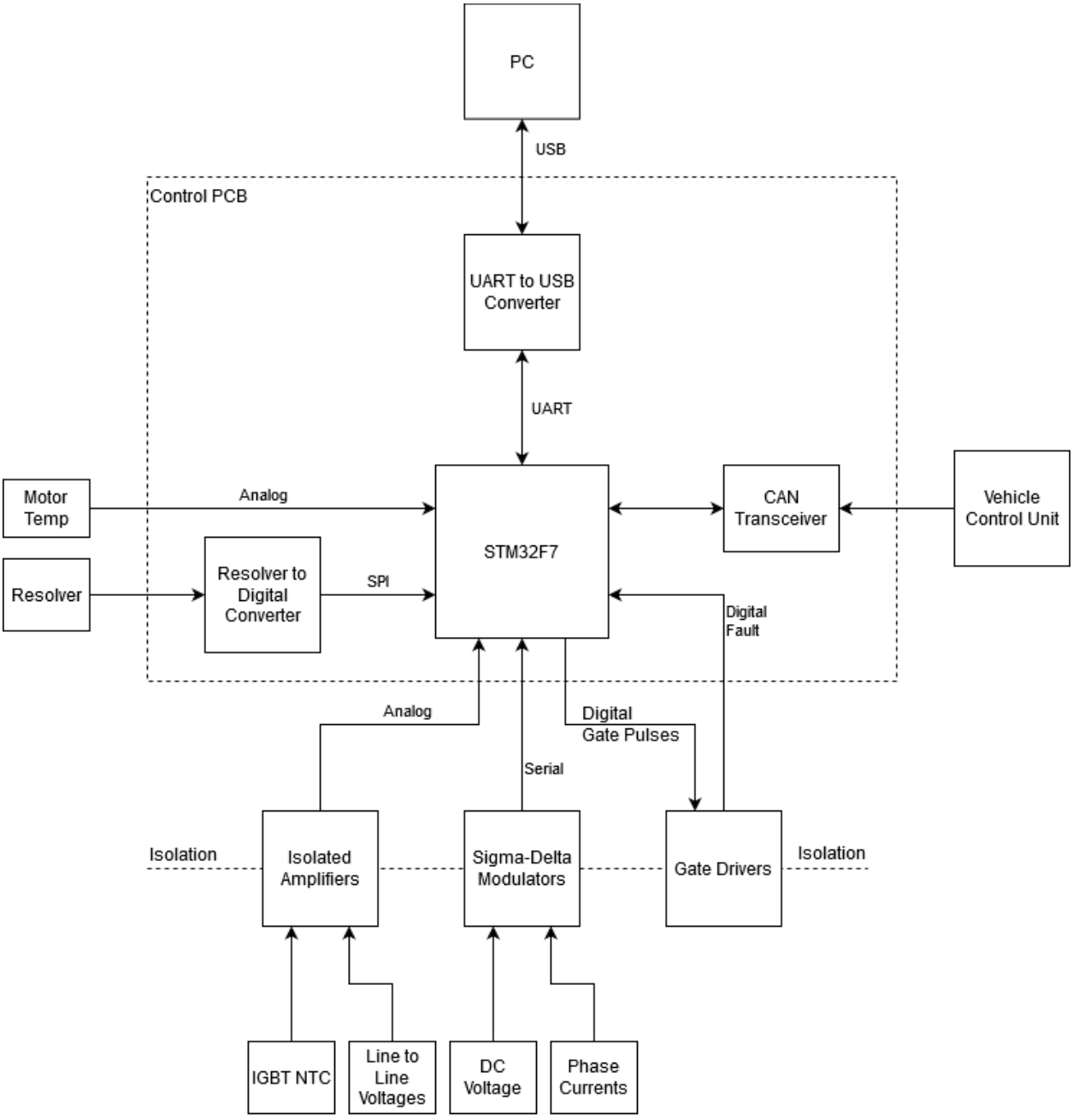


Figure 4-2: Block diagram of the control PCB

4.1.2 Considerations

In order to reduce noise on the PCB, a four-layer board was used. The top and bottom layers were ground, in order to shield the signals on the inner layers from EMI. In order to provide flexibility for additions in the prototyping stage, spare microcontroller pins and debugging ports were routed to easily accessible headers. Reverse polarity protection was added on the main power input to prevent damage to the device if the source was plugged in backwards. An image of the populated PCB can be found in figure 4-3.

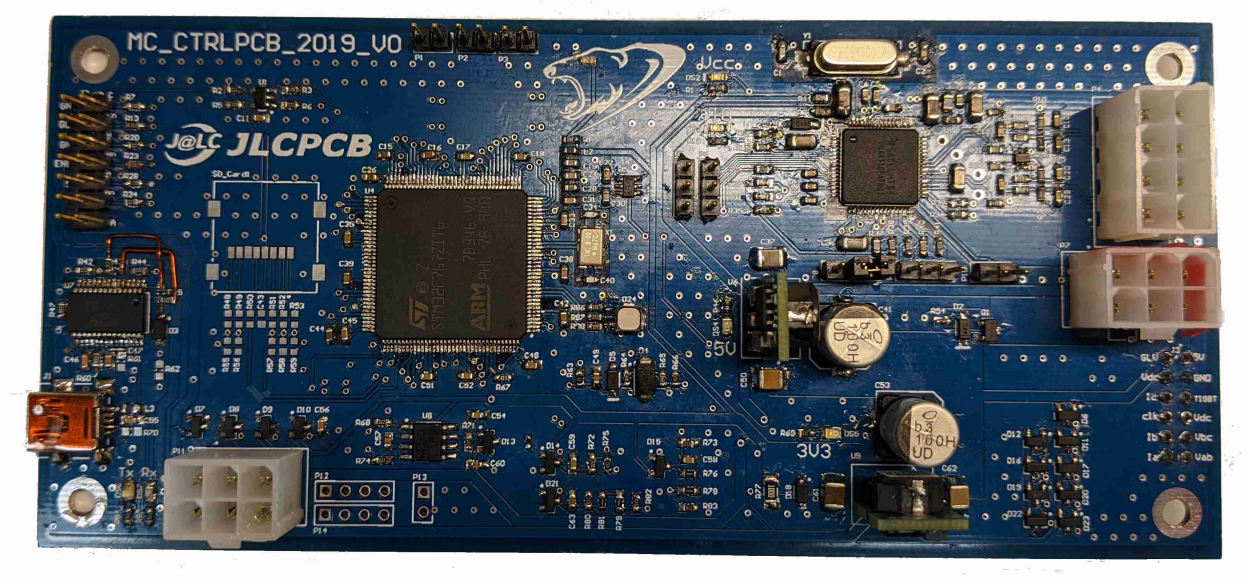


Figure 4-3: Image of the manufactured control PCB

4.2 Gate Driver PCB

4.2.1 Overview

The gate driver PCB contained the gate driver circuitry, the current and voltage measurement circuitry, and was directly soldered to the IGBT module. A block diagram showing all the interfaces can be found in figure 4-4.

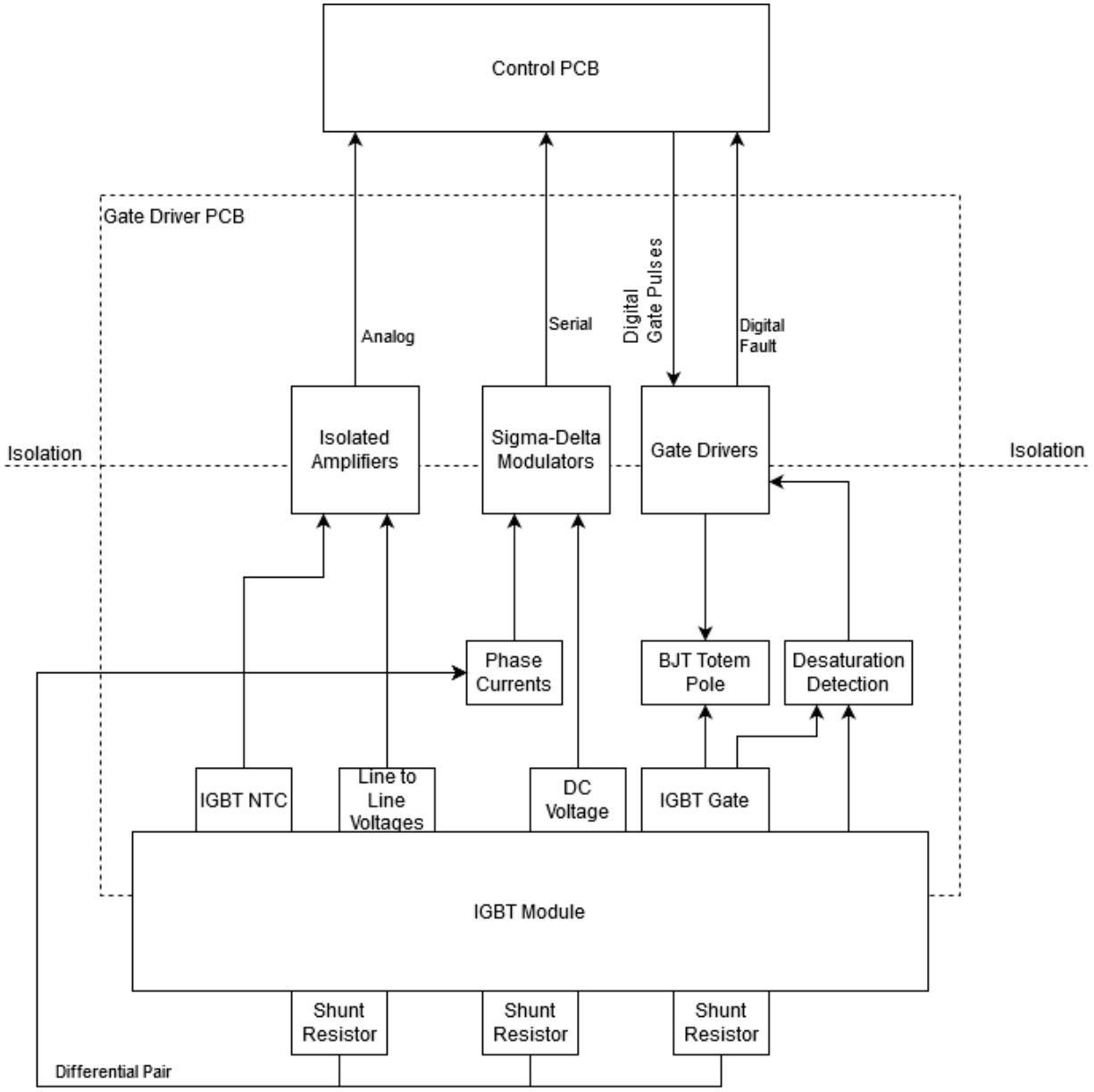


Figure 4-4: Block diagram of the gate driver PCB

4.2.2 Considerations

In order to reduce noise and improve packaging, a four-layer board was used. The PCB was designed to fit within the footprint of the IGBT module. In order to minimize gate inductance, the gate drivers were placed as close as possible to the IGBT's gates. Altium net classes were utilized in order to maintain clearance requirements between the six TS classes and GLV nets. This was both functional and to meet FSAE rules requirements. The six TS net classes were nets referenced to DC+, DC-, the DC midpoint, and the three output phases. The separation between TS classes was 1.5mm, as recommend in [23]. An image of the populated PCB can be found in figure 4-5.

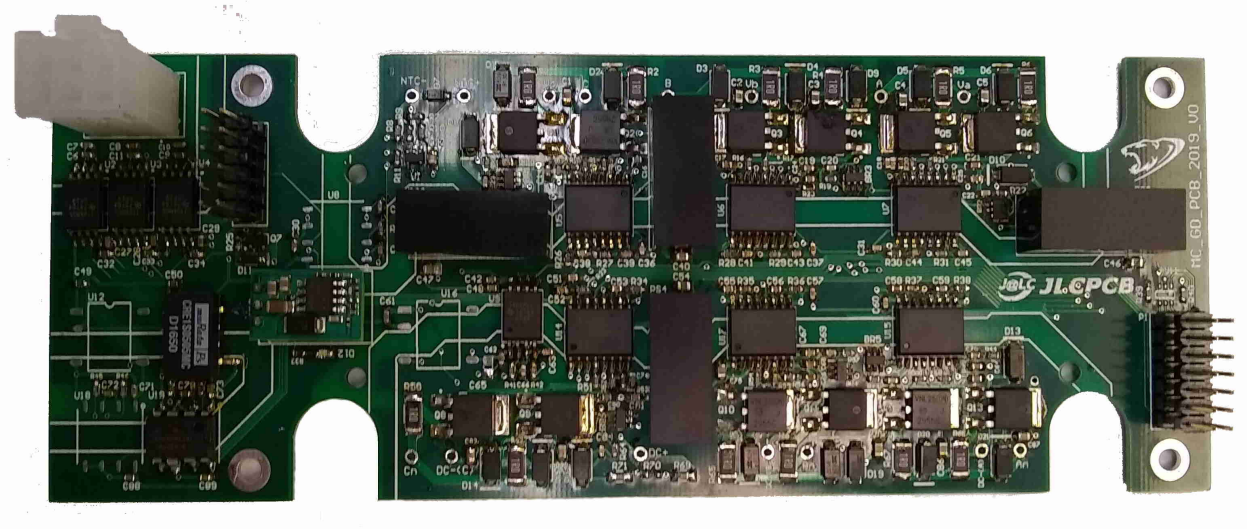


Figure 4-5: Image of the manufactured gate driver PCB

4.3 Current Sense PCB

4.3.1 Overview

In order to accommodate the hall effect current sensors, an additional PCB was designed. This housed external ADCs, and was located directly above the current sensors. The board communicated with the controller over SPI. A block diagram of the PCB can be found in figure 4-6.

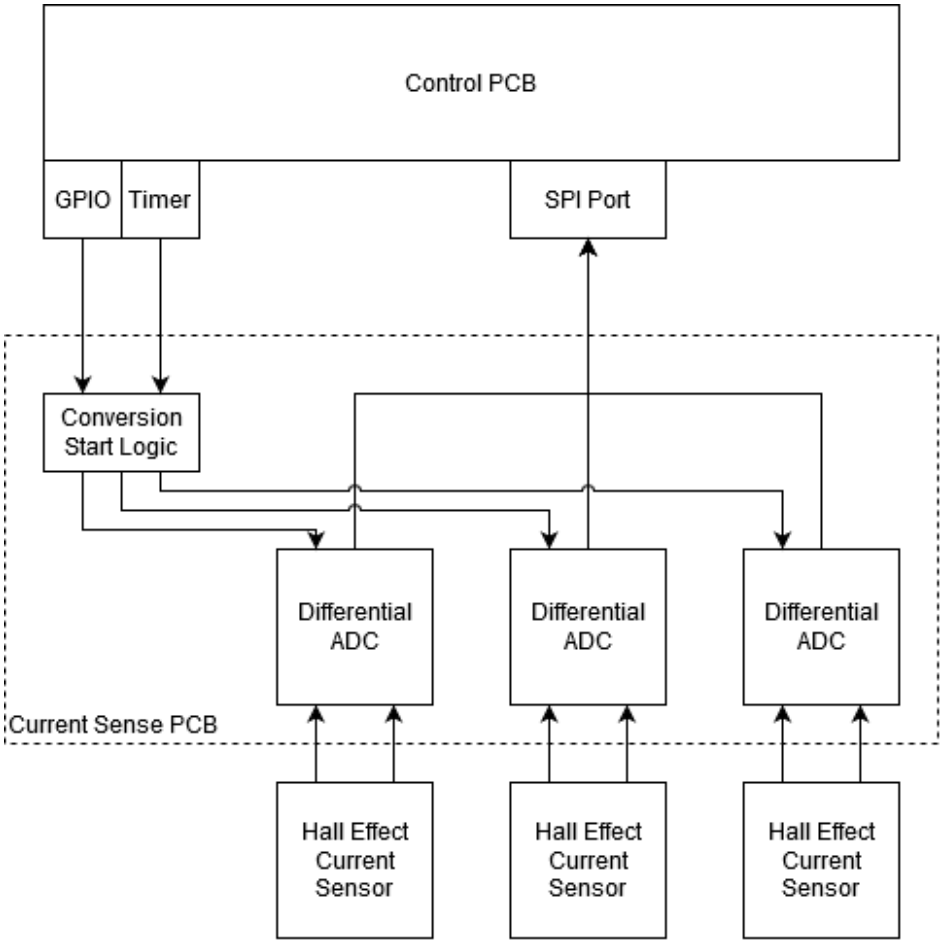


Figure 4-6: Block diagram of the current sense PCB

4.3.2 Considerations

Differential ADCs used in current measurements are located within 3 cm of the current sensor itself. An RC low-pass filter with a cutoff frequency equal to the switching frequency is used to reduce the noise generated from the IGBTs switching. A precision voltage reference helped ensure accurate ADC readings. Finally, the ADCs are triggered to start conversion at the start of the PWM clock cycle. This ensures as long as the controller is not outputting the maximum phase voltages, the measurements are taken at a time when none of the IGBTs are switching. An image of the manufactured PCB can be found in figure 4-7.

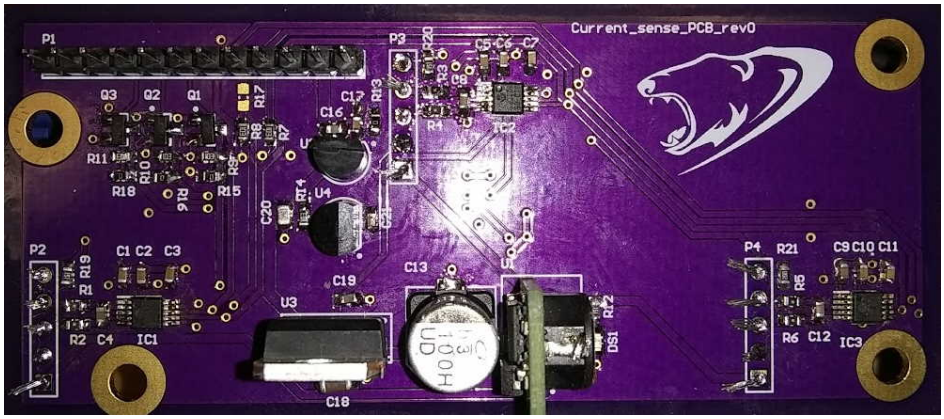


Figure 4-7: Image of the manufactured current sense PCB

Chapter 5

Embedded Systems

The embedded systems of the motor controller are carried out with the STM32F767ZE microcontroller. The software is written in embedded C using STM32Cube Hardware Abstraction Layer (HAL) libraries. While the HAL libraries are not as efficient, they were chosen due to their ease of use and reduced the time to write the motor controller's software.

The main system clock of the microcontroller is the maximum clock speed of the STM32F7 microcontroller: 216 Mhz. This chosen speed allows the microcontroller to run the main part of the control loop at 16 kHz as well as allow the microcontroller to measure variables for the control loop almost instantaneously.

5.1 State Machine

The software for the motor controller was designed as a state machine with four different states: Initialization, Idle, Enabled, and Fault. An LED on the control PCB will be coloured according to what state the motor controller is in. The design of these states simplify the motor controller use as well as clarify to the user what state the motor controller is in. A diagram of these states and how the states transition to one another can be seen in figure 5-1.

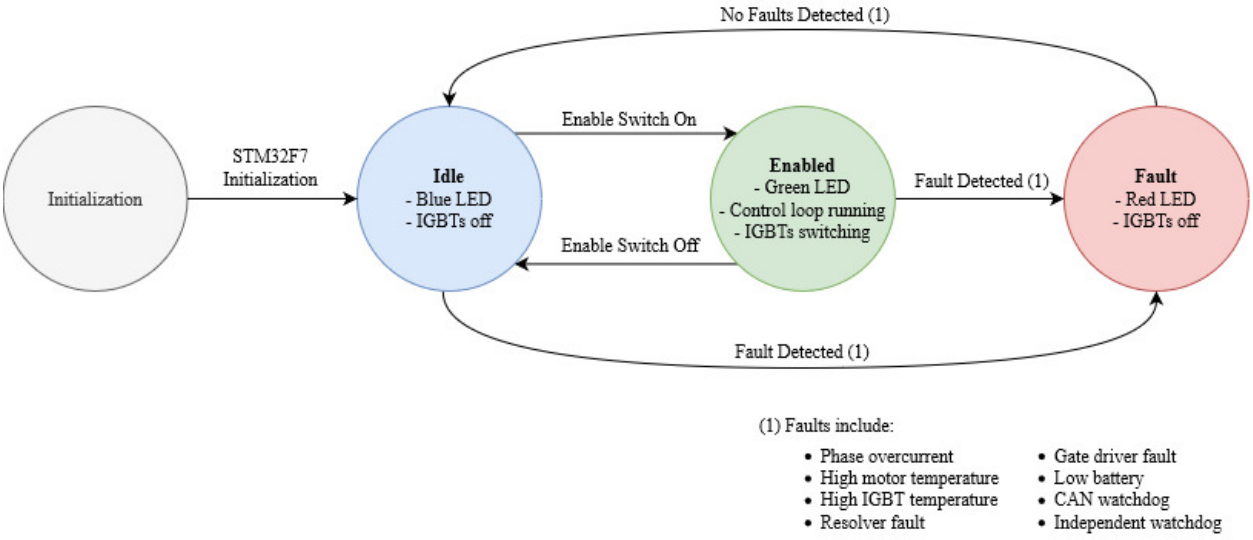


Figure 5-1: State machine diagram

5.2 Software Organization

The embedded software code is divided into two main parts: the main loop and the control loop interrupt.

