School of Engineering, Discipline of Mechanical Engineering and Mechatronics, pp. 1–122

ENGG4801B - NU Racing Design Engineer-Mechatronics

2023

Joshua Dawson [1]

[1] *Student of Mechatronics Engineering,*
The University of Newcastle, Callaghan, NSW 2308, AUSTRALIA
Student Number: 3305339
E-mail: Joshua.J.Dawson@uon.edu.au

Declaration

I declare that this thesis is my own work unless otherwise acknowledged and is in accordance with the University's academic integrity policy available from the Policy Library on the web at: http://www.newcastle.edu.au/policylibrary/000608.html

I certify that this assessment item has not been submitted previously for academic credit in this or any other course.

I acknowledge that the Faculty of Engineering may, for the purpose of assessing this thesis:

Reproduce this thesis and provide a copy to another member of the Faculty; and/or

Communicate a copy of this assessment item to a plagiarism checking service (which may then retain a copy of the item on its database for the purpose of future plagiarism checking).

Submit the assessment item to other forms of plagiarism checking.

Where appropriate, provide a copy of this thesis to other students where that student is working in/on a related project.

I further acknowledge that the Faculty of Engineering may, for the purpose of sector wide Quality Assurance:

Reproduce this thesis to provide a copy to a member of another engineering faculty.

I certify that the electronic versions of this thesis I have submitted are identical to this hardcopy version. Furthermore, I have checked that the electronic version is complete and intact on the submitted location. Student name: Joshua Dawson Signature:

ADD SIGNATURE HERE

I have not left a mess in the lab declaration page.

I, Joshua Dawson, having completed the project FSAE, and using laboratory spaces TA, hereby submit that the area has been cleaned to level equal to or better than when I commenced my work. All items I borrowed have been returned, and the area is now suitable for inspection and sign off by the appropriate laboratory manager.

signed ADD SIGNATURE HERE

date : 31/01/2024

appropriate manager signed: date:

Abstract

This report details the work completed by the author as a final-year mechatronics engineering project with the University of Newcastle Formula-SAE team, NU Racing. The primary focus of the project was the commissioning of the verifying and commisioning the motor controller systems for NU23, Commissioning an eddy current dynamometer and, Charger modifications for NU23.

The verification and commissioning of the motor controller involved understanding the programming software and capabilities of the Bamocar D3-700/400 inverter, verifying the implementation of the settings used by the 2022 team, researching other methods of increasing performance and, commissioning a Cascadia Motions CM200DZ inverter.

The commisioning of the dynamometer involved the modification of existing circuitry to allow an Electric Vehicle (EV) motor to be used in place of an Internal Combustion Engine (ICE) and facilitation of concept design for the electrical node and safety requirements of using the dynamometer.

The commissioning of the charger involved designing the front mounting plate,designing and commissioning an Electronic Control Unit (ECU) mounting components into the case, and verification of its functionality.

The project concludes with the following major achievements:

- The Design, manufacture, and implementation of:
  - A competition-ready Inverter.
  - A competition-ready Charger.
- Delivery of a functional dynamometer.
- Concept design of electrical node and safety requirements to use they dynamometer.

Acknowledgements

I would like to express my gratitude to my family and friends, their encouragement and understanding have made all the difference.

To my dedicated teammates, I am immensely grateful for your collaboration and the shared moments of triumph and challenge. Our teamwork has made being a part of this journey both memorable and fulfilling.

I extend my special thanks to Dr. Alexander Gregg and Malcolm Sidney for their invaluable guidance and mentorship. Without their expertise and encouragement NU-Racing would not be what it is today.

I am deeply appreciative of everyone who has played a role in supporting me, and I am fortunate to have such an incredible network of individuals around me.

Contents

List of Figures

List of Tables

1 Background

1.1 NU-Teams

NU-Teams is an organization set up by the University of Newcastle Engineering Society and is aimed at student team-based project design in a variety of projects. The projects currently offered as part of this program are focused on developing motivated, professional, and capable engineers through real-world, hands-on, design and build projects. The primary objective is to provide students with opportunities to apply theoretical knowledge in practical settings, fostering skills and experiences that go beyond traditional classroom learning.

One of the current projects within NU-Teams involves the conceptualization, design, and construction of an Indy-style race car for the FSAE competition-EV Class. This endeavour not only challenges students to apply engineering principles in a competitive environment but also aims to contribute to the advancement of electric vehicle technology in the realm of high-performance racing.

The other 2 projects available at NU-Teams are:

NU-Marine: This project involves the comprehensive conceptualization, design, and construction of an autonomous marine vessel for participation in the esteemed International Maritime Robot-x Challenge. The focus is on cutting-edge technologies in maritime autonomy, providing students with a valuable opportunity to apply theoretical knowledge to real-world challenges.

NU-Rocketry: The objective of this project is to conceive, design, and construct high-powered rockets for entry into the Spaceport America Cup. Emphasizing precision engineering and compliance with stringent aerospace standards, participants engage in a meticulous process that hones their skills in rocketry and collaborative teamwork.