5.2.1 Main Loop

The main loop runs the state machine and samples signals that aren't critical to the control systems. These functions are placed in the main loop since their timing is not as critical. The main loop also refreshes the independent watchdog. If software malfunctions, the independent watchdog will reset the controller and bring it to the idle state. The main loop flow chart can be seen in figure 5-2.

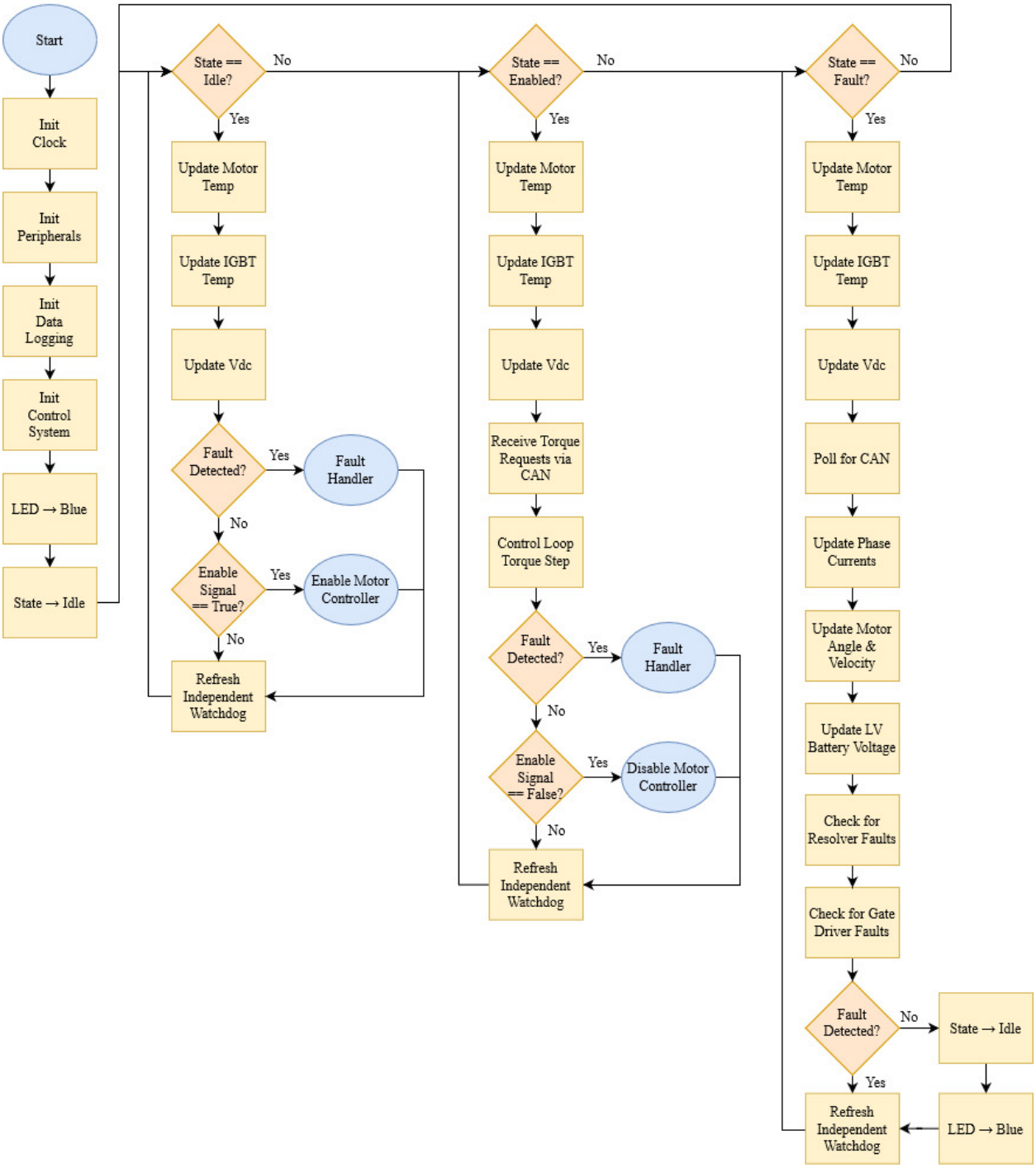


Figure 5-2: Main flow chart

5.2.2 Control Loop

The control loop is a timer interrupt with a frequency of 16kHz. The control loop will only interrupt when the motor controller is in the enabled state. The first task during the interrupt is to sample the variables required for the control loop, including the motor angle and velocity, phase currents, and DC voltage. Next, the interrupt runs through the main parts of the control loop: the current reference generator, the current controller, and the SVPWM generator. After this is completed, data is logged and transmitted to the GUI if it is connected. A flow chart of the control loop interrupt can be seen in figure 5-3.

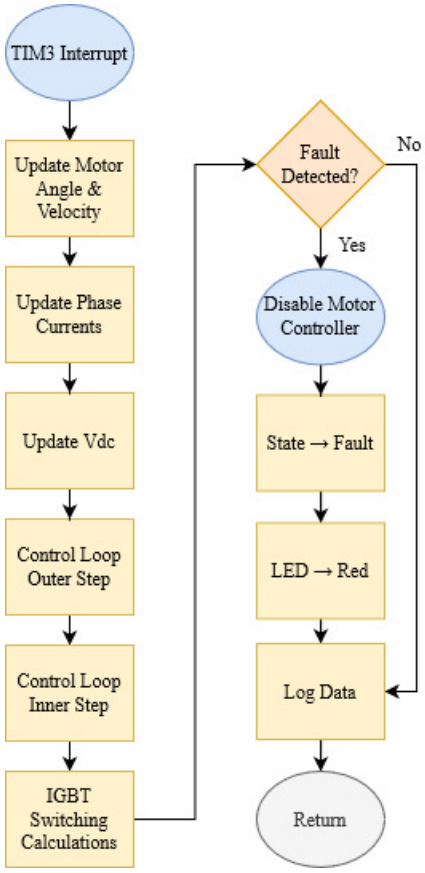


Figure 5-3: Control system flow chart

5.3 Control System Implementation

The control system designs and simulations were developed in Simulink and exported into C using Simulink's embedded encoder application. The exported code was refactored for readability and efficiency. This in-

cluded replacing doubles with floats and integers, and replacing sine and cosine functions with a function that interpolates sine and cosine lookup tables.

5.4 Peripherals

5.4.1 CAN

In order to make our project compatible with the University of Manitoba's SAE Formula Electric team, our motor controller receives torque requests using the same protocol: CAN. This has been tested using a previous vehicle control unit from the 2019 Formula Electric competition. The torque requests are sampled during the enabled state in the main loop at a bit rate of 500 kbits/second. Torque requests are 3 byte messages following the format described in figure 5-4.

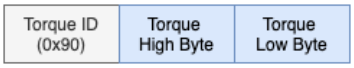


Figure 5-4: Torque request message format

5.4.2 Resolver IC

The microcontroller communicates with the resolver IC via SPI at a rate of 8 MHz. During initialization, the microcontroller writes to the resolver IC's registers to set an excitation wave of 7 Vrms at 10 kHz as required for the TS2620N21E11 resolver [26]. To read the motor angle and velocity from the resolver IC, the microcontroller reads the 12 bit angle value, theta_raw, and the 11 bit velocity value, velocity_raw, from the corresponding resolver IC registers. The electrical angle and velocity of the motor is calculated using equation (5.1) and equation (5.2) respectively, where pp is the number of pole pairs of the motor (10 for the EMRAX 228), and f_{clk} is the sampling frequency the resolver IC (which is 20MHz).

$$theta = \frac{(theta\ raw) \cdot 2\pi}{2^{12}} \cdot pp \tag{5.1}$$

$$velocity = 60 \cdot \frac{f_{clk}(velocity\ raw + 1)}{2^{25}} \cdot pp \tag{5.2}$$

5.4.3 DC Voltage Sensor

Sigma delta sensors were chosen to read the HV DC voltage and the voltage on the shunt resistors. The signals were processed by the microcontroller's built in sigma delta filters. The AMC1106 datasheet recommends the following filter [11] as having the best performance for a second order filter using the least amount of logic gates.

$$H[z] = (\frac{1-z^{-256}}{1-z^{-1}})^3$$

5.4.4 Current Sensors

Originally, sigma delta modulators read the current on the phase shunt resistors. However, during testing noise found on the sigma delta modulators was too large to use with the control system. Instead, HO 250-S hall effect current sensors were implemented [17]. An external differential ADCs is used to read the voltage of the hall effect current sensor [27]. The ADCs communicate with the microcontroller via SPI at a rate of 20 Mbits/s using a chip select to select which phase current to read. The 16 bit values are processed using equation (5.3), where *current raw* is the 16 bit signed ADC value and G is the gain of the HO 250-S sensor.

$$current = \frac{(current\ raw) \cdot 5V \cdot 2}{2^{16}} \cdot \frac{1}{G} \tag{5.3}$$

5.4.5 Temperature Sensors

There are two thermistors read by the microcontroller: the motor thermistor, and the IGBT thermistor. The motor's thermistor is a kty 81/210 temperature sensor [25]. This thermistor's resistance is almost linear with temperature, so the curve is approximated using two equations: one for the lower part of the curve and one for the upper part of the curve. Figure 5-5 shows the microcontroller voltage input as a function of temperature using the datasheet resistor values and the two linear equations.

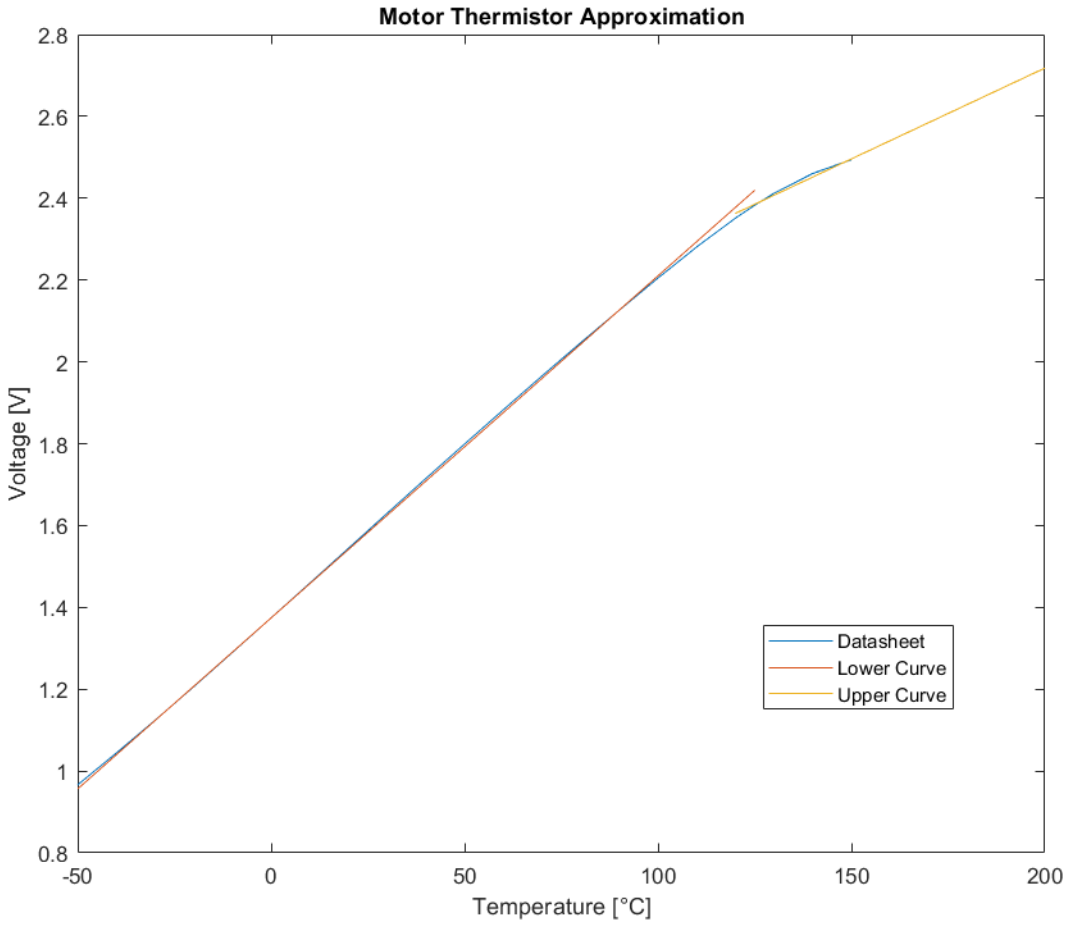


Figure 5-5: Motor Thermistor Approximation

The IGBT has an internal NTC thermistor with a temperature curve is described by equation (5.4), where T is the temperature of the NTC, R is the resistance of the NTC, T_0 is a known temperature and R_0 is the resistance at that temperature, and B is a measurement of how much the resistance changes with temperature. Since this equation is not linear, and the B value changes with temperature, a lookup table was implemented. Figure 5-6 shows the microcontroller voltage input as a function of temperature comparing equation (5.4) and the lookup table.

$$\frac{1}{T} = \frac{1}{T_0} + \frac{1}{B} ln(\frac{R}{R_0}) \tag{5.4}$$

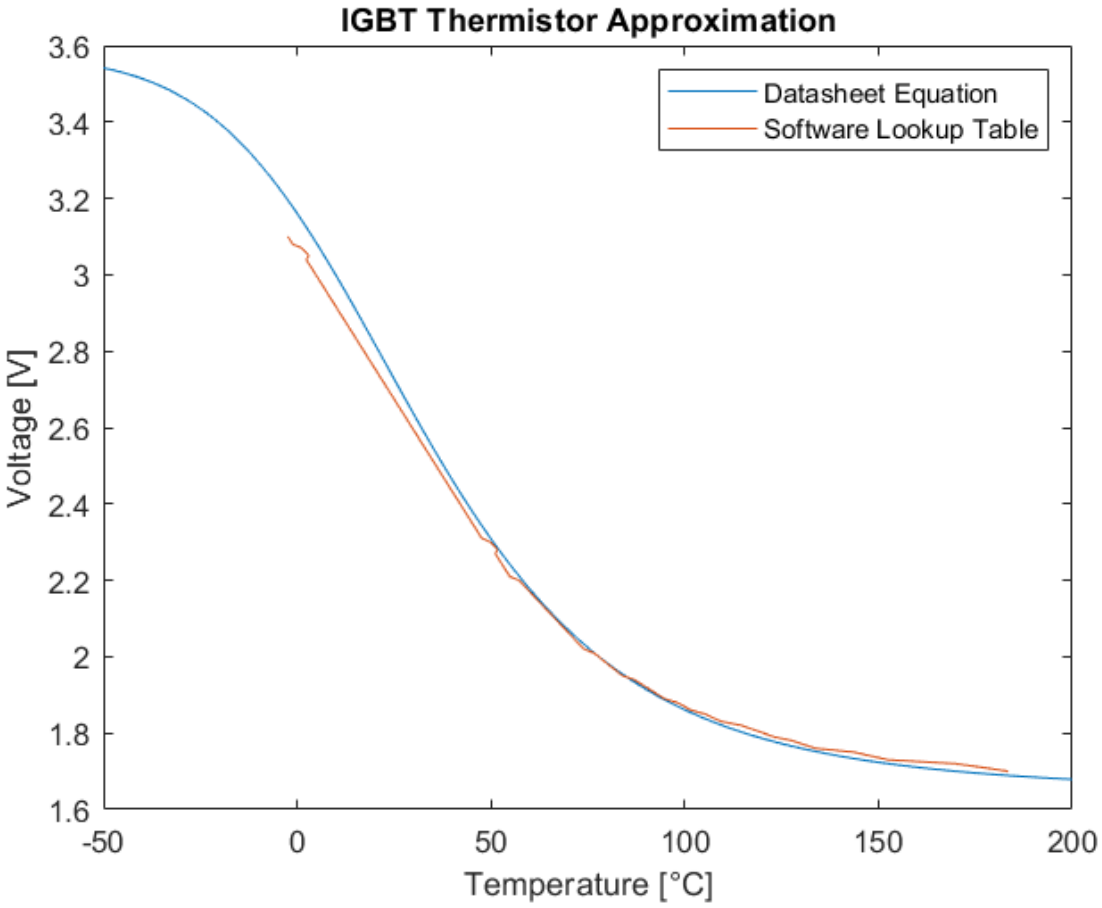


Figure 5-6: IGBT Thermistor Approximation

5.5 Flash Memory

The microcontroller has a 2MiB non-volatile Flash memory which is used to store both its software as well as the control system parameters. The Flash is organized into 12 sectors as described in figure 5-7.

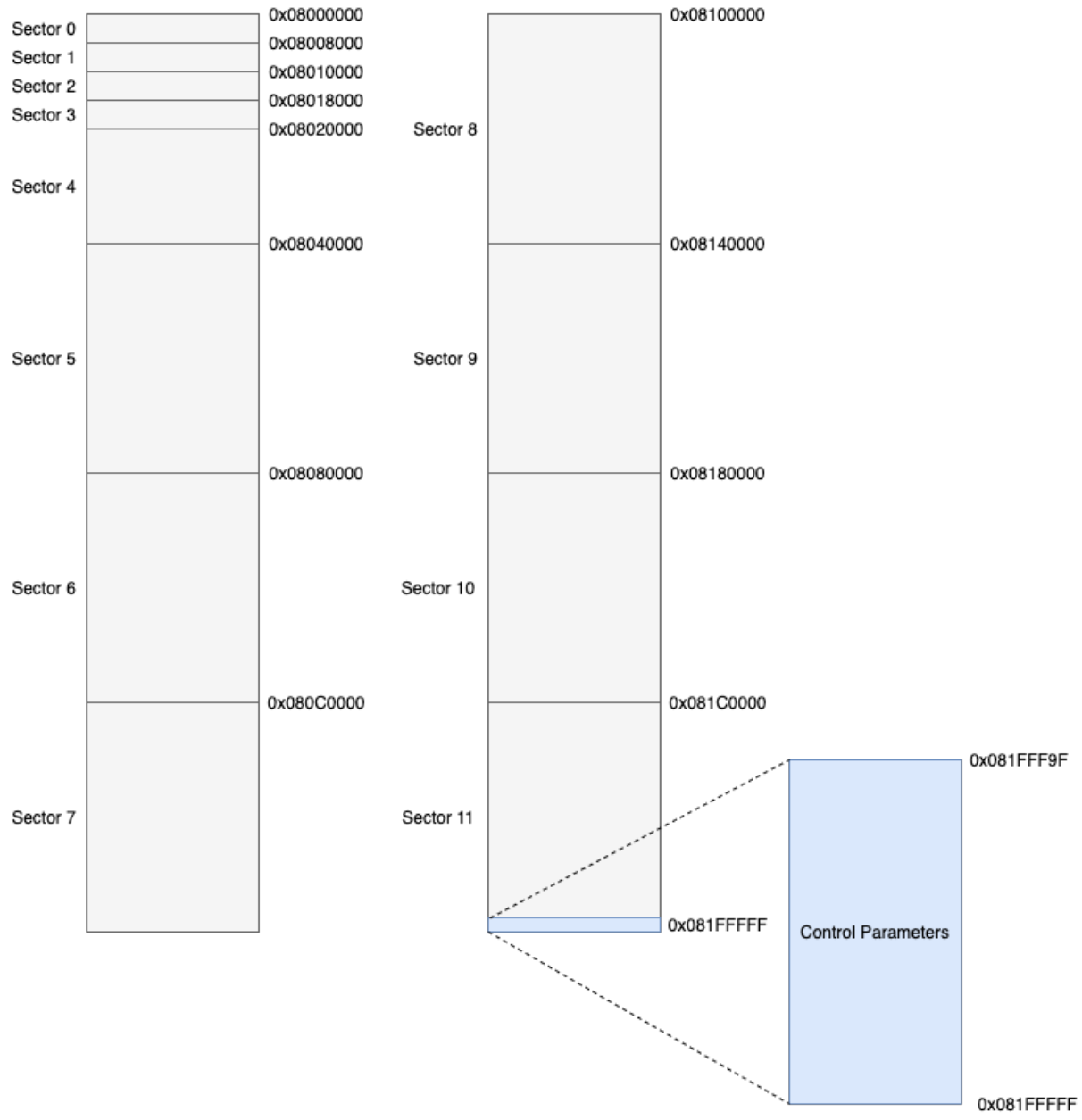


Figure 5-7: Organization of flash memory

The microcontroller's software is stored starting from the top of sector 0 and the control parameters are stored starting from the bottom of sector 11. The location of control parameters was chosen to allow for easy addition of additional parameters in the future.

Control parameters are stored as 32-bit values, therefore each require 4 bytes of memory. Currently 24

parameters are stored in Flash. Each parameter has a name, identifier and associated address as described in section 5.5.

Table 5-1: Control Parameters

Parameter Name	Parameter ID	Parameter Address
Pmax	0	0x081FFFFB
Tmax	1	0x081FFFF7
Ld	2	0x081FFFF3
Lq	3	0x081FFFEF
L0	4	0x081FFFEB
Rs	5	0x081FFFE7
psim	6	0x081FFFE3
p	7	0x081FFFDF
motorTemp_max	8	0x081FFFDB
motorTemp_corner	9	0x081FFFD7
rpm_max	10	0x081FFFD3
ke_ll_rpm	11	0x081FFFCF
fsw	12	0x081FFFCB
bw_idq_hz	13	0x081FFFC7
n_rpm	14	0x081FFFC3
n_tq	15	0x081FFFBF
t_ramp	16	0x081FFFBB
igbtTemp_max	17	0x081FFFB7
igbtTemp_corner	18	0x081FFFB3
wi	19	0x081FFFAF
wt	20	0x081FFFAB
Vhf_mag	21	0x081FFFA7
Tq_trigger	22	0x081FFFA3
rpm_trigger	23	0x081FFF9F

The parameter's ID and address are derived from one another, as each parameter is placed atop its

predecessor in Flash. As such, given an ID the address can be determined according to equation (5.5)

$$address = 0x081FFFFF - [(id+1)*4] \tag{5.5}$$

This scheme was chosen as it makes it simple to add additional parameters. By simply incrementing the ID, an address can be allocated.

5.6 UART

The UART module is responsible for communicating with a PC through one of the microcontroller's UARTs. Instead of using RS232 for our data-link—which would then be translated to USB to connect with a PC—the microcontroller and PC communicate using a USB to UART interface provided by the FT232R IC (datasheet available at [7]). This chip simplifies the circuitry as it was not required to translate from UART to RS232 to USB which would require stepping the signal levels up and down. This chip also lets the data-link operate at a baud rate of 921600 bps. Incoming messages are 7 bytes in length and adhere to the format described in figure 5-8

Figure 5-8: Incoming message format

The request byte refers to the action a user has requested. The payload is only used when requesting to write a control parameter into Flash memory and includes the 2 byte parameter ID (as described in section 5.5) and 4 byte parameter value. The full list of request bytes and requested actions is described in section 5.6.

Table 5-2: UART request bytes

Request Byte	Requested Action
0xEE	Erase the Flash memory
0x97	Program the Flash memory with the supplied payload
0x1F	Start high-frequency live logging
0x1E	Start low-frequency live logging
0x10	Cancel live logging
0x1D	Download the data log

The message format is primitive as it fulfills the requirements only of what was needed by the microcontroller to communicate with a PC. While a more flexible message protocol could be beneficial for performing more complex actions, it was not worth the extra development time.

The UART module also exposes the ability to transmit messages performed either in the standard blocking mode, or using DMA. When using DMA, a pointer to a callback function is also passed to the UART module to act as the transmission complete DMA interrupt function. This allows for different modules to send over DMA and perform different functions upon completion, e.g. to transmit more data or to change some system state.

5.7 Data Logging

The microcontroller logs many of the measured values during operation. These values are then sent to a PC application through the UART module. There exists three different modes of operation for the data log module as explained in section 5.7.

Table 5-3: Data log modes of operation

Mode	Active	Logging Frequency	Data Transmission
Standard	Always	16 kHz	When requested by user
High Frequency Live Log	When requested by user	16 kHz	Intermittently
Low Frequency Live Log	When requested by user	>1 kHz	Intermittently

As values are recorded they are stored in a C struct representing the current state of motor controller concerned variables. Once per control loop, a subset of these (19 variables) are then stored in another C struct representing a single log entry. This log entry is subsequently stored in a circular array of length 6000, as well as a smaller array of length 500 if high frequency live logging is enabled.

A single log entry consists of 19 floating point numbers, each represented by 4 bytes. The size of a full log entry is therefore 68 bytes. When logging in low frequency mode, only 15 of these variables are logged for a 60 byte entry. The logged fields, as well as their byte position within the data structure, is summarized in section 5.7.

Table 5-4: Mapping of byte position to data log field

Byte Positions	Data Log Field	High Frequency Logging	Low Frequency Logging
Byte[1:4]	id_ref	Yes	Yes
Byte[5:8]	iq_ref	Yes	Yes
Byte[9:12]	id	Yes	Yes
Byte[13:16]	iq	Yes	Yes
Byte[17:20]	Vdc	Yes	Yes
Byte[21:24]	vd_ref	Yes	Yes
Byte[25:28]	vq_ref	Yes	Yes
Byte[29:32]	TqRef	Yes	Yes
Byte[33:36]	TqLim	Yes	Yes
Byte[37:40]	TqRequest	Yes	Yes
Byte[41:44]	rpm	Yes	Yes
Byte[45:48]	motorTemp	Yes	Yes
Byte[49:52]	igbtTemp	Yes	Yes
Byte[53:56]	power	Yes	No
Byte[57:60]	faults	Yes	No
Byte[61:64]	ia	Yes	No
Byte[65:68]	ib	Yes	No
Byte[69:72]	ic	Yes	No
Byte[73:76]	theta	Yes	No
Byte[79:82]	vll_ab	Yes	No
Byte[83:86]	vll_bc	Yes	No

The log array sizes of 5000 and 500 were constrained by the available RAM of the microcontroller: 512kiB. Storing 5500 values at 76 bytes each equates to 408kiB—Roughly 80% of available RAM.

5.7.1 Standard Logging Mode

In the standard logging mode, a log entry is stored in a circular array every time the control loop runs. This continues for the duration that the control loop is active, or until one of two triggering conditions are met:

1. The requested torque surpasses Tq_trigger
2. The rpm surpasses rpm_trigger

Where the values of Tq_trigger and rpm_trigger are control parameters stored in Flash memory (section 5.5). Once a triggering event occurs, the log module continues for the next 2500 log entries—That is, half of the available data log. After this, standard logging concludes and the data log represents the state of the motor controller for 2500 log entries before and after the triggering event.

The user can request this data log from the PC application at which point the microcontroller will send log entries one at a time through the UART module to the PC.

5.7.2 High Frequency Live Logging Mode

In the high frequency live logging mode, a log entry is stored in the live log array every time the control loop runs. Once this array has been filled, it is sent through the UART module to the PC. Live logging temporarily suspends itself until all data has been sent through to the PC, at which point it begins again.

5.7.3 Low Frequency Live Logging Mode

In the low frequency live logging mode, a log entry is captured every time the control loop runs and immediately sent through the UART module to the PC. Live logging temporarily suspends itself until the data has been sent through to the PC, at which point it begins again.

A low frequency log message sent to the PC is 71 bytes in length. Sent at a rate of 921600 bps, this takes approximately 770µs for an approximate theoretical logging frequency of 1300 Hz.

Chapter 6

Graphical User Interface

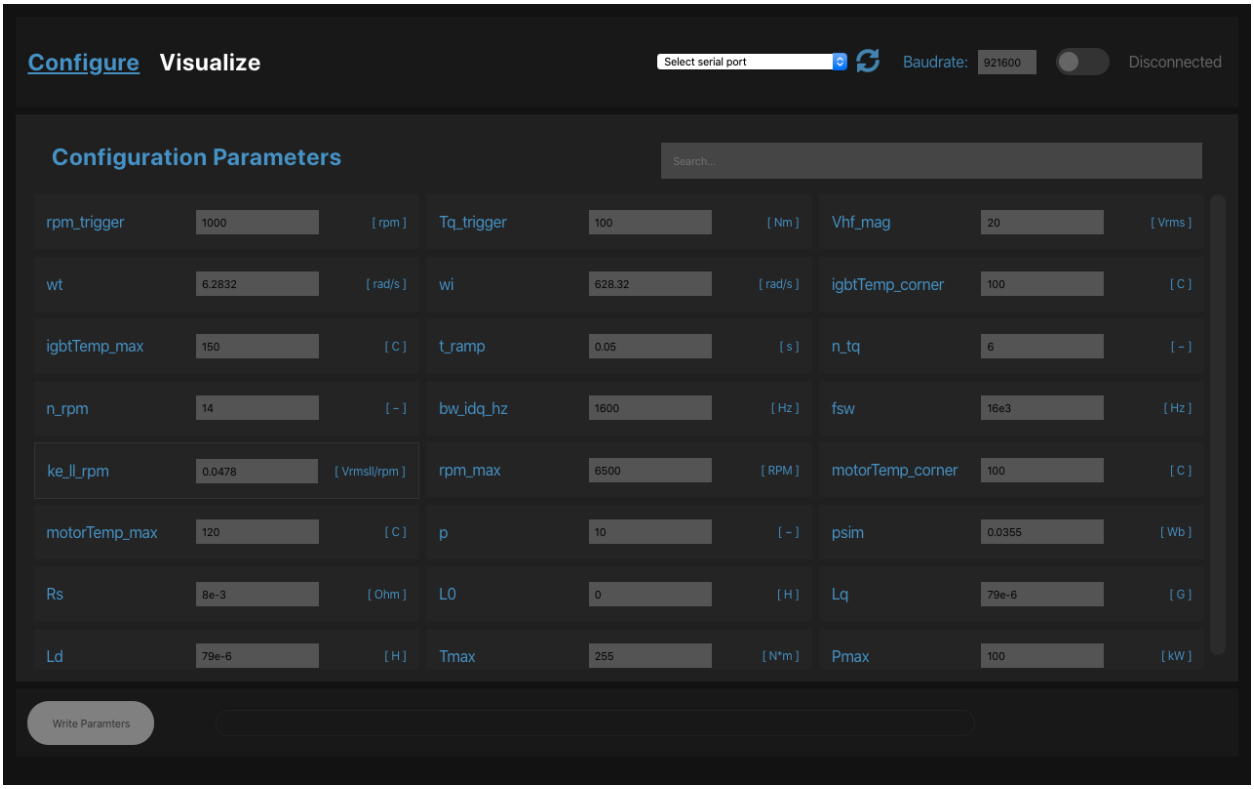


Figure 6-1: Configuration Module

Figure 6-2: Visualization Module

An **Electron** App built in **TypeScript** with **React** was developed. The two modes of operation for the GUI—Configuration and Visualization—can be seen in figure 6-1 and figure 6-2. The application allowed users to connect to the motor controller over a serial-link to configure the on-board control parameters of the system as well as download, view and analyze data logs.

6.1 Frameworks and Libraries

6.1.1 Requirements

The requirements for the GUI were as follows:

- Cross-platform support
- Ability to change control system gains over interface in less than 5 minutes
- Ability to receive data from motor-controller and provide the following functions

- Data visualization
- Export data as a CSV file.

6.1.2 Selection

Many frameworks exist for building cross-platform GUI applications such as Electron, QT and JavaFX. Of these, Electron was selected as a primary candidate. While there is both a computational and memory overhead with using Electron—in part due to its use of the Chromium rendering engine—the team's familiarity with JavaScript libraries, web technologies and programming practices meant that a professional looking and performant GUI could be built within the time-constraints in place. Additionally, it was decided to use TypeScript to enhance JavaScript with static typing to aid in the development process.

With the framework selected, JavaScript libraries were then chosen to meet the requirements for the application, as well as to aid in the interface design itself. The following libraries were selected:

- React
- Styled-Components
- SerialPort
- D3.js

React was chosen as the core library for building the GUI itself. It is a very powerful and established library that the team has experience with and as such would allow for ease of developing the various required components, as well as the data flow between related components and those component's internal state management.

Styled-Components was chosen to aid in the modular styling of GUI components. While it serves no functional purpose in the application itself, it aids in simplifying the development process and styling of the interface.

SerialPort was selected to satisfy the requirement of communicating with the motor controller. It allows for the application to use the host's serial ports to transfer data with the motor controller over a USB port.

D3.js was used to build data visualizations to meet the requirements of the project. It allows for dynamic

visualizations which benefits our requirements of live data-logging capabilities.

The organization of the application modules, along with the frameworks and libraries used, can be seen below in figure 6-3

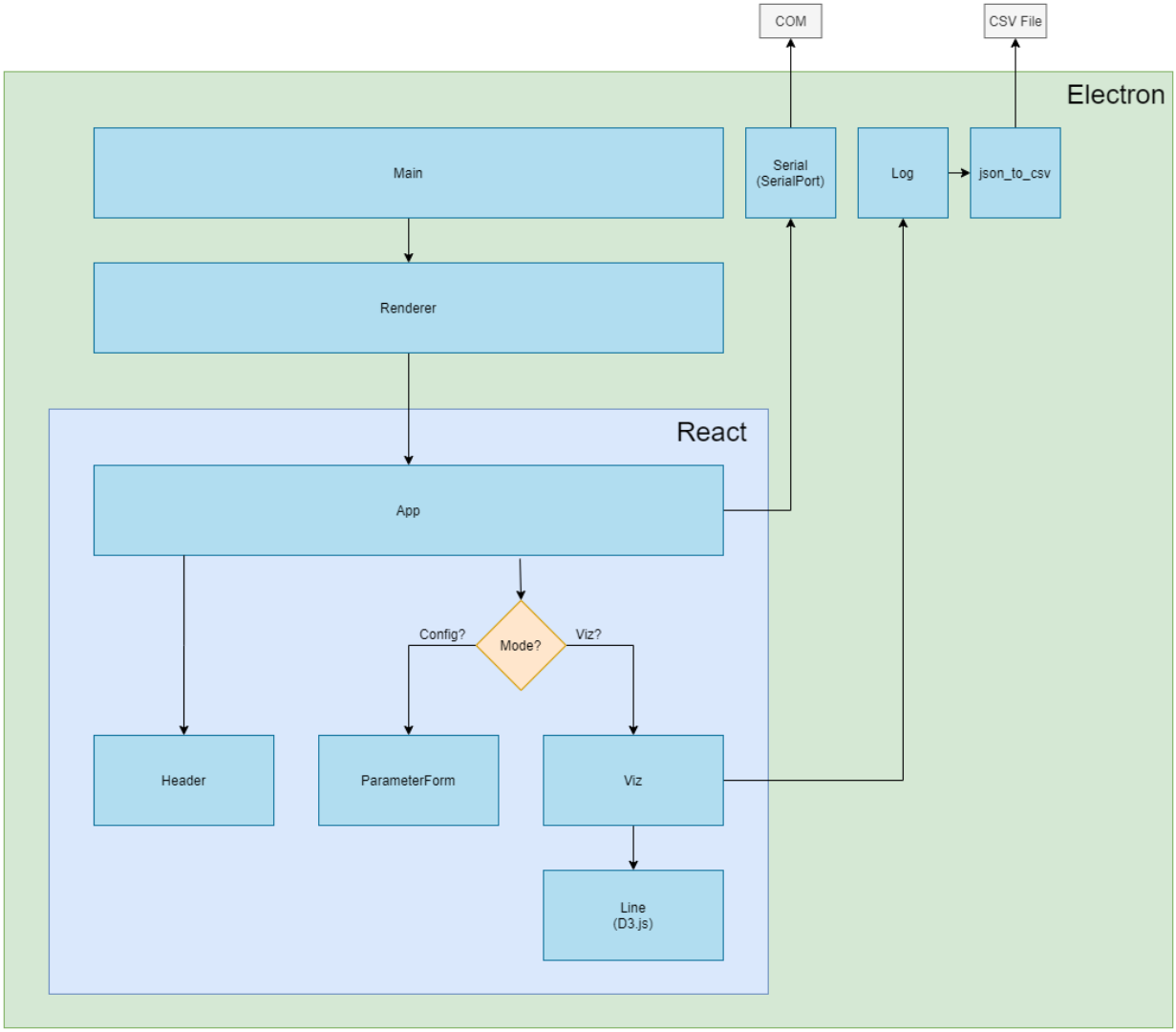


Figure 6-3: Application organization

6.2 Configuration Module

The configuration module—called “Configure” within the GUI—allows the user to configure motor controller parameters and transmit them to the motor controller to be written into flash memory.

The user configures parameters by filling out a simple form comprising of a variety of inputs corresponding each to a specific control parameter. If all inputs are validated successfully, the application sends a series of messages to the motor controller following the procedure described in figure 6-4. The format of messages sent to the motor controller is described in section 5.6.

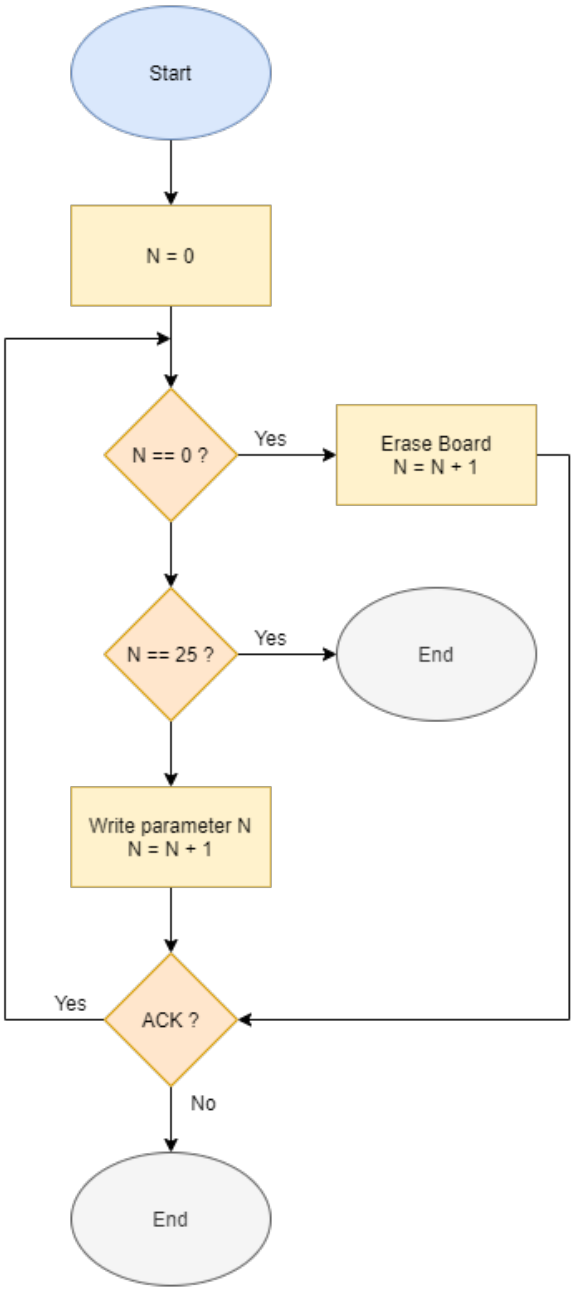


Figure 6-4: Configuration of motor controller

In figure 6-4, N refers to the current step in the parameter writing process. Upon receiving an acknowledgement N is incremented. If no acknowledgement is received in 15 seconds, the writing process is cancelled due to timeout error. N is used as both the index of the parameter to send, as well as to update a progress bar to provide feedback to the user. A superficial time delay is injected after each step of the process in order to give the user time to acknowledge the changing status text indicating what action is currently being performed by the system, as well as to give the progress bar time to animate.

The configurable parameters themselves are stored in the application as an array of objects (key-value pairs). The index of each object is the parameter's ID (as discussed in section 5.5). The objects themselves are defined as follows:

Table 6-1: Description of parameter structure within software

Key	Description of Value
name	The name of the parameter
description	The description of the parameter
type	'int' or 'float' for either 32-bit integer or floating point parameters
unit	The physical units of the parameter
value	The current value of the parameter in the form
validation	A function used to validate the user's input
valid	The validity of the user's input

The collection of objects defined in section 6.2 are then mapped to a collection of user input components as seen in figure 6-1.

6.3 Data Visualization Module

The Data Visualization Module—called "Visualize" within the GUI—allows the user to download data logs from the motor controller to visualize using the D3.js library, as well as tell the motor controller to enable live data logging.

6.3.1 Log Module

When live logging is enabled, or when the user chooses to download a data log, messages will be periodically received by the application from the motor controller. Upon data reception, the application determines the class of message and acts according to figure 6-5.