Figure 1: NU Teams Careers Expo

1.2 FSAE Competition

Formula SAE (FSAE) represents a student design competition officially sanctioned by the Society of Automotive Engineers, offering engineering students invaluable real-world project experience while pursuing their education. This unique competition, governed by the regulations of SAE International, empowers participants to channel their creativity and engineering skills into the conception, construction, and racing of a compact, Indy-style racing car. By engaging in this hands-on competition during their university years, students gain a significant head start in design experience, positioning them advantageously ahead of their peers in the engineering field.

1.3 EV3

EV3 is the name of the 3rd design iteration of the NU-Racing Electric Vehicle (EV) created by the 2022 NU-Racing team. A shortlist of relevant electrical design decisions are:

- Dual fixed RWD power train. This is driven by 2 Emrax 188 HV motors controlled by 2 Bamocar D3-700/400 inverters.
- Robust Modular electrical design. This was chosen over a singular ECU setup with central wiring harness as the electrical system, being a prototype, would have multiple revisions before being finalised. Implementing revisions to singular complex ECU was thought to more difficult than a having those systems spread over multiple simpler ECUs.

1.4 NU23

The over arching design goals for NU23, the 4th iteration of NU Racing's EV, was aligning bottlenecks, create simplified self-documenting systems, reduce part count and prioritise long term team performance.

Aligning bottlenecks is in reference to the performance mismatch between systems on the EV3 platform. A great example of this was the tractive system. EV3's powertrain, comprised of 2 Emrax 188 PMSM motors, had a continuous power output of 120kW combined. While this is already over 40kW more than the competition allows, it was not possible to reach the 80kW limit because the Accumulator was not capable of producing 80kW. Due to the voltage drop caused by the motors back EMF, the most power ever drawn was 55kW during acceleration at competition. This led to an over-designed drivetrain and under-designed power source. By aligning these bottlenecks, NU23 was designed to maximise the performance of all components all while minimising the weight.

As NU-Racing is a final year student project, much of the knowledge gained each year can be very easily lost as new students come in. If this transfer of knowledge is facilitated adequately there can be a regression in performance. Creating a self-documenting systems and designs would allow future student to quickly grasp the knowledge and engineering principles used and effectively improve upon them.

Reducing part count factors into the simplicity of good design methodology. It is not necessarily correct that the most complex solution is the most optimal one. For example, the powertrain of EV3 took over 10 hrs to remove and refit to the chassis and required specific knowledge to correctly align to the wheel shaft. NU23's powertrain was encased in a machined enclosure, named The Powerbox, that only required 50 minutes and no specific knowledge to align.

In previous years NU-Racing has attempted to implement more design goals than was feasibly to achieve in a single year. The 2023 team set achievable goals with reduced complexity to prioritise the long term performance of NU-Racing as a whole.

1.5 CAN-BUS

The following description of CAN-BUS is paraphrased from the NU-Teams Wiki page created by Dr Alexander Gregg. The wiki goes into far greater detail than is explained here and can be located here:

https://training.nuteams.org/electrical-software/can-bus-zero-to-hero/introduction-to-can-bus

The *Controller Area Network* (CAN) Bus serves as a message-based communication protocol facilitating the intercommunication of electronic control units (ECUs) and is widely used in the automotive and aerospace industries. In comparison to alternative messaging schemes, CAN Bus has the following advantages:

- **Efficient wiring and physical hardware requirements** - only two wires are required to connect all ECUs on a CAN network, allowing the potential for daisy-chaining ECUs with high efficiency.
- **Resilience to electrical noise and disturbances** - The CAN standard employs differential digital logic and twisted-pair cabling to minimize the impact of noise generated during transmission, ensuring robustness in challenging environments.
- **Simplicity of implementation** - Off-the-shelf hardware handles and removes the majority of complexity associated with managing message communication, simplifying the overall implementation process.
- **Relative feature-richness** - The standard offers capabilities such as straightforward message prioritization and arbitration, broadcast functionality, and the ability to initiate on-demand information requests.

To implement a CAN network the following is required:

1. **Hardware Requirements:**
    - CAN Transceivers
    - CAN Controller
2. **Power Supply:**
    - Stable power supply for the CAN nodes
3. **CAN Bus:**
    - Physical Wiring (CAN_H and CAN_L)
    - Termination Resistors between CAN high and low at 2 points of a bus containing any number of nodes. This resistance should equate to 60Ω, regardless of the number of nodes on the bus.
4. **Communication Protocol:**

- Implementation of the CAN communication protocol

5. **Node Addressing:**
   - Unique CAN identifiers (IDs) for each control node
6. **Data Format:**
   - Definition of the data format
7. **Message Structure:**
   - Structure of CAN messages (identifier, data payload, control bits)
8. **Error Handling:**
   - Implementation of error handling mechanisms
9. **Timing Considerations:**
   - Configuration of timing parameters (bit rate, synchronization, sampling points)
10. **Software Implementation:**
    - Development of firmware or software for CAN communication
11. **Testing and Debugging:**
    - Thorough testing of communication
    - Use of diagnostic tools for debugging
12. **Documentation:**
    - Documenting CAN communication protocol, node IDs, message formats, etc.

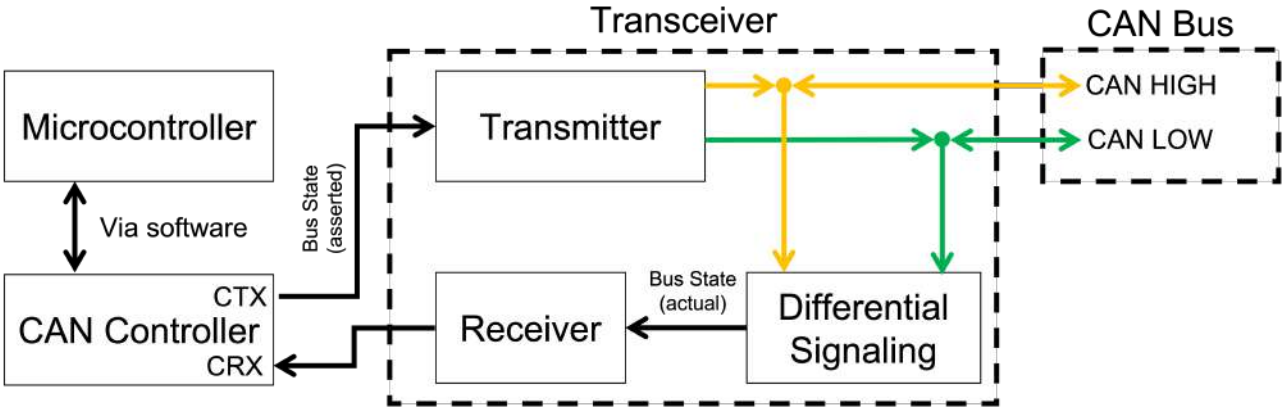


Figure 2: Functional block diagram of a CAN ECU/CAN Node from the CANBUS Zero-to-Hero wiki

1.6 Terminology/Nomenclature

This is a list of acronyms or terms used in this report with a brief description of each for simplicity.

- FSAE : Formula SAE competition.
- PID : Proportional-Integral-Derivative control is a feedback mechanism used to regulate systems. It consists of three components - Proportional (P), Integral (I), and Derivative (D) - which collectively minimize the difference between the desired setpoint and the actual system output, ensuring stability and responsiveness.
- TSMS : Tractive System Master Switch.
- HVTS : High Voltage Tractive System.
- GPIO pin : General Purpose Input or Output pin
- Powerbox : Referring to the assembly that houses the electric motor, gear reduction drive and main axle of the drivetrain.
- CAN/CanBus : Controller Area Network (CAN) bus communication protocol
- RCD : Residual Current Device.
- RCBO : Residual Current circuit Breaker with Over-current protection.
- scrutineering: Term used to describe the process that engineers do at the tech inspection for the FSAE competition. This process is inspecting and checking that the vehicle is compliant with the rules and regulations of the competition.
- Hard Faults:
  - BSPD: the Brake Pressure Sensor Device monitors hard braking, throttle openings over 10% while braking, loss of braking or throttle sensor signals for more than 100ms, and removal of power from its circuit.
  - IMD: The insulation Monitoring Device monitors the insulation resistance between the chassis and each pole of the accumulator. If at least one of the accumulator poles becomes not-insulated from the chassis, the IMD notices and shuts off the car.
  - PDOC: Precharge, Discharge, Open Circuit checks the temperature of thermistors placed directly adjacent to the power resistors for Precharging and discharging the HVTS. An open circuit is assumed to have occurred if these thermistors are above a certain temperature when they are not supposed to be.
  - AMS/BMS: The Accumulator Management System is responsible for monitoring the health of the Accumulator, and triggering the shutdown circuit should a fault occur.This is also referenced as Battery Monitoring System as the fault originates in that device. The main parameters that are monitored are the cell voltages/temperatures, and pack current.

2 Motor Controller Optimisation and Commissioning

2.1 Abstract

The initial controller setting testing was completed by the author. The Troubleshooting and Cascadia Controller sections were completed in conjunction with Jacob Bush[2] and Jye Hollier[4].

2.2 Bamocar D3-700/400

The Bamocar D3-700/400 motor inverter, seen in figure 3, is part of the UNITEK servo controllers family; known for its standardized high-performance digital control electronics that is used globally. Designed to be a robust and reliable controller for use in automotive vehicles, the following is a shortlist of design features:

- Robust IP65-rated aluminium casing.
- Over-specified power semiconductors and dc link capacitors for longevity.
- 6-12 L/m liquid cooling system.
- 12-700V_{AC} input voltage range and up to 400A_{RMS}.
- Digital interfaces (RS232, CAN-BUS).
- Programmable analog/digital inputs/outputs.
- Resolver or incremental encoder support.
- Current limit control for precision.
- Standardized fully digital control unit.
- Inherent safety features and short-circuit protection.