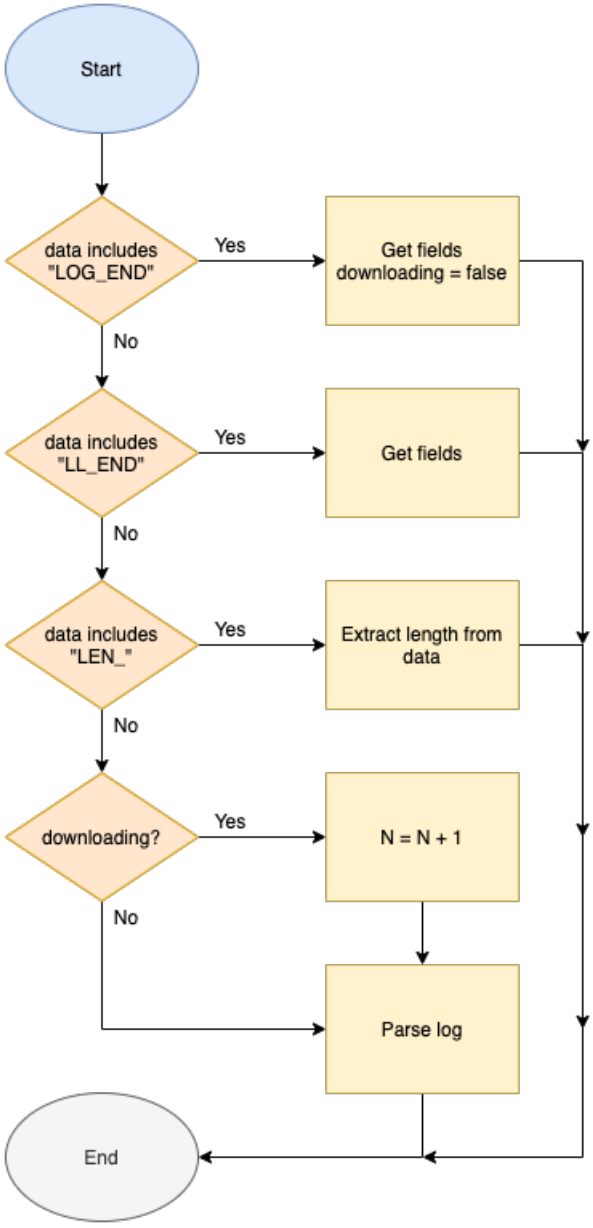


Figure 6-5: Log message reception procedure

When receiving a end of log message—"LOG_END" or "LL_END"—all available data fields are fetched from the internal log module and displayed to the user for selection. A length message tells the application the length of the log to be downloaded, this is used to display a progress bar for user feedback. Regular log messages are parsed and stored by the log module. When parsing log messages, first the log module verifies that the first byte of the data payload is 0x11. If so, the log module loops over the payload extracting 4 bytes at a time and converts them each to a 32-bit floating point value. The log entries are stored in a key-value collection. The keys are the data fields and the values are arrays of the converted floating point values. The data field is derived from the bytes' positions in the data payload as described in section 5.7 from section 5.7.

6.3.2 Line Module

When data is available to view, the available fields will be displayed to the user. The fields may be clicked on to commence visualization. Selecting a field fetches the most recent 6000 data points from the log module. Fetching a subset of the data was a design choice to limit the amount of data loaded into the GUI at a time to improve performance. This data—an array of floating point values—is then passed to the Line module for visualization.

The line module utilizes D3.js to place data points and draw lines between those points, additionally it also generates the axes and colours the data points. Of the 6000 data points passed to the line module, a window of 500 data points is visualized at a time. Only visualizing 500 data points at a time was a design choice to enhance the visibility and reduce overall visual noise. This windowing can be seen in figure 6-6.

Figure 6-6: Visualization window on data log

The Line module keeps track of an offset—a value between 0 and 500 subtracted from the total data length. When the user scrolls on the visualization window the offset is increased (or decreased) by an amount derived from the scroll wheel delta. The wheel delta is a value provided by the **DOM** scroll event and corresponds to the user's actuated scroll distance.

The 500 data point window is then used to draw the line chart itself. The chart's x-axis ranges from the offset value to the offset value plus 500. The chart's y-axis ranges from the minimum value found in the 6000 point dataset to the maximum value. The height and width of the visualization component in the GUI are provided to D3.js and the data points are drawn according to the mapping in equation (6.1)

$$(Index + offset, Value) \mapsto (XPixel, YPixel) \tag{6.1}$$

The index refers to the data point's index within the array of data values in the derived window. The mapping itself it handled by D3.js as it knows the range of the x and y axes as well as the pixel width and height of visualization component.

Points are also given colour using a linear colour gradient between the RGB Hex colours 0x464BB4 and 0xB4464B. This roughly corresponds to a colour gradient between blue and red, however the specific colours were chosen to better match the GUI's colour scheme while still conveying information. The colours are determined according to the mapping: $Value \mapsto Colour$

The value is the data point's numeric value using the same range as the y-axis. In this mapping, the absolute minimum value would be given the colour 0x464BB4 (blue) while the absolute maximum would be given the colour 0xB4464B (red). Values in between follow a linear mapping between the two extremes.

Finally, a curve is drawn through all placed data-points. The JavaScript code to generate the line chart can be seen below. The resulting line chart may be seen in figure 6-2.

Listing 6.1: Drawing a line chart with **D3.js**

```
1  const x = d3.scaleLinear()
2  .domain([offset, offset + 500])
3  .range([0, width]);
4
5  const y = d3.scaleLinear()
6  .domain([d3.min(data) || 0, d3.max(data) || 0])
7  .range([height, 0]);
```

```
 8
 9  const color = d3.scaleLinear<string, string>()
10  .range(['#464bb4', '#b4464b'])
11  .domain([d3.min(data) || 0, d3.max(data) || 0])
12
13  const lineChart = d3.line()
14  .x((d,i) => x(i + offset))
15  .y((d) => y(d))
16  .curve(d3.curveMonotoneX)
```

6.3.3 Exporting Data Logs

The final functionality provided by the Visualize module is the ability to export data logs as a CSV file. This is done by the json_to_csv module. JSON is JavaScript Object Notation, the format of the key-value collections used within the application.

CSV files consist of a rows of values separated by delimiters. Commas were used to delimit values and newline characters were used to delimit rows. These characters were chosen as they are standard in CSV files. The json_to_csv module first looped over each field present in the data log and wrote each to a file in the first row which corresponds to the header row—this labeled each column. Next, the first value of each field's corresponding data log was written to the following row, followed by the second value of each data log in the subsequently following row. This was repeated until all data log values were written. If ever a data log had run out of values and could not write to the currently examined row, a '0' was written in its place.

Due to the size of the data logs being operated on, this processing and file system IO could take upwards of 5 seconds or longer. Initially this caused the GUI to block and become unresponsive. To combat this, the processing was moved to a worker thread and performed in parallel with the main GUI thread.

6.4 Serial Module

The serial module is in charge of managing serial communication for the application using the SerialPort library. The main functions provided by the serial module are as follows:

- Listing available serial ports
- Connecting to and disconnecting from serial ports
- Sending outgoing messages
- Handling incoming messages

The user can view a list of available serial ports and select one to initiate a connection with a user-configured baud rate. The status of this connection is exposed to the user as seen in figures 6-1 and 6-2.

The serial module exposes the ability to attach data handlers to handle incoming messages. A message is any sequence of data (a byte array) terminated by the bytes '0x21, 0x21, 0x21, 0x21', or '!!!!' in ascii. This termination sequence was chosen initially to be simply the single byte '0x21', however it was increased arbitrarily to four repeated bytes in order to minimize the risk of false terminations in regular data. Upon receiving a message the serial module passes the incoming message to all attached data handlers. When communication is required with the Motor controller, both the Configure and Visualize modes of the application attach their data handlers with the serial module and then subsequently detach them when communication has completed.

Chapter 7

Faults & Failures

7.1 FMEA

A Failure Mode and Effects Analysis (FMEA) was conducted to identify potential failures in the operation of the motor controller. The intent was to protect both the motor controller and connected devices. The completed FMEA can be found in figure 7-1.

Item	Function	Failure Mode	Failure Cause	Failure Effect	Failure Detection	Failure Handling
Resolver	Senses motor position for control system.	Motor position resolver failed.	Wiring or resolver damaged.	The controller will be unable to correctly drive the motor, potentially resulting in IGBT damage.	The resolver IC detects faults such as open circuit and short circuits. The main controller polls the resolver IC's fault registers to check for these conditions. If the main controller reads a fault in one of these registers, it raises a resolver fault flag. The main controller also calculate the CRC during SPI communication. If the calculated CRC doesn't match the recieved CRC, then the main controller raises a CRC fault flag.	Upon failure detection, the microcontroller will enter a fault state and turn on a red LED. If the previous state was the enable state, the microcontroller will disable PWM outputs to the gate drivers shutting down power to the motor.
IGBT	Semiconductor switches used to convert DC to AC for motor.	Device is short circuited.	Internal IGBT failure, or outputs short circuited.	This will damaged the IGBTs.	The gate driver ICs have short circuit detection capability. This signal will be read by the main controller, which will turn on a fault LED.	Upon failure detection, the microcontroller will enter a fault state and turn on a red LED. If the previous state was the enable state, the microcontroller will disable PWM outputs to the gate drivers shutting down power to the motor.
CAN	Used to send torque commands to motor controller.	Motor controller stops receiving new commands.	CAN wiring is damaged or disconnected.	Motor controller may still be driving motor even though the vehicle should be stopping.	The controller has been programmed to measure the time in between CAN messages.	Upon failure detection, the microcontroller will enter a fault state and turn on a red LED. If the previous state was the enable state, the microcontroller will disable PWM outputs to the gate drivers shutting down power to the motor.
GLV Power	Power the control hardware.	Power failure.	Vehicle is turned off or GLV supply fails.	If the motor is being driven when the vehicle shuts off, the next time it is turned on, the controller will immediately try to drive the motor. This is due to the values in the PWM register not reseting after a power down event.	The main controller reads the low voltage supply. If the voltage goes below 10V, the main controller will raise a low voltage flag.	When the low voltage flag goes high, the control PCB will turn off pwm outputs to the IGBTs shutting off power to the motor. The 3.3V DCDC has capacitors that will supply the main controller with enough power to shut itself down properly.
DC Bus	Supplies the input DC power for motor control.	DC is hooked up with reverse polarity.	Human error.	The free wheeling diodes of the IGBTs may become damaged due to a large amount of current. This large current draw may also damage the DC supply.	The controller is able to measure a negative input DC voltage.	The controller will read the negative DC voltage before enabling the pwm outputs to the gate driver and prevent the controller from going into enable mode.
Processor	Runs the control system.	Processor hung up during task.	Code error or communication is disrupted between other ICs.	The processor will not be able to update the PWM values, resulting in the motor coming to a dead stop.	The controller has an independent watchdog (IWDG) implemented on it. This watchdog is refreshed in the main while loop. If it is not refreshed within 1ms, the microcontroller will reset itself. The user will be able to detect the fault by viewing the LED turning from green to blue or by the LED flashing blue.	When the microcontroller resets itself, it will not automatically go back into enable mode and stay in idle. The user will view the LED changing from green to blue (enable to idle state) or view a blue flashing LED. The user can diagnose were the software is being help up by running the software in debug mode.
Current Sensors	Provides phase currents for control system	Microcontroller receives incorrect phase current sensors	Hall effect sensors, external ADCs, or wiring become damaged .	The output to the motor will behave unpredictably	The SPI to the external ADCs is pulled high. If communication with the ADCs fail the microcontroller will read "0xFFFF" and raise a current sensor fault flag. The main controller also adds up the three phase currents each time it reads. If the sum of these phase currents exceeds 20A then it will rase a current sensor fault flag.	Upon failure detection, the microcontroller will enter a fault state and turn on a red LED. If the previous state was the enable state, the microcontroller will disable PWM outputs to the gate drivers shutting down power to the motor.

Figure 7-1: Failure modes and effects analysis

7.2 Embedded Software Fault Detection

The microcontroller is continuously polling for faults. If a fault is detected the microcontroller transitions to the faulted state and a red LED turns on. If the previous state is the enabled state the microcontroller will disable the PWM signals to the gate drivers shutting down power the motor. The microcontroller will not be able to change from the faulted state. The user will have to diagnose the fault, fix it, and restart the motor controller. The user can use the GUI to determine what fault occurred as seen in figure 7-2.

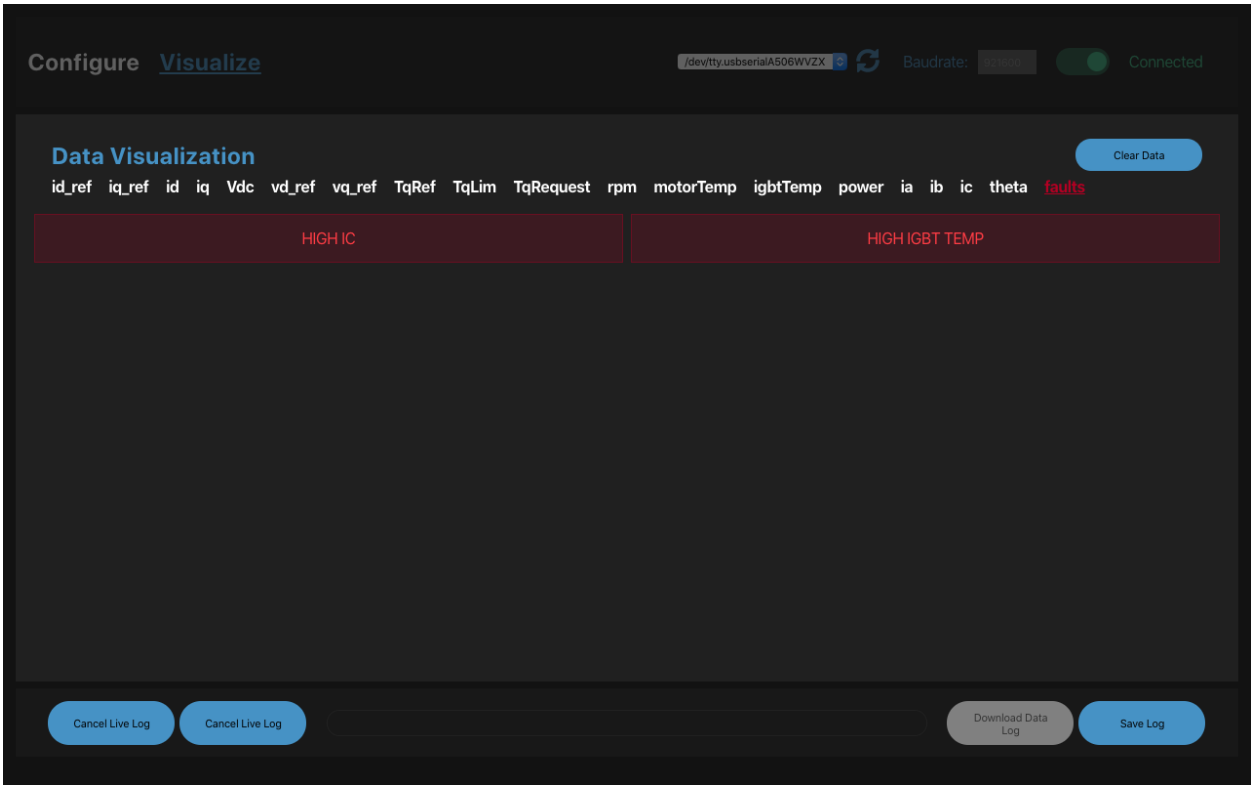


Figure 7-2: GUI Fault Display

As seen above, this indicates a “HIGH IC“ fault and a “HIGH IGBT TEMP“ fault. Currently the following faults are reported by the software:

1. HIGH VDC
2. DFSDM TIMEOUT
3. HIGH IA

4. HIGH IB
5. HIGH IC
6. IABC SPI
7. RDC CRC
8. RDC FAULT REG
9. HIGH MOTOR TEMP
10. HIGH IGBT TEMP
11. IGBT THERMISTOR
12. ADC
13. GATE DRIVER
14. LOW BATTERY
15. HARD FAULT

These are reported to the GUI in form of a 16 bit fault register. Each fault is mapped to a bit, and a bit value of 1 indicates that the system is faulty. Errors are subsequently revealed to the user upon arrival during regular data logging.

Chapter 8

Discussion

8.1 EMI

8.1.1 EMI Basics

There are two main sources of EMI inside a motor controller: noise due to the voltage switching, and noise due to current. The high frequency switching of voltages generates electric fields and introduce noise through capacitive coupling, while large currents generate magnetic fields. The high frequency switching noise originates from the IGBT module and the DC link capacitors, while the magnetic fields would occur around the three phases, IGBT module, and the DC inputs. Protecting against these different types require somewhat different precautions, but proper layout and grounding can improve all three. Protecting against capacitive coupling happens during PCB design, requiring careful isolation of signals. Radiative coupling can be mitigated by shielding. Magnetic shielding is done with magnetic materials, typically steel or another specialized magnetic material, such as mu-metal.

8.1.2 Switching Noise

When the PWM switching was first enabled at a high voltage, large voltage spikes started appearing on signal lines, and the communication between a laptop and the control PCB's microcontroller was disrupted. An image of the noise on the 5V power rail can be found in figure 8-1, where the noise spikes clearly lined up with the IGBT switching. This noise was visible as spikes occurring at a rate of twice the switching frequency.

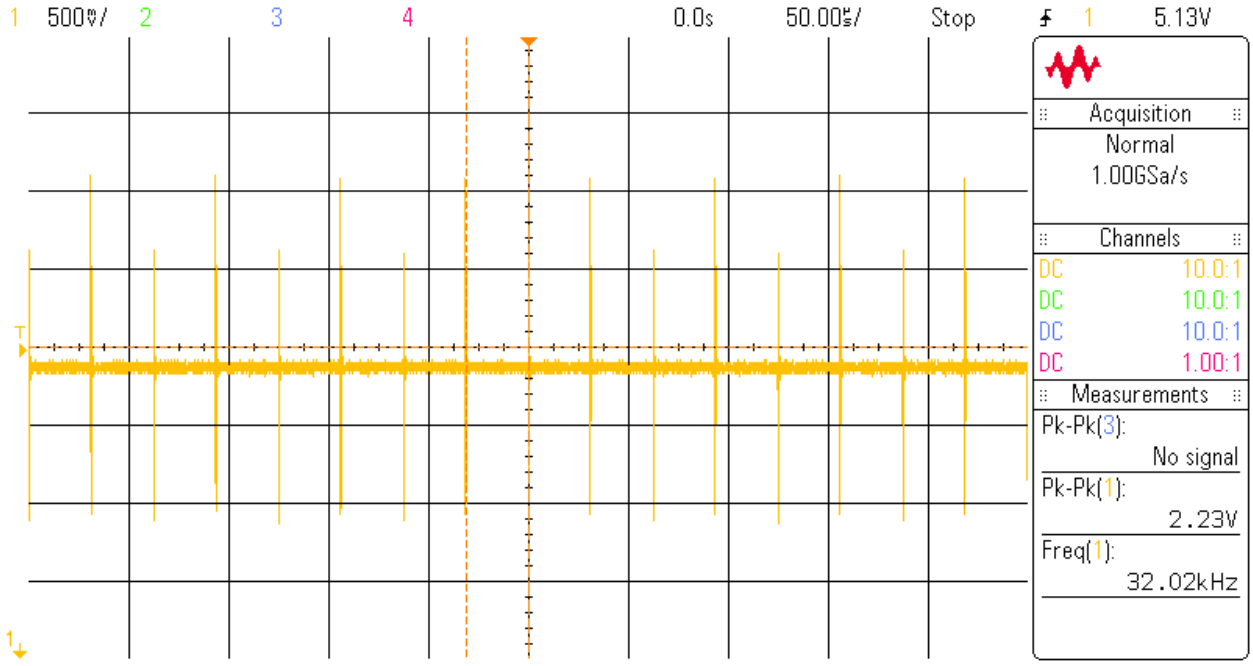


Figure 8-1: Oscilloscope capture of the noise on the 5V GLV rail

Addition shielding was introduced to fix the problem, and properly grounding the cooling plate had a noticeable positive effect. Shielding was added between the control PCB and the gate driver PCB, which also reduced the radiated noise, however this was later proved to be unnecessary. It was found that by disconnecting the shunt resistor from the current sensors, the PC would not disconnect while the controller was switching. In addition, the IGBT gates were checked for oscillations. When tested without a DC voltage applied, the gate voltages appeared as seen in figure 2-7. However, when measured with a voltage applied, the waveform in figure 8-2 was observed. As it would have been difficult to determine whether these oscillations were the cause of or an effect of the noise problems, the gate resistances were increased. Afterwords, the PC would not disconnect, even if the shunt resistors were plugged in.