For further detail on the Bamocar's commissioning and tuning, please refer to G.Horsnell's thesis entitled *Formula SAE - Tractive System Design, Manufacturing and Testing,*[5]. The manual for the Bamocar can be found at:

https://www.unitek-industrie-elektronik.de/images/pdf/BAMOCAR%20Digital/BAMOCAR-PG-D3-700-400_EN.pdf

Figure 3: Bamocar D3-700/400 Inverter

2.3 NDrive v3xx

NDrive v3xx is the software interface used to program the controller, designed by Unitek. The system allows users to have access to any setting that could be changed on the controller which is very useful for optimisation of settings and parameter tuning. This can at the same time be very challenging as incorrect parameters can be detrimental to the controller's function, requiring a steep learning curve for anyone that isn't familiar with motor controller operation. The software manual gives enough detail and recommendations on each parameter's purpose and general range. For further detail on initial parameter setup and tuning, please refer to G.Horsnell's thesis entitled *Formula SAE - Tractive System Design, Manufacturing and Testing,*[5].

The software manual for Ndrive can be found at:

https://www.unitek-industrie-elektronik.de/images/pdf/NDrive/NDrive_EN.pdf

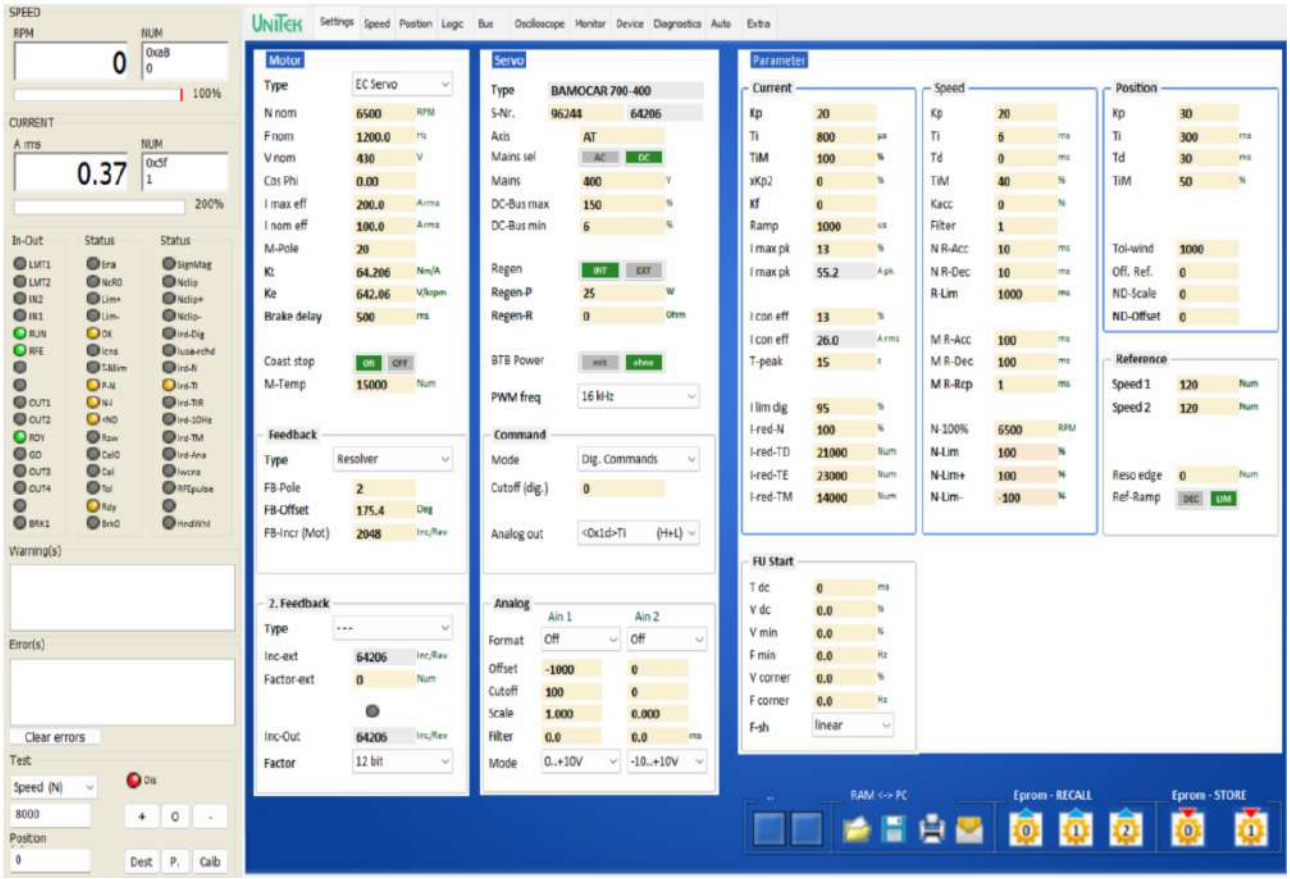


Figure 4: NDrive settings. February,2023

2.4 Controller Settings

A number of different parameters were tested to determine if any would have an appreciable effect on dynamic performance. The settings tested were:

- I Max Peak: Devices Peak current limit [A]
- T peak: Permitted overcurrent time above continuous current limit.
- I Conn Eff: Devices effective Continuous current limit [Arms]
- Limiting fault code states for the following conditions:
  - Icns: Current limit reduced to continuous current.
  - Ird-dig: Current limitation via switch.
  - Iuse-rchd: Current reduction limit reached.
  - Ird-N: Current reduction via speed.
  - Ird-Ti: Current reduction via output stage temperature enabled.
  - IrdTiR: Current reduction to continuous current via output stage temperature is active.
  - Great10Hz: Current reduction deactivated at a rotation frequency higher than 10 Hz.
  - IrdTM: Current reduction via motor temperature.
  - IrdAna: Current reduction via analogue input (if $\leq$90%).
  - Iwcns:Current peak warning.

Functionality had to be designed and implemented to be able to extract the fault code data from the controller via CAN. The existing controller file, created by G.Horsnell, was modified, as seen in listing 1. The completed code file is included in appendix 8.

After track testing changes to these parameters there was no appreciable change in controller performance, nor was the controller de-rating to a continuous current limit at any point.

Listing 1: Delimiting Fault Code in C.

```
//Addition: Reads the fault codes for the current derating
// that the controller has been set to do
// will output the codes contained in the address 0x40- 4 bytes of data
// each bit corresponds to a fault state. only bits 5, 20-28 are relevant (10 total bits)
else if(Message_in2.buf[0] == 0x40){
// bits that you care about
int bits[10] = {5,20,21,22,23,24,25, 26, 27, 28};
// a place to store the 1s and 0s
int vals[10];
// loop over bits we care about
for (int i=0; i<10; i++) {
  // printf("bit = %d, ",bits[i]);
  // find out which byte bit belongs to (0 indexed)
  int byte = bits[i] / 8; // integer division
  // printf("byte = %d, ",byte);
  // find out what position in the byte this corresponds to (0 indexed)
  int rel_pos = bits[i] - 8*byte-7;
  // printf("rel pos = %d, ",rel_pos);
  // get the bit value by bit shifting and masking
  vals[i] = (Message_in2.buf[byte] >> rel_pos) & 1;
  //  printf("bit val = %d, \n",vals[i]);
}
// Bit 5 of 0x40 - Icns
*outputVar2[7] = vals[0];
// Bit 20 of 0x40 - Ird-dig
*outputVar2[8] = vals[1];
// // Bit 21 of 0x40 -  Iuse_rchd
*outputVar2[9] = vals[2];
// Bit 22 of 0x40 -  Ird-N
*outputVar2[10] = vals[3];
// Bit 23 of 0x40 -  IrdN
*outputVar2[11] = vals[4];
// Bit 24 of 0x40 - IrdTiR
*outputVar2[12] = vals[5];
// Bit 25 of 0x40 - Great10Hz
*outputVar2[13] = vals[6];
// Bit 26 of 0x40 -  IrdTM
*outputVar2[14] = vals[7];
// Bit 27 of 0x40 -  IrdAna
*outputVar2[15] = vals[8];
// Bit 28 of 0x40 -  Iwcns
*outputVar2[16] = vals[9];
```

After Completing a gradient calculation based on the power output, shown in figure 5, against the Emrax 188 rpm vs power output graph, shown in figure 6, peak power is being achieved throughout the course of an acceleration run.