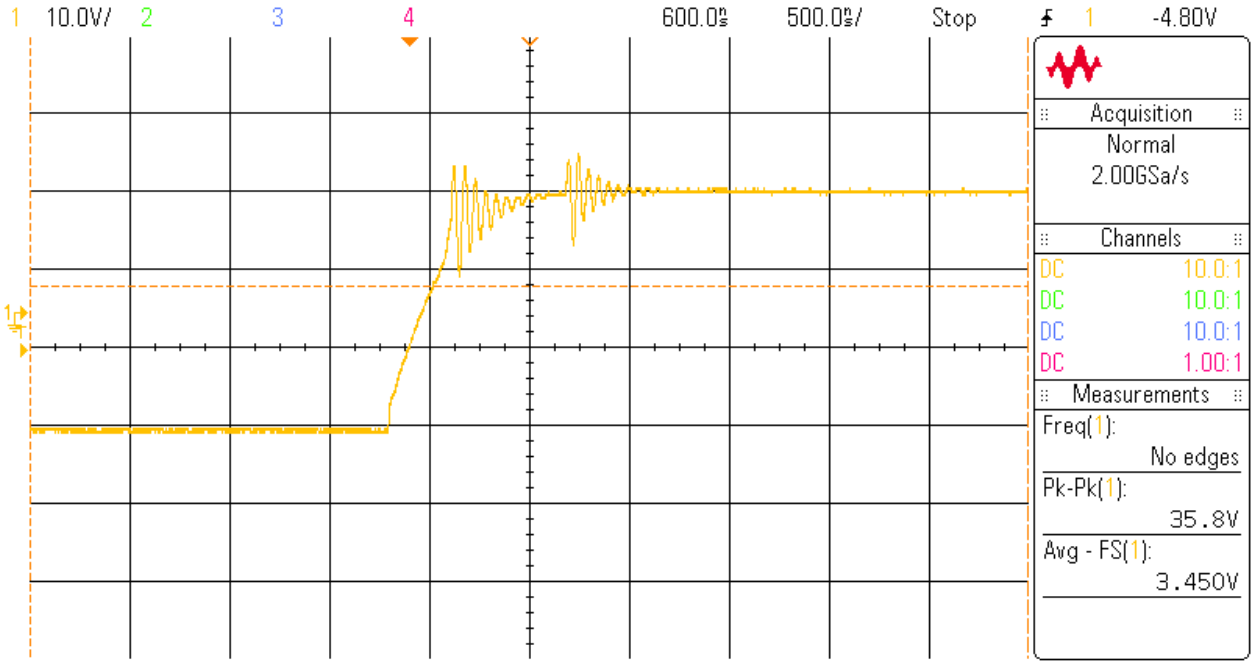


Figure 8-2: Oscilloscope capture of oscillations on the IGBT gates

These results suggested that the noise problems were due to capacitive coupling. Increasing the gate resistances increased the switching times lowering the $\frac{dV}{dt}$, which would reduce the magnitude of coupled noise. The three output phases, which the shunt resistors were connected to, experience the largest $\frac{dV}{dt}$ in the controller. This explains why disconnecting them reduced noise problems. The noise would have propagated to the GLV signals through capacitance between the input and output of either the isolated DCDCs or isolated ADCs.

8.1.3 Current Measurement Noise

While the controller was switching, the current measurements peaked at 60 A when the expected current was zero. An oscilloscope was used to measure the signal on the current sensors with the shunt resistors disconnected, the results of which can be seen in figure 8-3. An FFT of this data can be found in figure 8-4.

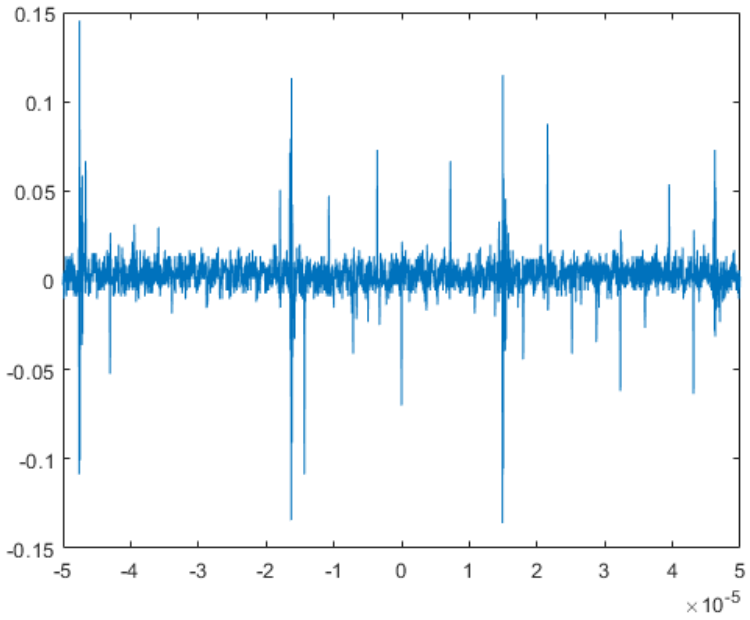


Figure 8-3: Oscilloscope data capture of noise on current sensing line

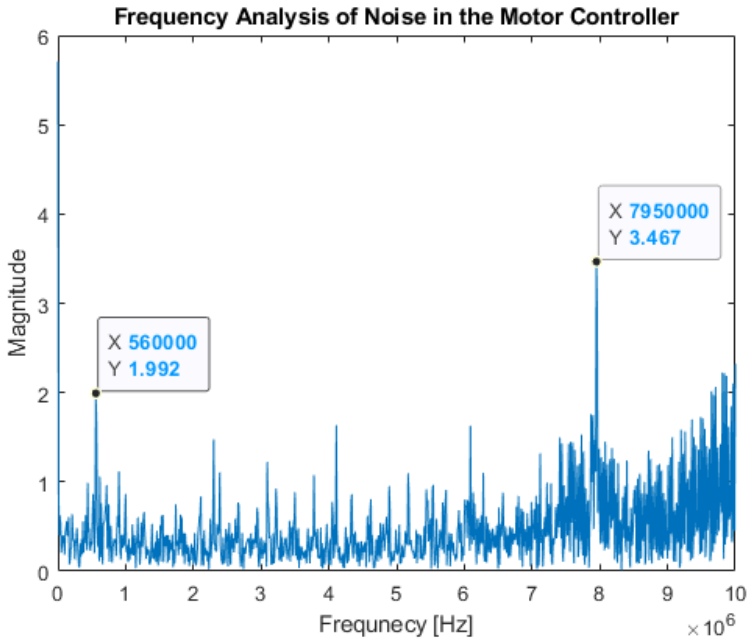


Figure 8-4: Fourier transform of the signal in figure 8-3

With the full scale of the current sensing ADC being +/- 50 mV, the noise on the line rendered it

useless. Instead, hall effect current sensors were purchased, as discussed in section 3.4. These provided addition isolation as there is no electrical connection between the measuring and measured systems, and could be located further away from the IGBT module and DC link capacitors, the two main sources of noise. Additionally, the full scale differential voltage output was +/- 800 mV. As the results in figure 8-5 indicate, this new measurement method significantly reduced the noise. The noise was reduced from 60 A peak to less than 0.3 A peak to peak.

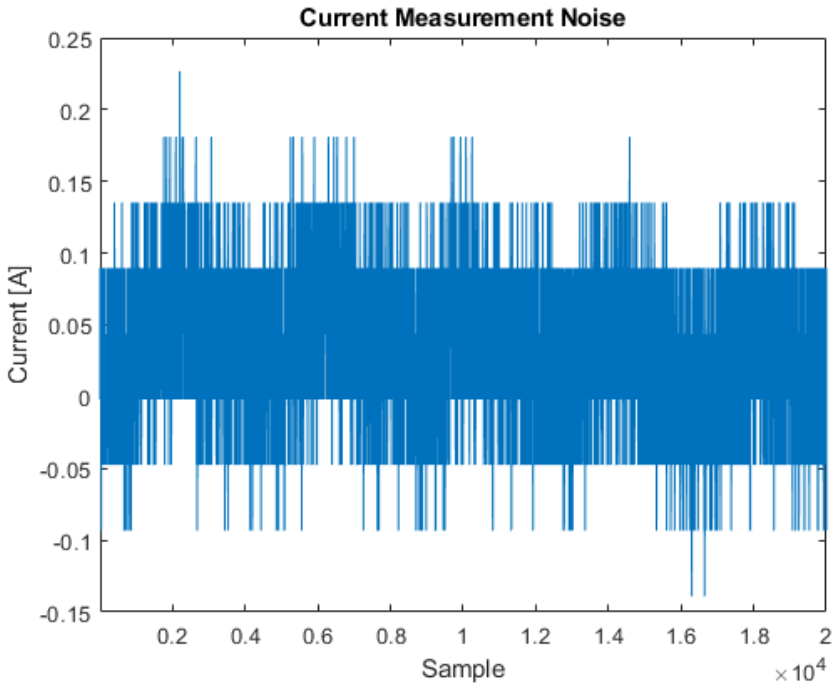


Figure 8-5: Phase current measurements taken with the hall effect sensor

8.1.4 Spectrum Analysis

A spectrum analysis was used to measure the magnitudes and frequencies emitted by the motor controller while switching. Two probes were used: an electric field probe, and a magnetic field probe. Measurements were taken in the room far away from the motor controller, and compared to measurements taken directly in between the DC link capacitors. The results of electric field measurements can be found in figure 8-6 and figure 8-7, and the results of the magnetic field measurements can be found in figure 8-8 and figure 8-9. These measurements show the presence of the electric fields and magnetic fields caused by the motor

controller. There are two noticeable peaks in the electric field, located at 5.6 MHz and 8.3 MHz. However, there is a significant increase in the magnetic field around the controller across the entire spectrum. These measurements suggest that in order to further reduce noise (that is not caused by capacitive coupling), magnetic shielding should be added.

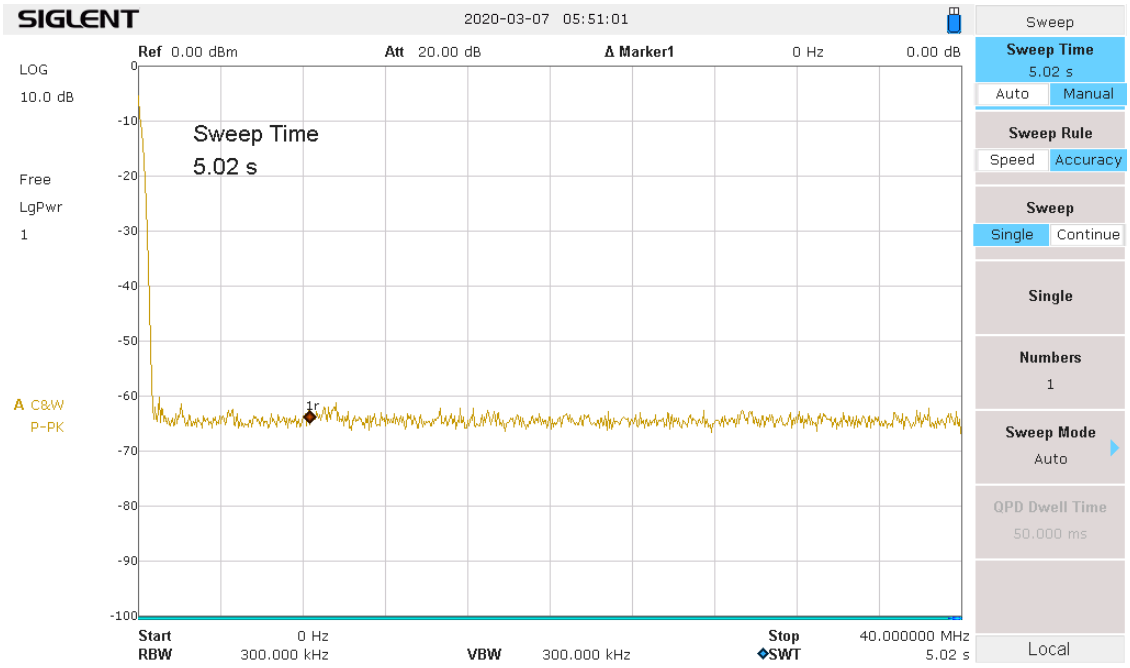


Figure 8-6: Results of a spectrum analyser sweep taken one meter away from the motor controller, using a electric field probe

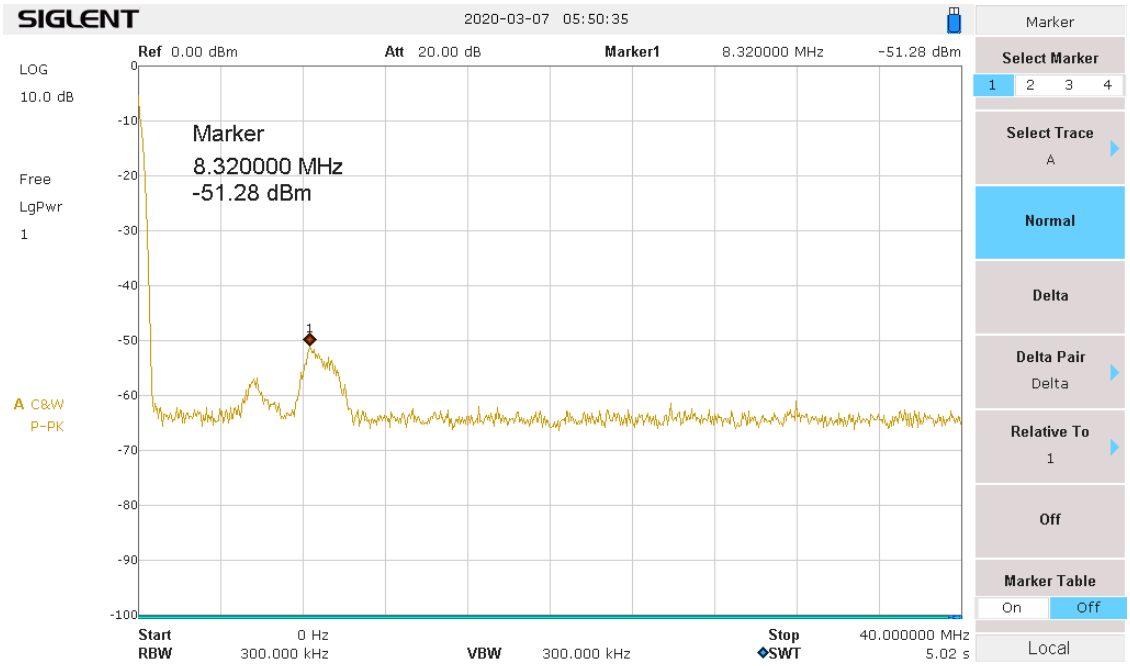


Figure 8-7: Results of a spectrum analyser sweep taken between the DC link capacitors, using a electric field probe

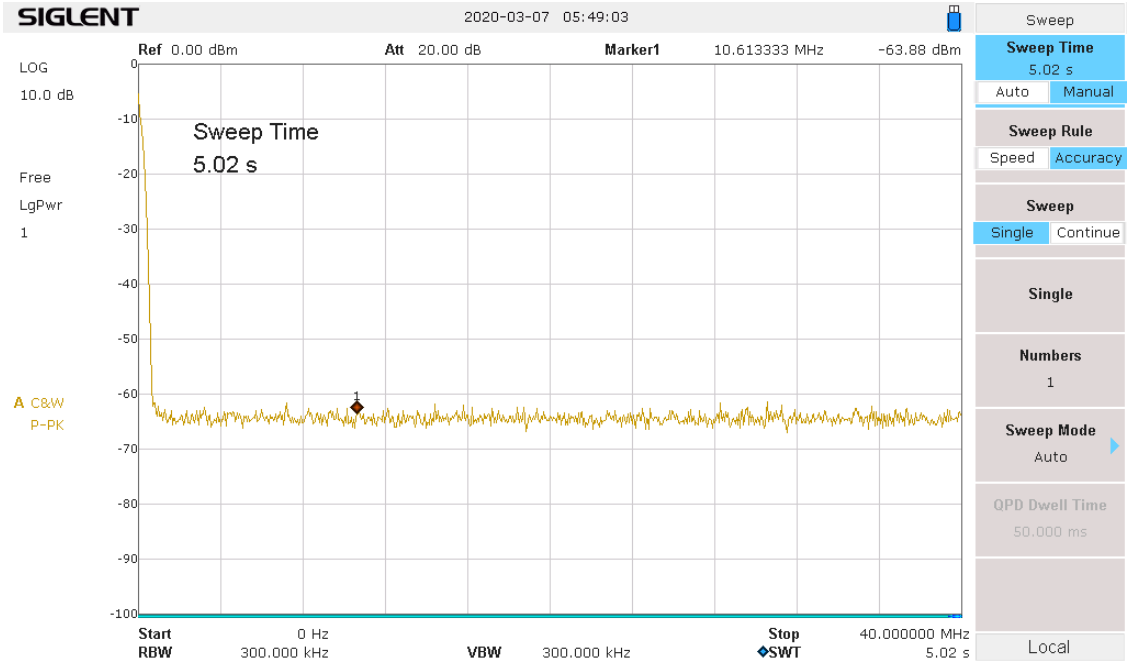


Figure 8-8: Results of a spectrum analyser sweep taken one meter away from the motor controller, using a magnetic field probe

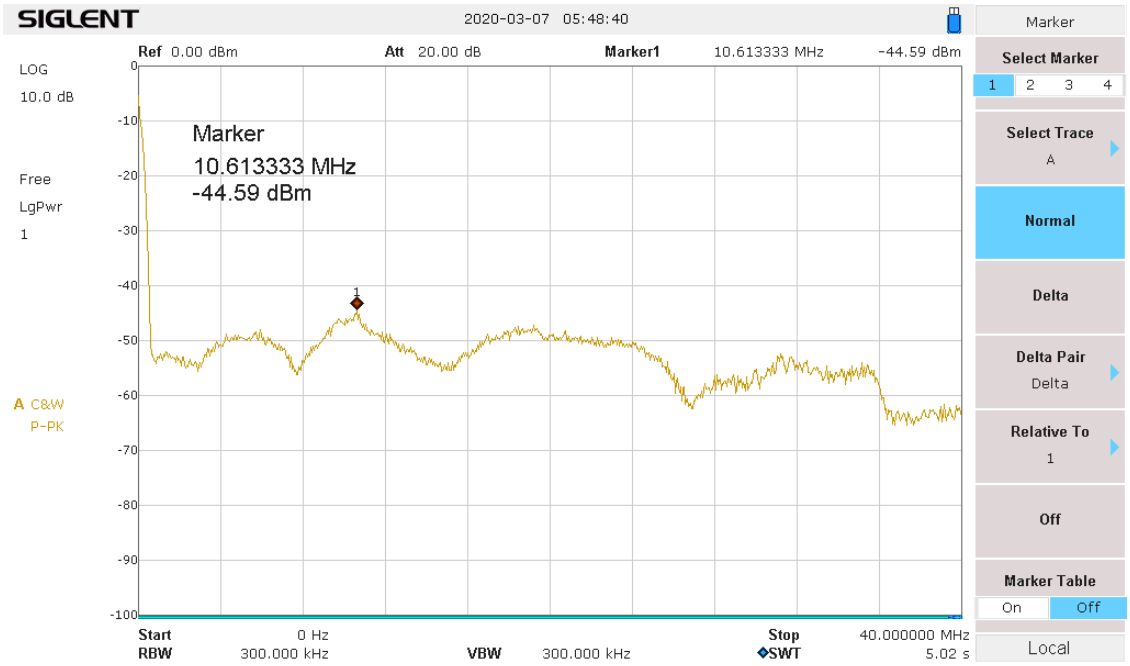


Figure 8-9: Results of a spectrum analyser sweep taken between the DC link capacitors, using a magnetic field probe

8.2 GUI Diagnostics

The ability to view live and log major control parameters proved to be invaluable during testing. An example of this is given below. The first time the motor was spun, all visual and audible indicators suggested that the controller was functioning properly. This was not the case when looking at the logs. The current of phase C can be found in figure 8-10. The current was expected to be largely sinusoidal, however, when looking at the measurements there is a clear ripple. This was due to the voltage measurements, which can be found in figure 8-11, showing a peak to peak ripple of almost 60 V. When the DC voltage was measured with an oscilloscope, the ripple was not present. A DC voltage measurement smaller than the actual voltage would increase the phase current, while a larger voltage measurements than the acutal voltage would reduce the phase current. This was the observed behaviour in figure 8-10 and figure 8-11. The cause of the measurement ripple, which was only present when switching the IGBTs, was an improperly tuned filter. The logging capabilities of the GUI made diagnosing this problem fast and easy.

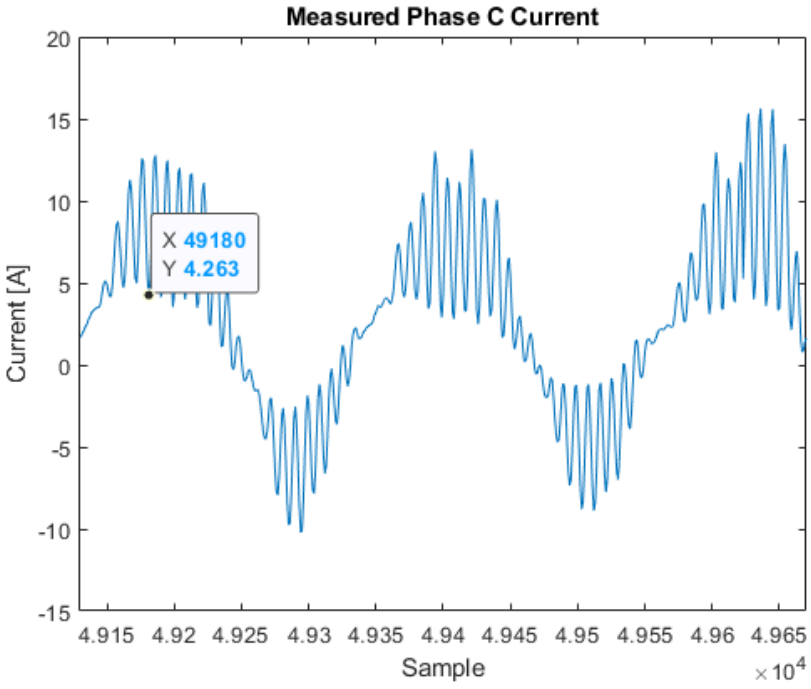


Figure 8-10: Phase current measurements logged during testing

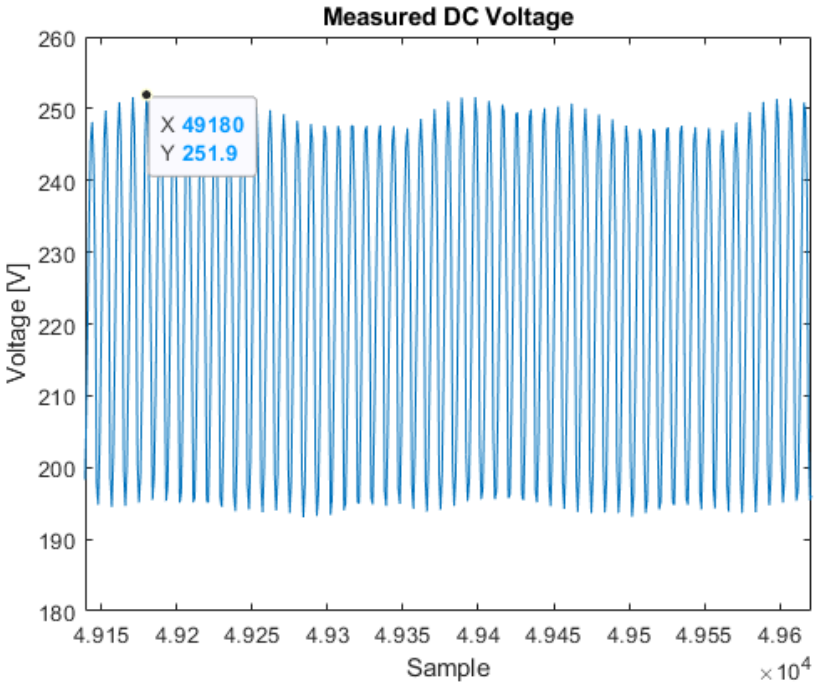


Figure 8-11: DC voltage measurements logged during testing

In order to reduce the noise, a larger capacitance was used. The RC filter was tuned to have a cutoff frequency half a decade lower than the switching frequency. The measurements in figure 8-12 were taken after the filter was fixed. This resulted in an improvement from nearly 60 V peak to peak of noise, to less than 1 V peak to peak of noise. The effect of this can be found in figure 8-13, where the measured phase currents are largely sinusoidal.

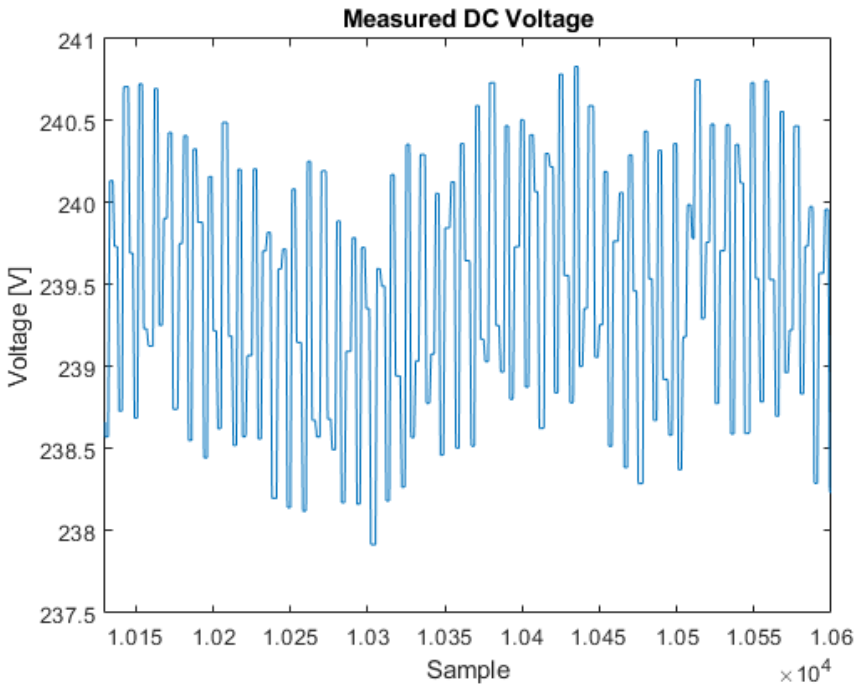


Figure 8-12: DC voltage measurements after properly tuning the RC filter

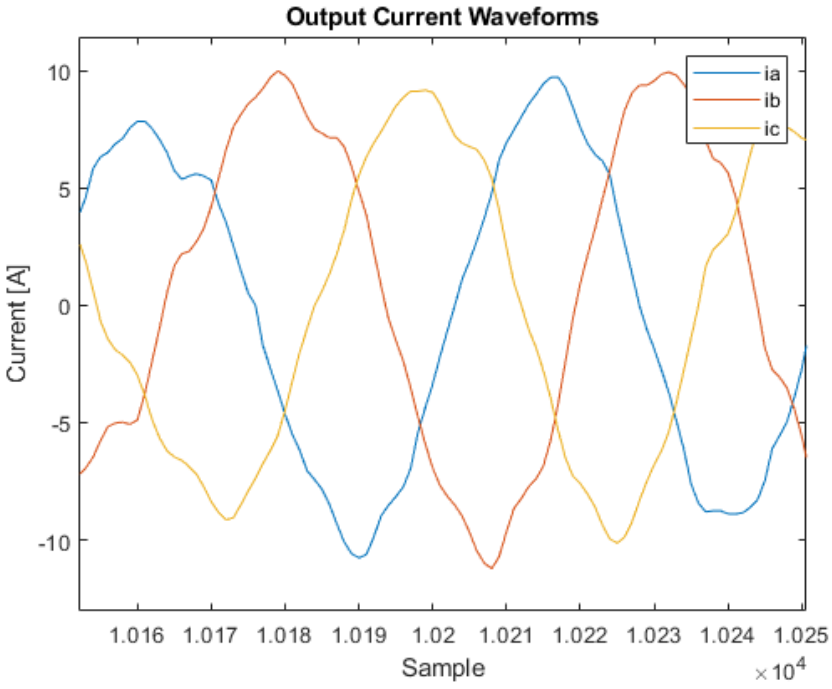


Figure 8-13: Output phase currents after adjusting the DC voltage measurement

Chapter 9

Testing

9.1 Preliminary Validation

All components and aspects of the prototype were tested before a motor was driven for the first time. A list of the key tests can be found in table 9-1. The results of these tests are described in further detail in the following sections.

Table 9-1: Validation performed before first motor test

Test	Passed?
LV Smoke Test	Yes
HV Smoke Test	Yes
Correct PWM dead time	Yes
IGBTs switching properly	Yes
Control system code can be run at 16kHz	Yes
Control system code calculations line up with simulations	Yes
Current sensors function properly	Yes
Calibrate and validate resolver offset	Yes
Motor and IGBT temperature sensors operational	Yes
Controller shuts down if fault is detected	Yes

9.1.1 PWM Dead Time Verification

The results of the PWM dead time can be seen in the scope image in figure 9-1. Channel 1 (yellow) and Channel 2 (green) show the PWM gate driver signals for the high and low side of phase A. The dead time is 980 ns when both of these signals are low. As the programmed dead time was 1 µs, the dead-time passed .

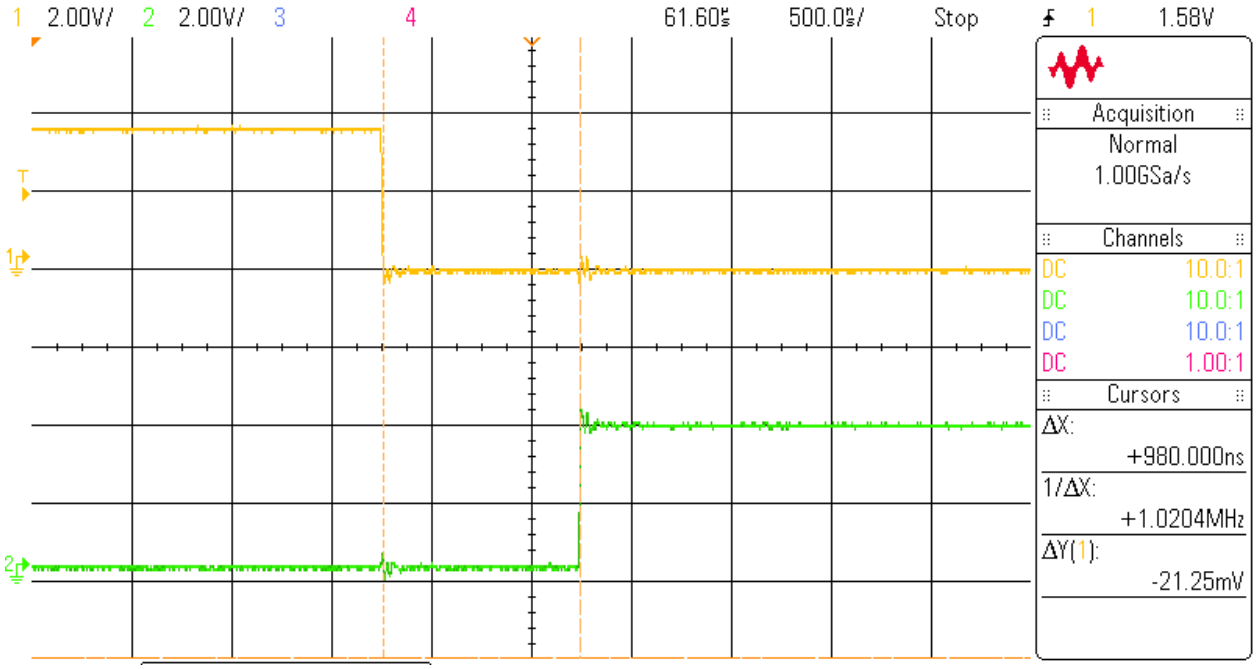


Figure 9-1: Micro controller complimentary PWM outputs showing dead time

9.1.2 Control Loop Speed Tests

The results of the controller system interrupt timing can be seen in the scope image in figure 9-2. The two oscilloscope channels make up the control system interrupt that runs at 16 kHz. Channel 1 (yellow) is the time it takes to read and interpret the sensors for the control system loop, 32.5 μs, and Channel 2 (green) is the time it takes to run the inner loop and Space-vector Pulse Width Modulation (SVPWM) calculations of the control loop, 12.9μs. With a total time of 45.4μs, this is more than enough time to run the interrupt at 16 kHz. The 17.1μs between the control loop interrupts will run the main loop of the microcontroller which contains functions who's timing is not critical to the control loop.

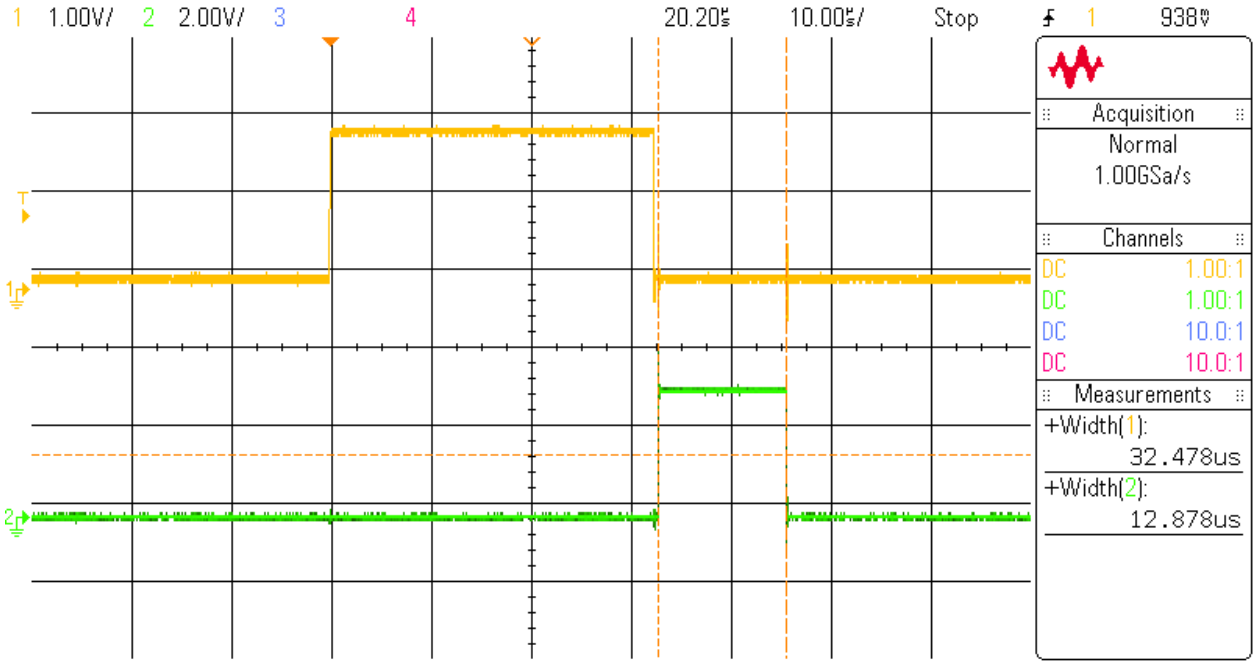


Figure 9-2: Control system interrupt timing scope

9.1.3 Control System Code

Control system inputs and outputs were generated from the Simulink model, which were fed into the adapted control system C code in order to verify its operation. The results of the control system code and the Simulink results can be seen in figure 9-3, where V_d makes up part of the V_{dq} vector that is an input into the Space-vector Pulse Width Modulation (SVPWM) calculations. For this test the control system code calculated V_d using same inputs as used in control system simulation in Simulink. From the graph, the calculated V_d from the control system code closely follows the Simulink. Similar results were found for V_q. These comparisons verify that the modified control system code is a accurate approximation of the Simulink model.

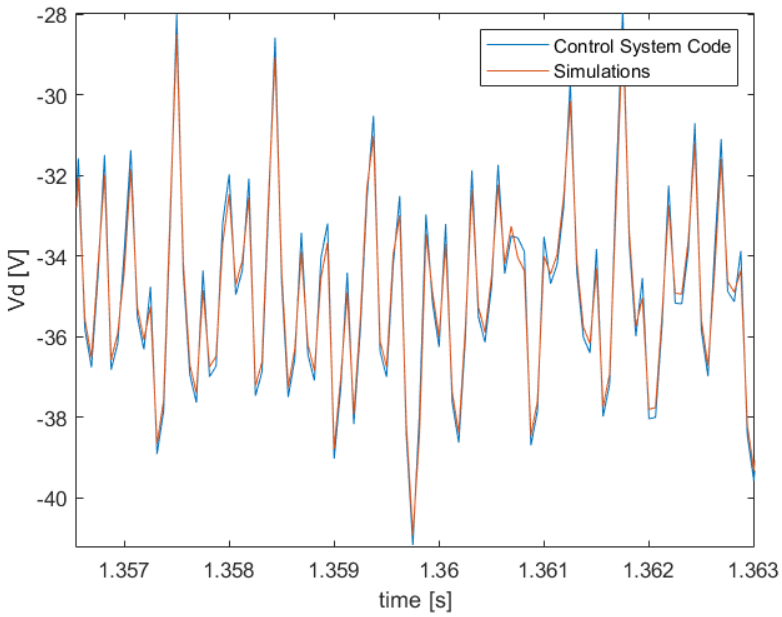


Figure 9-3: Comparison of V_d from the control system code and simulations

9.1.4 Fault Shutdown Validation

The results of the controller shutting down when fault a detected can be seen in the oscilloscope screen in figure 9-4. In the oscilloscope screen, Channel 1 (yellow) is the output signal form the micro-controller that is high during the function that disables the PWM signals that go to the gate driver, Channel 2 (green) is the output switching voltage of phase C from the IGBT. Channel 1 shows that the disable function takes 59.84 μs. Channel 2 shows the output phase C voltage turns off within these 59.84 μs. This proves that the controller shuts down in 59.84 μs when a fault is detected.

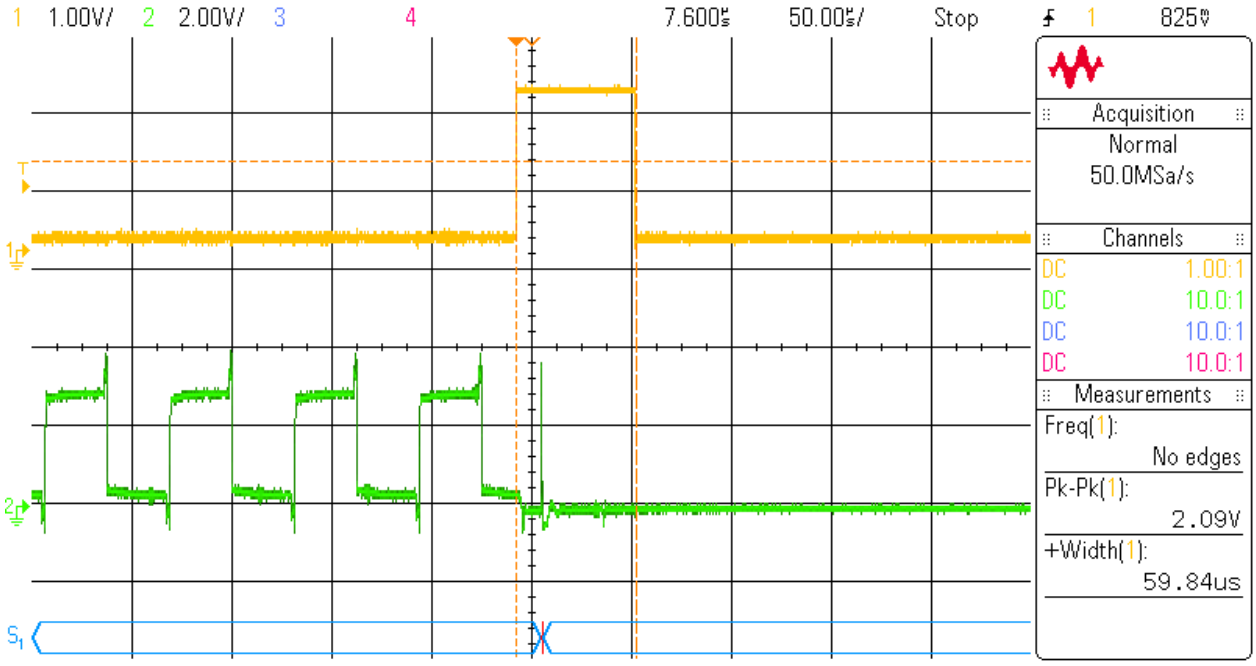


Figure 9-4: Fault shutdown time scope

9.1.5 IGBT Operation

In order to verify the IGBT operation, the microcontroller generated the PWM waveforms for an input of zero torque. With a stationary motor and a torque request of zero, the output voltages should have a duty cycle of around 50%. This is the behaviour seen in the waveforms found in figure 9-5.

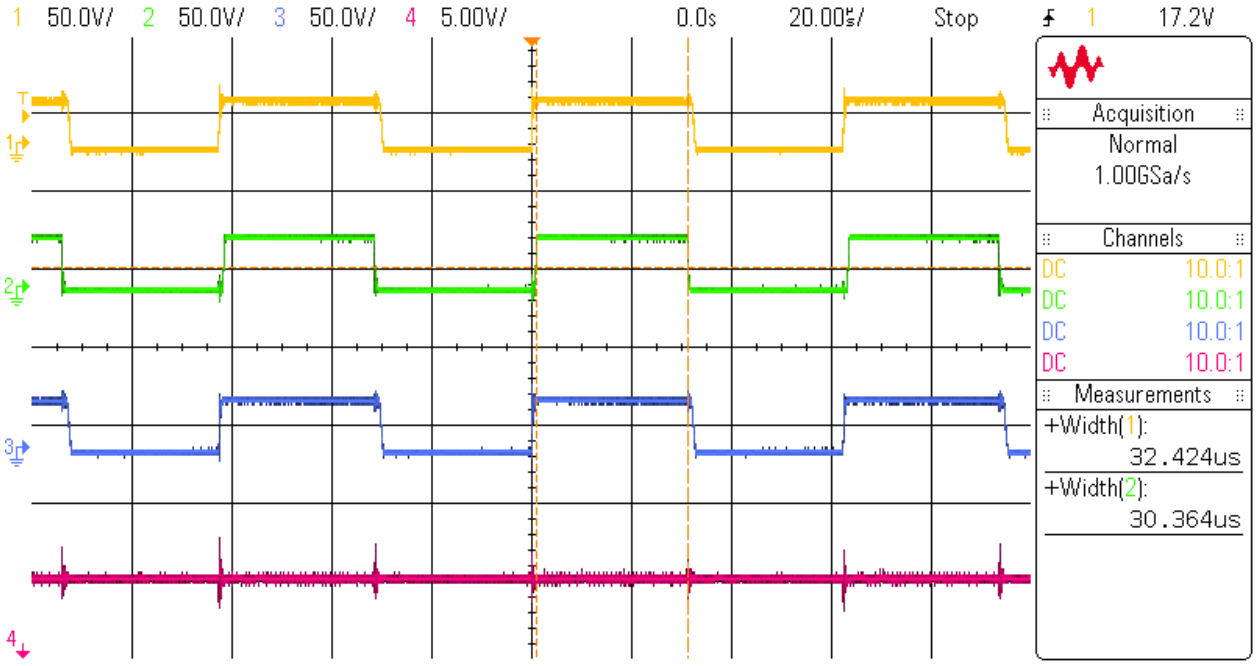


Figure 9-5: Output waveforms of the IGBT module

9.1.6 Current Sensor Validation

In order to verify the operation and calibration of the LEM current sensors, a known current was measured with the device. A DC load capable of outputting up to 5 A supplied the current, and a bench multimeter was used to confirm the current measured. A wire was wrapped through the current sensor 5 times, which allowed for testing of up to $+/-5 \cdot 5 = +/-25$ A. In order to reduce the effects of noise, the average ADC value of 100 measurements was taken, and compared against the measurement current. The results of this test for phase C can be found in figure 9-6. From this data, the offset and slop of the transfer function were determined.