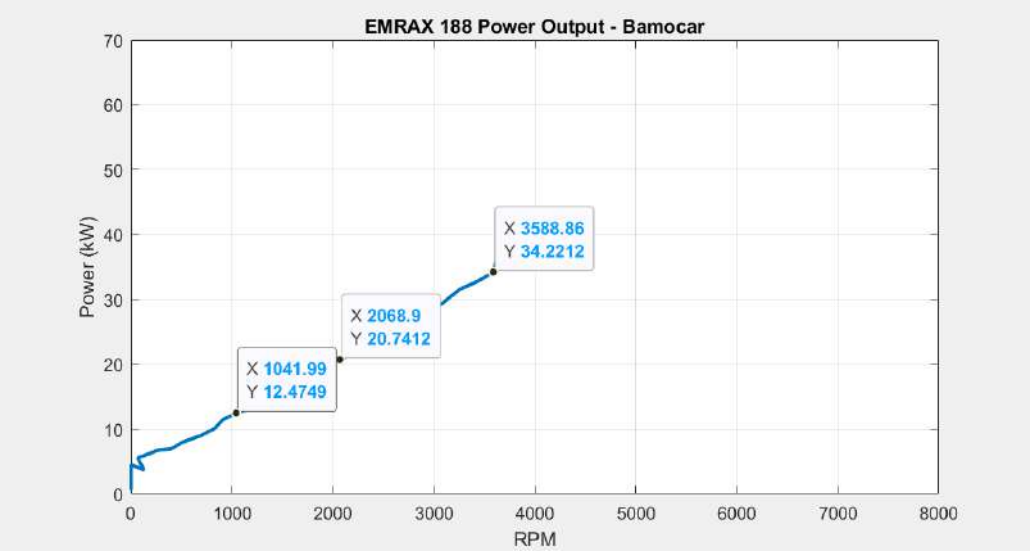


Figure 5: Power Output of NU23 from track day data

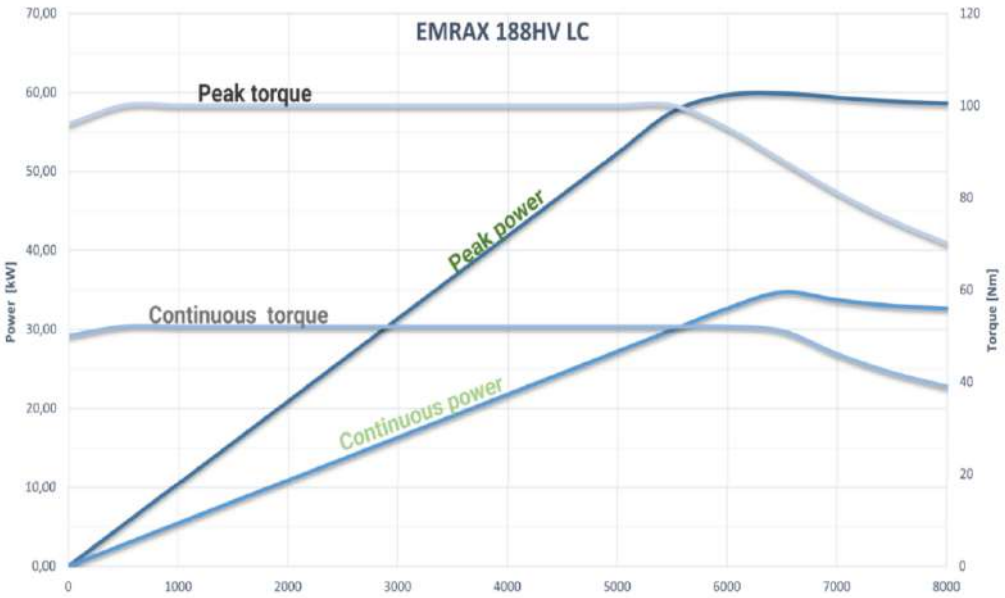


Figure 6: Emrax188 HV Power & Torque vs RPM Graph-Source: Emrax 188 Technical Datasheet

2.5 Field Weakening

Field weakening is a control method that involves intentionally reducing the strength of the magnetic field in the motor's stator, enabling the rotor to spin at speeds beyond the motor's nominal rating. In the context of EVs, field weakening extends the speed range of the electric motor, allowing for efficient operation at high speeds. By adjusting the stator current frequency, EVs can achieve optimal energy utilization and an extended driving range. The Ndrive Manual includes a graph, seen in figure 7.

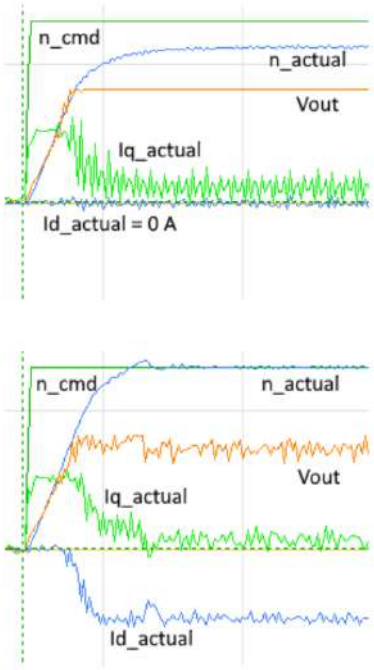


Figure 7: Field Weakening graph

Figure 8: Legend for Figure 7

$n_{\textbf{cmd}}$: Requested speed value of the motor.
$n_{\textbf{actual}}$: True speed value of the motor.
$V_{\textbf{out}}$: Current voltage output of the motor.
$i_{q_{\textbf{actual}}}$: Real current (active torque) applied to the motor.
$i_{d_{\textbf{actual}}}$: Magnetizing current applied (reactive torque) to the stator to counteract the current being induced by the motor.

Without field weakening, as seen on the top graph of figure 7, n_cmd is never reached by n_actual as the motor reaches its maximum speed. Field weakening allows this maximum speed to be increased by inducing a magnet field that counteracts the field generated by the motor, as shown in the second graph of figure 7. The limiting factor for the Emrax 188 is that its maximum motor rpm is mechanically limited to 8000 rpm. Only 1 track day was completed where field weakening control was implemented on the Bamocar controller. The graph, figure 9, details that it was possible to reach speeds greater than those normally allowed by the controller.