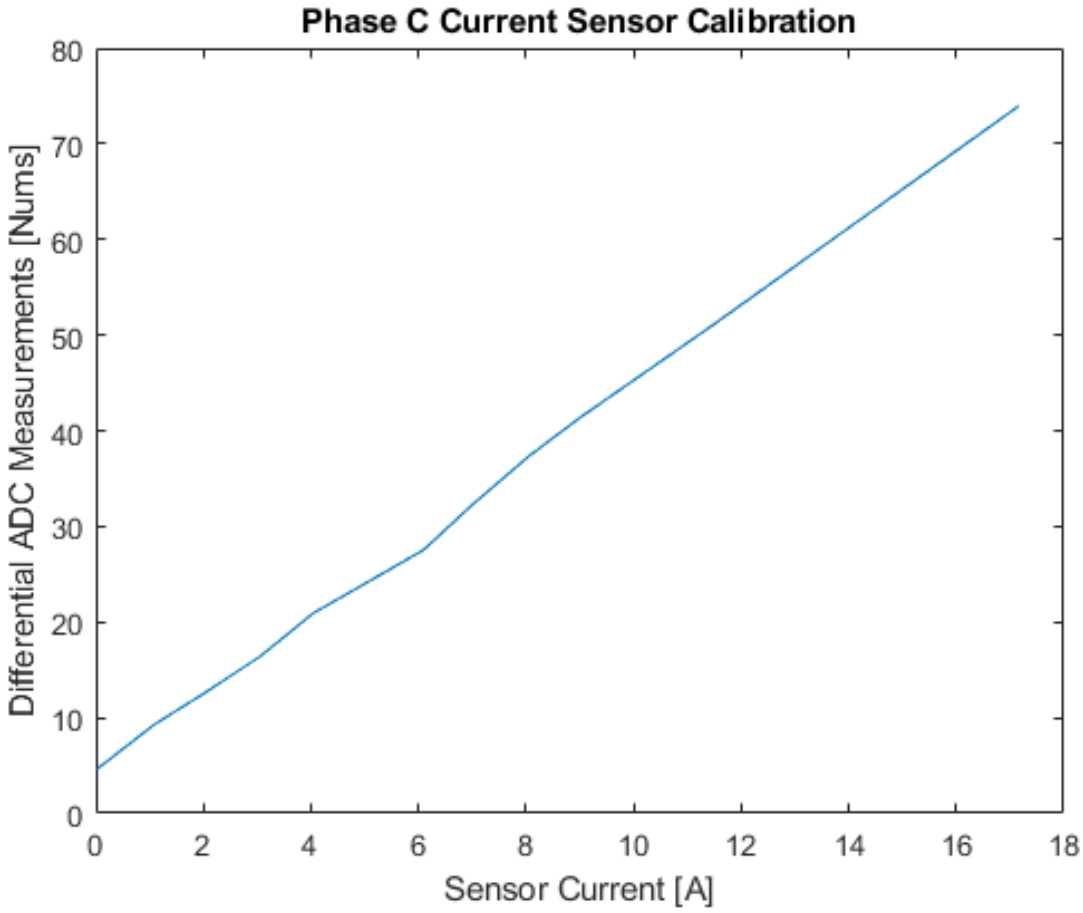


Figure 9-6: Hall effect current sensor calibration

9.1.7 Resolver Calibration

In order to run the control system, the correct resolver offset must be programmed on the controller. As the EMRAX 228 had relatively low saliency, the magnetic angle of the motor was determined through the back emf. To generate the back EMF, the motor was spun using an external motor, which was rated for 1000 rpm. The setup used can be found in figure 9-7. Using the GUI, the line to line voltage measurements and the resolver angle were logged. Using this data, the electrical angle was aligned with the peak of phase voltage A. The phase connections used can be found in table 9-2.

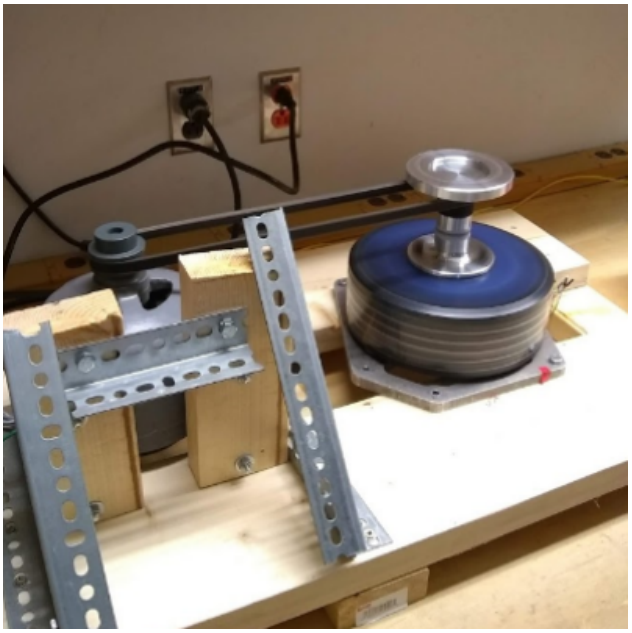


Figure 9-7: Test setup used for resolver calibration

Table 9-2: Phase connection order

Controller Phase	Emrax 228 Phase Colour
A	Red
B	Blue
C	Black

The logged data was then processed using Matlab. A low pass filter was used to reduce noise in the voltage measurements. The processed data taken from a resolver zero crossing can be found in figure 9-8. Several zero crossings were measured at different speeds in order to estimate the resolver angle offset.

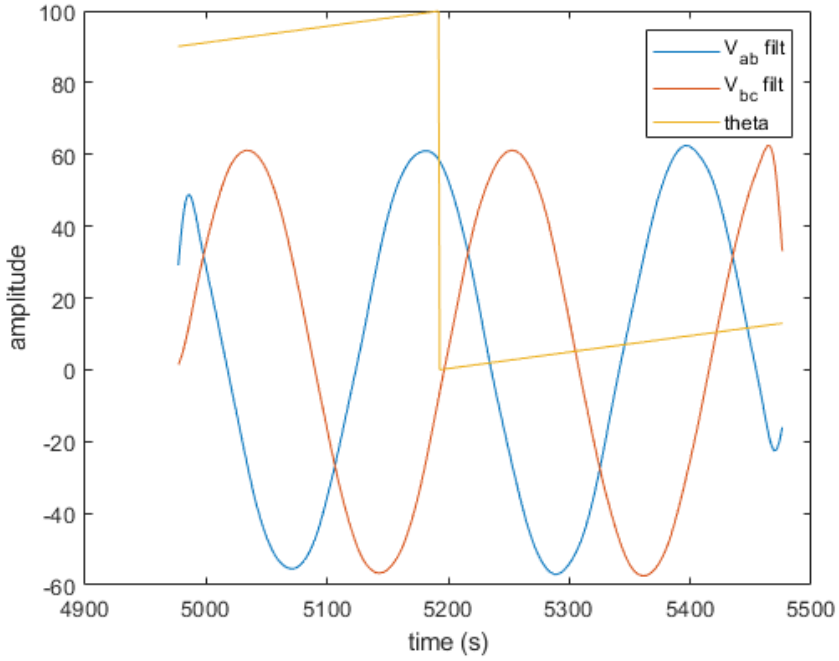


Figure 9-8: Data used to determine resolver angle offset

9.1.8 Temperature Sensor Validation

The temperature sensing circuitry was first tested by measuring a resistance simulating room temperature, and then a resistance simulating the upper operating limits of the motor and IGBT. In order to test the current sensors themselves, the motor and IGBT module were allowed to heat up. A temperature gun was used to take external measurements of the temperature of the devices, once they had heated up, and once they reached ambient. The results can be found in table 9-3.

Table 9-3: Results of temperature sensor testing

Device	Temperature read by motor controller	Temperature read by heat gun
Motor	29°C	30.6°C
Motor	35°C	33.5°C
IGBT	28°C	25.5°C
IGBT	40°C	37.5°C

9.1.9 Motor Controller Parameter Configuration Timing

The time to configure motor controller parameters was validated by measuring the elapsed time from pressing the ”Write Parameters” button in the GUI to when the final acknowledgement was received—indicating successful configuration. Measurements were taken both in software and using a smartphone which both gave congruent results. Output from the in-software timing can be seen in figure 9-9.

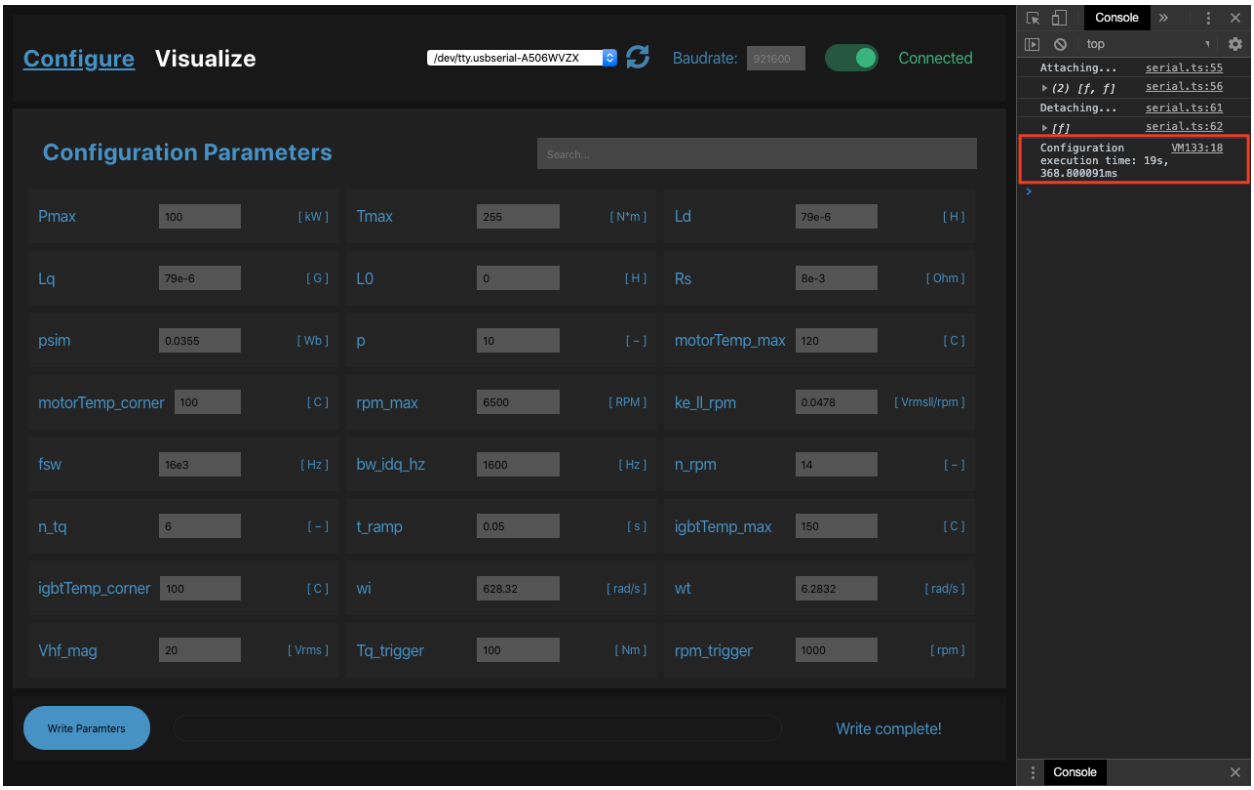


Figure 9-9: Timing requirement validation of parameter configuration

As seen above, the entire configuration process took less than 20 seconds which is far less than the requirement of less than 5 minutes. This gives the user over 4 and a half minutes to launch the application and fill out the configuration form.

9.2 Motor Controller Testing

Once all aspects of the controller were proven to be working, testing for the metrics could begin. The three metrics requiring the motor to be tested were the maximum motor speed, the motor controller efficiency,

and the maximum controller current.

9.2.1 Speed Test

The results of the speed test can be found in figure 9-10. The motor was able to sustain a speed greater than 3500 RPM, meeting the speed metric. The output voltages obtained during the test can be found in figure 9-11. In this image, the three PWM waveforms can be seen operating correctly.

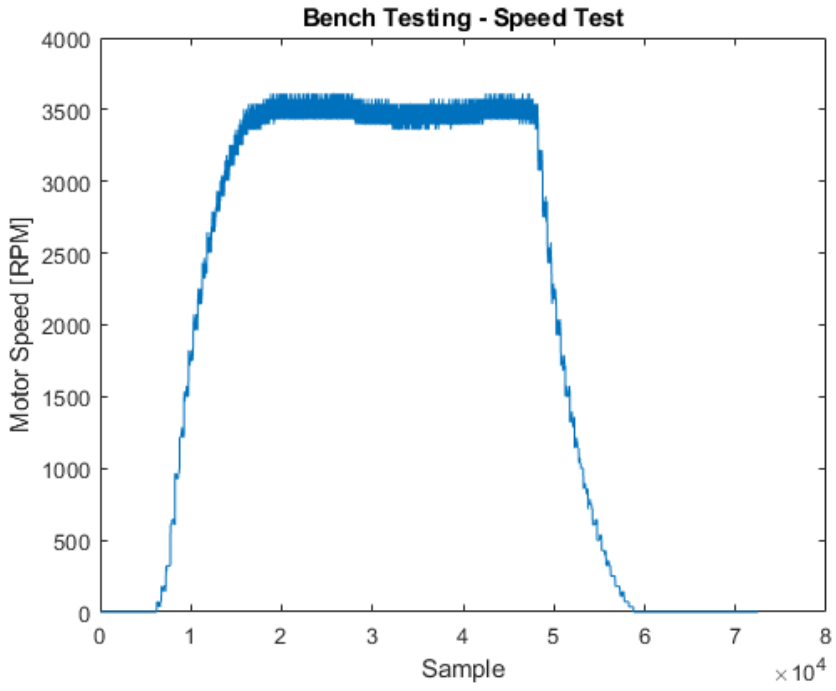


Figure 9-10: Results of the motor speed testing

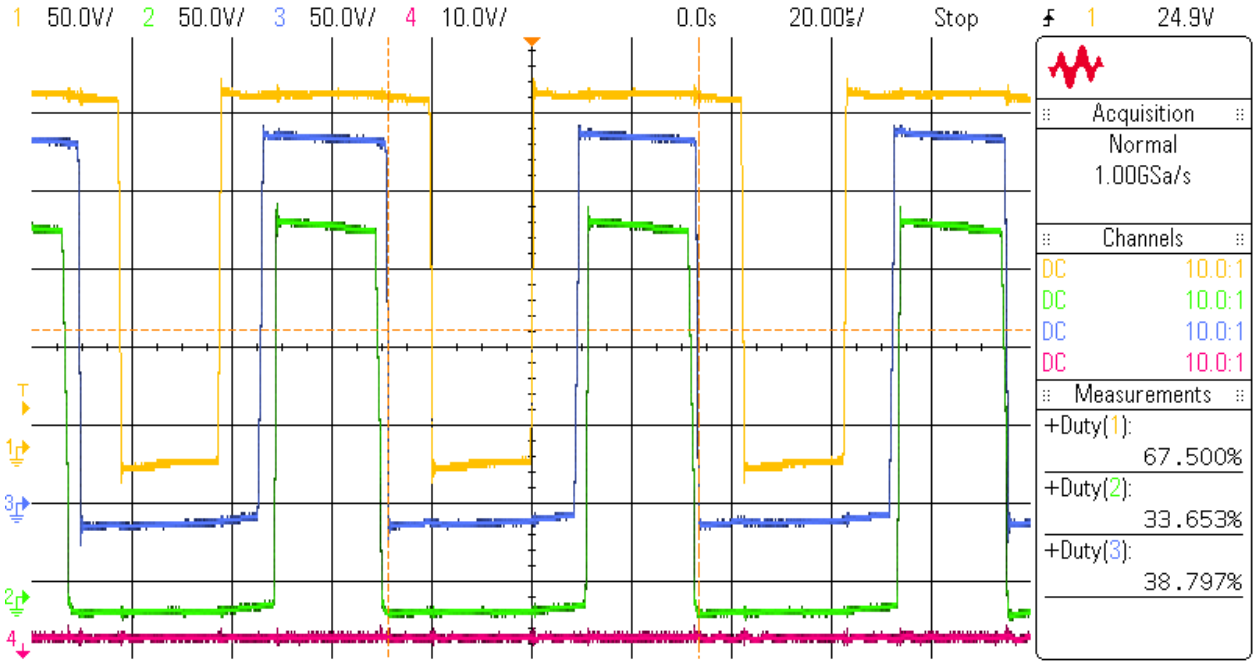


Figure 9-11: Oscilloscope capture of output voltages

During testing, the motor output voltages along with the phase A current waveforms were captured, as seen in figure 9-12. in this image, the phase current can be seen increasing and decreasing along with the PWM output.

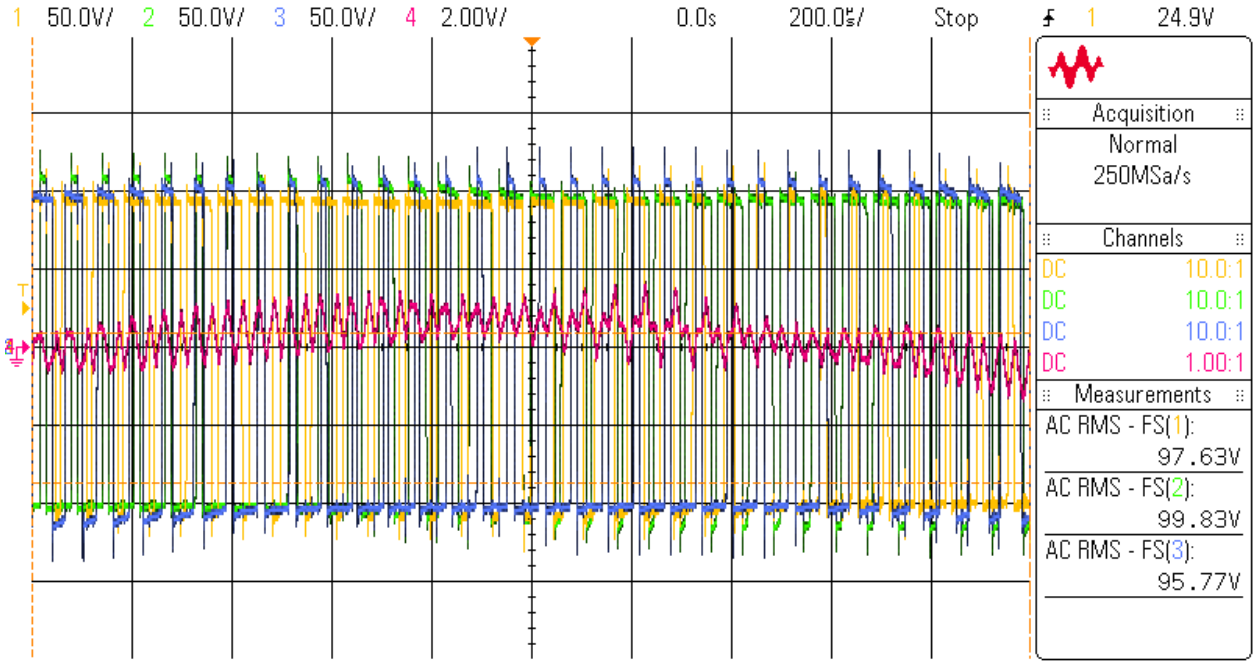


Figure 9-12: Oscilloscope capture of output voltages

9.2.2 Efficiency Test

Originally, the input and output powers were calculated on the motor controller. The input power was calculated by measuring the DC input voltage and currents, and the output was calculated using the phase currents and the estimated output phase voltages. This gave the waveforms found in figure 9-13, and suggested and efficiency of only $\frac{P_{out,avg}}{P_{in,avg}} = \frac{371.3}{703.8} = 52.7\%$. However, as the motor controller did not heat up during testing, this data was invalid. The IGBT module would have seen a noticeable increase in temperature if it were dissipating 300 W of heat.

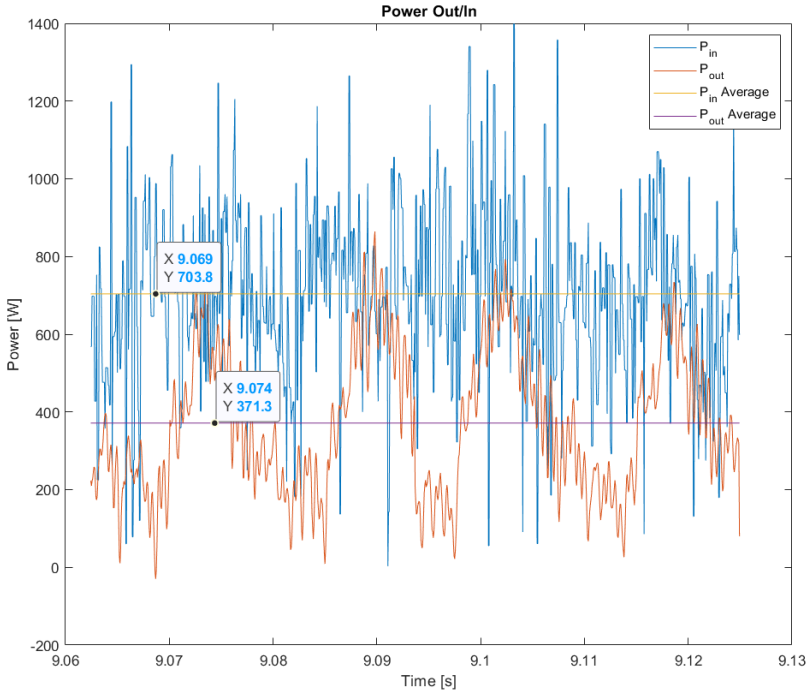


Figure 9-13: Calculated power waveforms

Instead, the input power was calculated by the the displayed current and voltage output of the 300 V supply, and the output power was calculated from the motor's losses. The measurements were taken at once the test reached steady state, where the motor was rotated at a constant speed, so that the output power of the motor controller would exactly cancel out the motor losses. The free run losses indicated in the datasheet can be found in figure 9-14. During the efficiency test, the motor reached 3500 RPM. Using figure 9-14, the estimated losses for this speed are 1.4 kW. The 300 V supply was outputting 298 V and 5.5 A. This gives an estimated efficiency of $\frac{1400}{298 \cdot 5.5} = \frac{1400}{1639} = 85.4\%$, which does not meet the efficiency metric of 90%. This lower than expected efficiency is thought to be due to additional motor losses. The motor datasheet is known to have optimistic values, so the given losses may be less than the actual losses. Additionally, the motor used has metal shavings inside, which would increase the frictional losses.

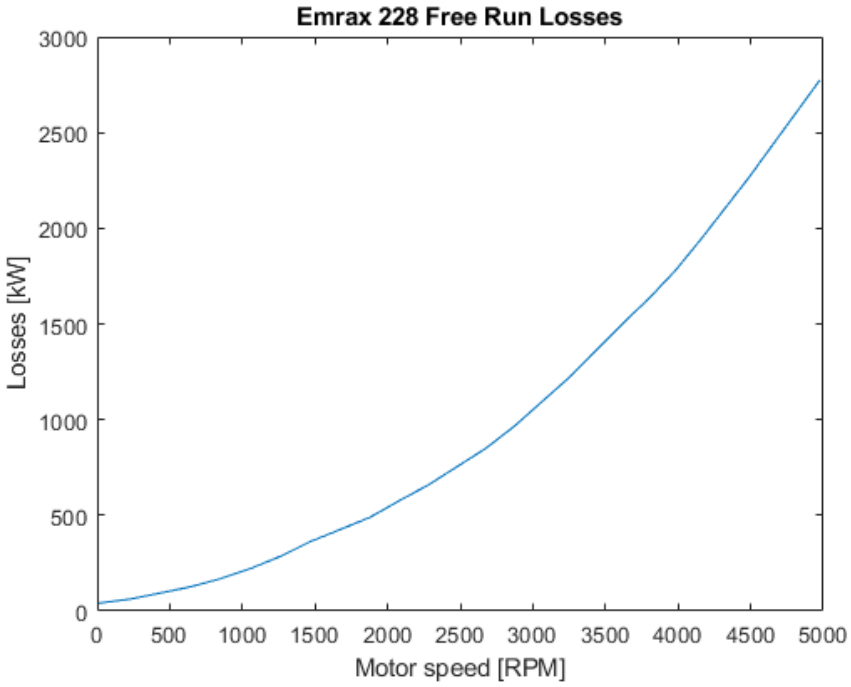


Figure 9-14: Emrax 228 free run losses

9.2.3 Current Test

In order to test the maximum IGBT currents, the rotor of the motor was held stationary and a torque command was send to the controller. The resultant current measurements can be found in figure 9-15 and figure 9-16. The test ended destructively, with the gate drivers detecting a fault. The controller was able to detect the failure, and shut down without damaging the 300 V supply. One of the phases had become permanently shorted to DC-. During the test, the IGBT module itself was at room temperature, so the failure was not due to inadequate cooling. Instead, the failure was due to the currents exceeding the IGBT's operating conditions. Not captured on the graphs below, the final measured phase current of the damaged phase was in excess of -300 A, but it is likely that this was due to failure, and not the cause. A second phase read 200 A when the short was detected, but was still operational. Based on the measured q-axis current just before device failure, the maximum operating current of the motor controller seems to be 115 A. This is less than the performance metric of 200A, and fixing this problem would require selecting a new IGBT module.

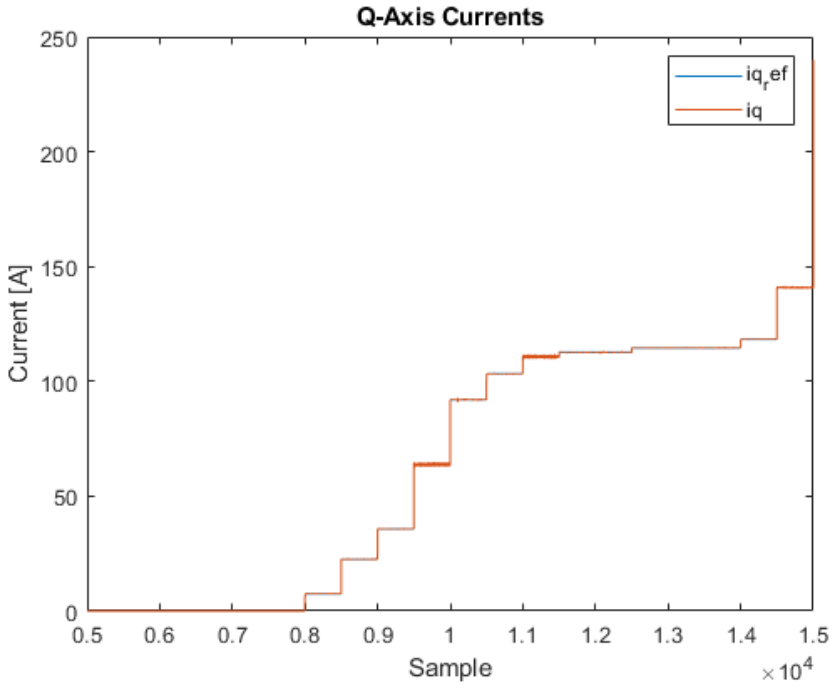


Figure 9-15: Q-axis currents from blocked rotor test

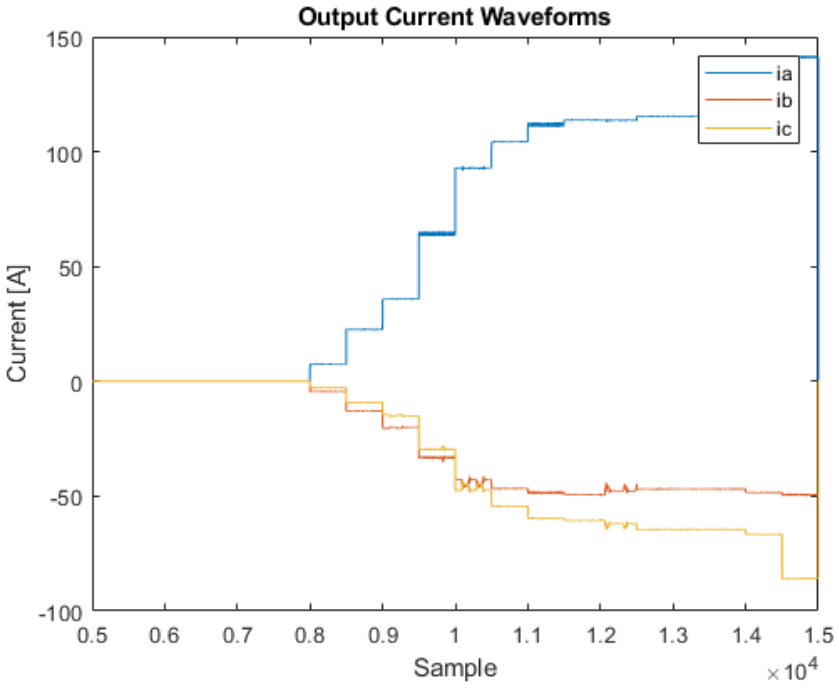


Figure 9-16: Measured phase currents from blocked rotor test

9.3 Summary of Metric Validation

At this time, seven metrics have been tested, five of which were passed. The specifics are found in table 9-4.

Table 9-4: Project metrics

Performance Metric	**Target**	**Measured Performance**	**Passed?**
Efficiency	90% at 1kW	85.4%	No
Output Current	200 A	115 A	No
User Interface	$<$ 5 minutes to configure motor controller	$<$20 s	Yes
Switching Frequency	$>$ 12 kHz	16 kHz	Yes
Max Output Mechanical Frequency	$>$ 50 Hz (3000 RPM)	3612 RPM	Yes
Inner Control Loop Frequency	$>$ 800 Hz	16 kHz	Yes
Fault Shutoff Time	$<$ 5ms	59.84 μs	Yes

Chapter 10

Future Considerations

10.1 Hardware Changes

Several hardware architecture changes would improve the overall design. Currently, the motor controller relies on an external discharge circuit to discharge the DC link capacitors. However, if this could be packaged well, this circuit should be inside the motor controller.

If the GLV power signal was regulated to a constant voltage, more options are available for DCDCs. Currently, the entire system is designed to operate over a voltage range of 12-16.8 V, which is the range of the GLV system battery. If this constraint was no longer in place, there would be more flexibility in the gate driver design. Smaller package sizes and additional voltages are available.

In order to reduce the noise present on the control PCB, the isolation barrier could be moved from the gate driver PCB to the control PCB. This would keep the control signals further away from the primary source of noise: the IGBT module. Additionally, the number of analog signals could be reduced. The line-to-line voltage sensors and IGBT temperature sensor signals could be converted into digital signals on the gate driver PCB, instead of running the analog lines back to the micro controller.

10.2 IGBT Module

The IGBT module tested was very appealing. It was one-third of the cost and size of comparable solutions. However, it was too good to be true, and was not able to supply the required 200 A per phase. In order for the controller to be used effectively in the vehicle, larger IGBTs should be selected, capable of supplying 200 A continuously.

10.3 Embedded Systems Changes

Usage of the HAL libraries should be eliminated in the future. While the HAL libraries were very useful in rapid prototyping, they have an undesirable performance impact. Custom libraries and peripheral modules could be written to more closely satisfy the system's functional and timing requirements.

10.4 Graphical User Interface Changes

There are a number of additions that could be made to the software. Notably, control parameter configurations could be saved as a file and loaded into the application to avoid reconfiguration, or to define specific configuration profiles.

Additionally, more data analysis can be baked into the "Visualize" module—and perhaps be renamed "Analyze". It could also be possible for the user to load data transformations and analysis functions as user defined modules—potentially using Web Assembly. These modules could then be called from within the application without the need to export the data for analysis using other software. Further, currently live logging stores all incoming data in a single, consecutive array of values which gives the false impression that they are temporally equidistant. In the future, complete logs should be disconnected from one another as they are separate entities. This could also be solved if time stamps could be sent from the motor controller as part of the log structure.

10.5 MOSFETS

Silicon carbide MOSFETs are becoming increasingly popular in high power motor controllers, especially in tractive applications. These devices offer operation at higher switching frequencies, which would reduce harmonics and improve the efficiency of the motor. However, they are significantly more expensive, and would not have been feasible within the given budget.

10.6 Field Weakening

Field weakening is a technique that allows for a motor to operate beyond its rated speed, without increasing the DC link voltage. A D-axis current is used to weaken the field of the magnets, reducing the back EMF, while reducing the maximum torque output. At the time, this extended operating range was not deemed a

requirement by the UMSAE Formula Electric team, and was not pursued further.

Chapter 11

Conclusion

The goal of this project was to develop and test a prototype of a high performance motor controller suitable for the Formula Electric team. While only five of seven metrics were completed, fixing these issues would require significant hardware redesign. While the developed control is not suitable for driving an EMRAX 188 motor, it still provides a useful testing platform for motor control systems, and provides the groundwork and testing for future iterations. Critical shortcomings and potential improvements were identified, a promising IGBT module was stress tested, a motor control system validated, and a highly useful GUI was developed and implemented, which can easily extend to other projects within the Formula Electric team. The team members had an opportunity to learn, and had the experience of developing a project from the ground up. Overall, the project was successful, and a functional motor control was designed.

References

[1] A.H. Bjørko and S.A. Tinderholt. "Design and Testing of a Race Car Inverter". July 2016. URL: https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2402189.

[2] S. Carpiuc, D. Patrascu, and C. Lazar. "Optimal torque control of the Interior Permanent Magnet Synchronous Machine". In: *2011 XXIII International Symposium on Information, Communication and Automation Technologies*. Oct. 2011, pp. 1–8. DOI: 10.1109/ICAT.2011.6102091.

[3] Concept. *IGBT and MOSFET Drivers Correctly Calculated*. URL: https://gate-driver.power.com/sites/default/files/product_document/application_note/AN-1001_IGBT_and_MOSFET_Drivers_Correctly_Calculated.pdf.

[4] Vishay Dale. *WSBS8518_20 series datasheet*. URL: https://www.vishay.com/docs/30341/wsbs8518_20.pdf.

[5] Cornell Dubilier. *Type 944U Polypropylene, DC Link Capacitors Datasheet*. URL: https://www.mouser.ca/datasheet/2/88/944U-22433.pdf.

[6] EMRAX. *Technical Data and Manual for EMRAX Motors / Generators*. URL: https://emrax.com/wp-content/uploads/2017/10/user_manual_for_emrax_motors.pdf.

[7] FTDI. *Future Technology Devices International Ltd. FT232R USB UART IC Datasheet*. URL: https://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT232R.pdf.

[8] Infineon. *How to Calculate and Minimize the dead time requirement for IGBTs Properly*. URL: https://www.infineon.com/dgdl/Infineon-Deadtime_calculation_for_IGBT_modules-ApplicationNotes-v01_00-EN.pdf?fileId=db3a30431a5c32f2011a5daefc41005b.

[9] Infineon. *IPOSIM*. URL: https://iposim.infineon.com/application/en (visited on 01/25/2020).

[10] Infineon. *Technical Information FS200R07PE4*. URL: https://www.infineon.com/dgdl/Infineon-FS200R07PE4-DS-v02_01-en_de.pdf?fileId=db3a304333227b5e01335963ee7908dd.

[11] Texas Instruments. *AMC1106x Datasheet*. URL: http://www.ti.com/lit/ds/symlink/amc1106m05.pdf.

[12] Texas Instruments. *AMC1106x datasheet*. URL: https://www.ti.com/lit/ds/symlink/amc1106m05.pdf.

[13] Texas Instruments. *External Gate Resistor Design Guide for Gate Drivers*. URL: https://www.ti.com/lit/an/slla385/slla385.pdf.

[14] Texas Instruments. *ISO5852S Datasheet*. URL: http://www.ti.com/lit/ds/symlink/iso5852s.pdf.

[15] Texas Instruments. *PGA411-Q1 datasheet*. URL: https://www.ti.com/lit/ds/symlink/pga411-q1.pdf.

[16] SAE International. *2020 Formula SAE Rules*. URL: https://www.fsaeonline.com/cdsweb/gen/DocumentResources.aspx.

[17] LEM. *Current Transducer HO-S series*. URL: https://www.lem.com/sites/default/files/products_datasheets/ho_50_250-s-0100_series.pdf.

[18] N. Maleki, Mohammad Alizadeh Pahlavani, and Iman Soltani. "A Detailed Comparison Between FOC and DTC Methods of a Permanent Magnet Synchronous Motor Drive". In: *Journal of Electrical and Electronic Engineering* 3 (Jan. 2015), pp. 92–100. DOI: 10.11648/j.jeee.s.2015030201.30.

[19] Mathworks. *PMSM Field-Oriented Control*. URL: https://www.mathworks.com/help/physmod/sps/ref/pmsmfieldorientedcontrol.html.

[20] Cascadia Motion. *PM100 datasheet*. URL: https://www.cascadiamotion.com/uploads/5/1/3/0/51309945/pm100_-_cascadia_motion_2019_catalog.pdf.

[21] Lars Helge Opsahl. "Design and Testing of Voltage Source Inverter and Motor Control System for Electric Vehicle". July 2015. URL: https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2368215.

[22] Wisconsin Racing. *120kW Inverter for All-Wheel Drive Electric Racecar*. URL: https://static1.squarespace.com/static/57e8888fc534a547699d733d/t/5b9590f988251b7b96966427/1536528657982/Quad+Inverter+Report.pdf.

[23] Lazar Rozenblat. *Calculating Spacing Between PCB Traces for Various Voltage Levels*. URL: https://www.smps.us/pcbtracespacing.html.

[24] ON Semiconductor. *MJD2955, MJD3055 Complimentary Power Transistors Datasheet*. URL: https://www.mouser.ca/datasheet/2/88/944U-22433.pdf.

[25] NXP Semiconductors. *KTY81 Series Datasheet*. URL: https://www.nxp.com/docs/en/data-sheet/KTY81_SER.pdf.

[26] Tamagawa. *Smartsyn resolvers datasheet*. URL: https://images.100y.com.tw/pdf_file/79-TAMAGAWA.pdf.

[27] Microchip Technology. *MCP33131D Datasheet*. URL: http://ww1.microchip.com/downloads/en/DeviceDoc/MCP33131D-Data-Sheet-DS20005947B.pdf.

[28] Mean Well. *SPBW03, DPW03 Series Datasheet*. URL: https://media.digikey.com/pdf/Data%5C%20Sheets/Mean%5C%20Well%5C%20PDF's/SPBW03,DPBW03_Series_Ds.pdf.

[29] Dave Wilson. *Teaching Old Motors New Tricks*. Texas Instruments. Mar. 2015. URL: https://training.ti.com/teaching-old-motors-new-tricks-part-1-introduction-motor-control-pi-controllers-pid-controllers-and?context=1137615-31562-31457 (visited on 02/28/2020).

[30] Dave Wilson. *Teaching Your PI Controller to Behave*. Texas Instruments. July 2015. URL: https://e2e.ti.com/blogs_/b/industrial_strength/archive/2015/07/20/teaching-your-pi-controller-to-behave-part-i (visited on 02/28/2020).

Appendix A

Appendix

A.1 Original Metrics

The original project metrics, as found in the project proposal, can be found below in table A-1.

Performance Metric	Target
Efficiency	90% at 1kW
Output Current	200A
User Interface	$<$ 5 minutes to change control parameters
Switching Frequency	$>$ 12 kHz
Max Output Mechanical Frequency	$>$ 83 Hz (5000 RPM)
Inner Control Loop Frequency	$>$ 800 Hz
Fault Shutoff Time	$<$ 5ms

Table A-1: Project metrics, as found in the original proposal

A.2 DC Link Capacitance Derivation

This section needs more detail.

$$\frac{V_{min}}{V_{nom}} = k$$

$$E = \frac{1}{2}CV^2$$

$$\frac{V_{min}}{V_{nom}} = \sqrt{\frac{E_{min}}{E_{nom}}}$$

$$\frac{E_{min}}{E_{nom}} = k^2$$

$$\Delta E = E_{nom} - E_{min}$$

$$E_{nom} = \frac{\Delta E}{1-k^2}$$

$$C_{min} = \frac{2 \cdot E_{nom}}{V_{nom}^2}$$

$$C_{min} = \frac{2\Delta E}{V_{nom}^2(1-k^2)}$$

A.3 Source Code

The embedded systems and graphical user interface software can be found on GitHub at the link below:

https://github.com/uofm-capstone-2020