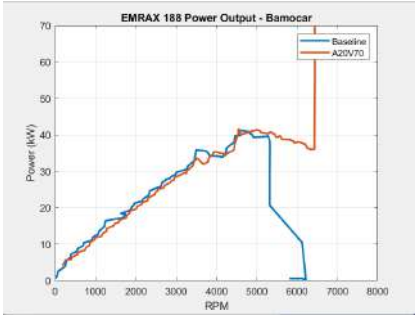


Figure 9: Field Weakening implementation on the Bamocar Controller

Figure 10: Legend for Figure 9

Baseline: The baseline data from an acceleration run with no field weakening control.

A20V70: Data from an acceleration run with field weakening control active after V_{out} reaches 70% of its maximum, creating a setpoint for $i_{d_{\text{actual}}}$ at 20% of its maximum.

Unfortunately an accident occurred in the TA workshop while performing a Start-up procedure after charging the accumulator. Details of this event can be found in Current Locking Fault2.6.1 Due to this event, it was determined that field weakening, while not solely responsible for this accident, did contribute to the motor being capable of reaching speeds that could induce mechanical failure. This resulted in no further experimentation with it for the remainder of the year because of safety concerns.

While determining why the speed limit set in the Ndrive software, N_Lim, it was discovered that setting the speed limit to 100% of N-100%(in RPM) does not mean that the speed limit is equal to the value of N-100%. Doing this will turn the speed limit off and allow the motor to spin as fast as it is electrically capable. To account for this N-100% and N_Lim, was set to 87% of 8000RPM implying the actual speed limit set by the controller is 6960 RPM. This method was verified by reducing the effective motor speed limit to 300rpm and verifying a rotation of 1 revolution per second at the wheel upon visual inspection.

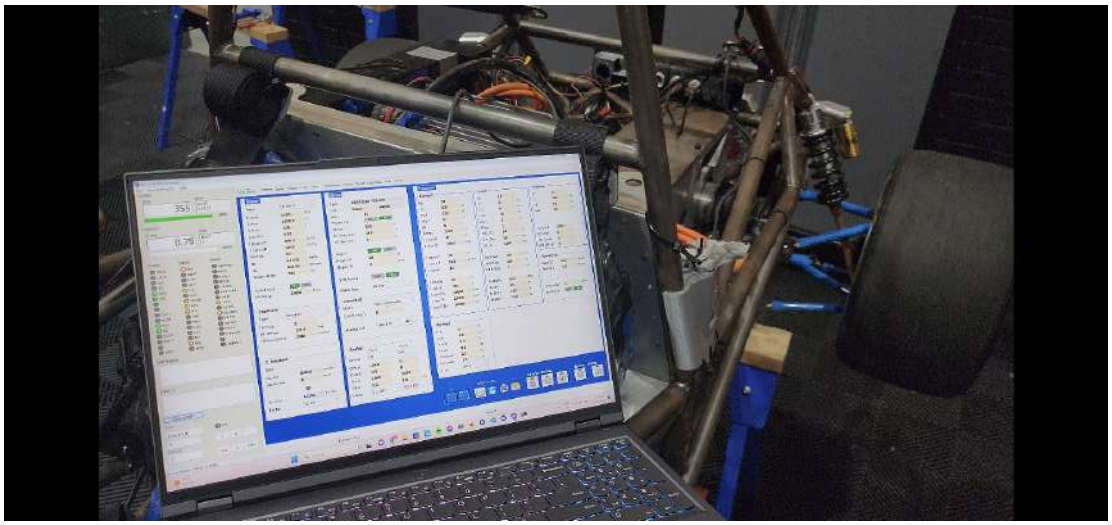


Figure 11: RPM limit Test

2.6 Troubleshooting

The following describes the main issues encountered with the Bamocar controller over the course of 2023.

2.6.1 Current Locking Fault

An error started occurring, seemingly at random, where the controller would lock in a maximum current output state. The driver had to E-stop or trigger the BSPD, causing a hard fault state. Unfortunately this occurred during a workshop startup test where NU23 was on stands off the ground; effectively under no load conditions which has the potential to cause both electrical and mechanical damage to components of the tractive system. The data indicated that the torque request was only kept on for less than 0.8 seconds, as seen in figure 12, but the motor controller continued to draw a maximum current. This resulted in the motor rpm going beyond the data range's detectable maximum value, 1800 to 6500 rpm, in under 100 ms and continued increasing to an unknown speed. Luckily Jye Hollier managed to engage an e-stop in under 2.5 seconds, stopping any further damage to NU23. The motor experienced mechanical failure; likely implying it reached a speed greater than 8000rpm. This was supported by the manufacturer in email communications facilitated by J.Bush[2], details of which can be found in his report.

Figure 12: MoTeC Data from Locking Fault event 21/08/2023

2.6.2 Error 9:Overcurrent Fault

A motor controller error code, error 9: Overcurrent, became a reoccurring issue once the original Bamocar controller, labelled controller 1, was replaced with the spare Bamocar controller, labelled controller 2. This was originally integrated in an attempt to mitigate the current locking issue with controller 1 in October of 2023 by J.Bush[2]. A more detailed description of this issue and its solutions can be found in his report.

2.7 Cascadia Motions CM200DZ

Given that this overcurrent error inconsistently persisted through October and November track day testing with the combined effort of the entire team being unable to pinpoint a failure mode, the option of purchasing and integrating a Cascadia Motions CM200DZ was proposed by A.Gregg. This was unanimously agreed upon by all members of NU-Racing. This decision implied that we would only have at most 3 chances for track day testing assuming it worked directly following its integration.

The CM200DZ, seen in figure 13, was chosen because of its robust software design that requires minimal setup and tuning by the user. Cascadia Motions has already created general tuning profiles for a number of popular EV motors, including the Emrax 188 that NU-Racing uses. Below is a shortlist of design features; for more detail on the CM200DZ's specifications, see appendix 7.

- RS232 Programming and Diagnostic Connection
- Integrated DC-Link EMI Filter
- Operating voltage $200 - 840V_{DC}$ at $400A_{peakRMS}$
- ISO20653 high-pressure wash rated IP6K9K / IP67
- Easy to use CAN-based network node with Custom .dbc messaging
- Extensive feedback broadcast messaging for datalogging
- PC-based setup and programming tools available free
- Robust, fault-tolerant IGBT power stage
- Variable PWM Rate

The software implementation was handled by J.Bush[2], while the mechanical mounting and HV cabling was completed by J.Hollier[4], further details of both can be found in their reports. The author commissioned the motor controller programming cable and LV loom.

Figure 13: Cascadia Motions CM200DZ

2.7.1 Motor Controller LV Loom

The controller utilises a Molex CMC 48 way connector, shown in figure 14. a pinout document, generated by J.Bush, is summarised in figure 15. The procedure for making this loom is as follows:

- Measure the length required for the loom.
- Cut the appropriate number of wires at appropriate gauges. Power connections are 16AWG, signal are 22AWG.
- Strip, crimp each wire with the appropriate size crimp. CP 0.6 size for signal and CP 1.5 size for power.
- Cut Raychem heatshrink to length.
- Concentrically twist wiring. see P.Gleeson's thesis entitled *Formula SAE - Low Voltage Systems Design, Manufacture, and Testing*[3] for a procedure on how to create a concentrically twisted loom.
- Verify each connection is correct with a DMM.
- Shrink Raychem with a Makita heat gun.

Figure 14: Molex 48-Way Connector

Pin	Description
M1	Power
M2	Power
A2	CAN High
B2	CAN Low
F4	Serial- Ground
G4	Serial- RS232_RXD
H4	Serial- RS232_TXD
B1	Resolver - SIN+
A1	Resolver - SIN-
D1	Resolver - COS+
C1	Resolver - COS-
E1	Resolver - EXC Out
F1	Resolver - EXC In
G1	Resolver - Shield GND
C4	Sensor pin
A4	Ground
K3	Power
C4 to K3	3.01k resistor bridge

Figure 15: CM200 LV Loom Pinout

2.7.2 Motor Controller programming cable

The Cascadia Motions CM200 controller needed a different programming cable as it needs an RS232 connection. a RS232 breakout connector was used, shown in figure 16, because a switchable ground connection is required to update firmware on the controller. The pinout is shown in figure 17.

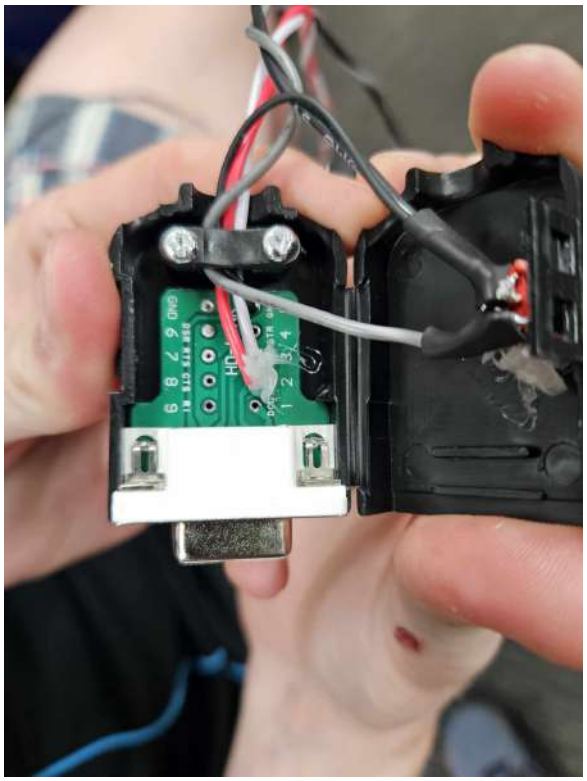


Figure 16: Serial connector modification

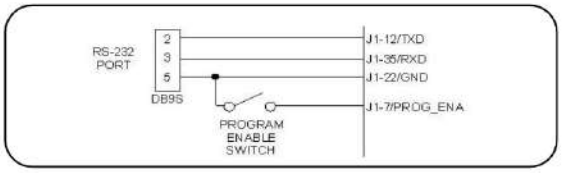


Figure 17: Controller Programming loom pinout

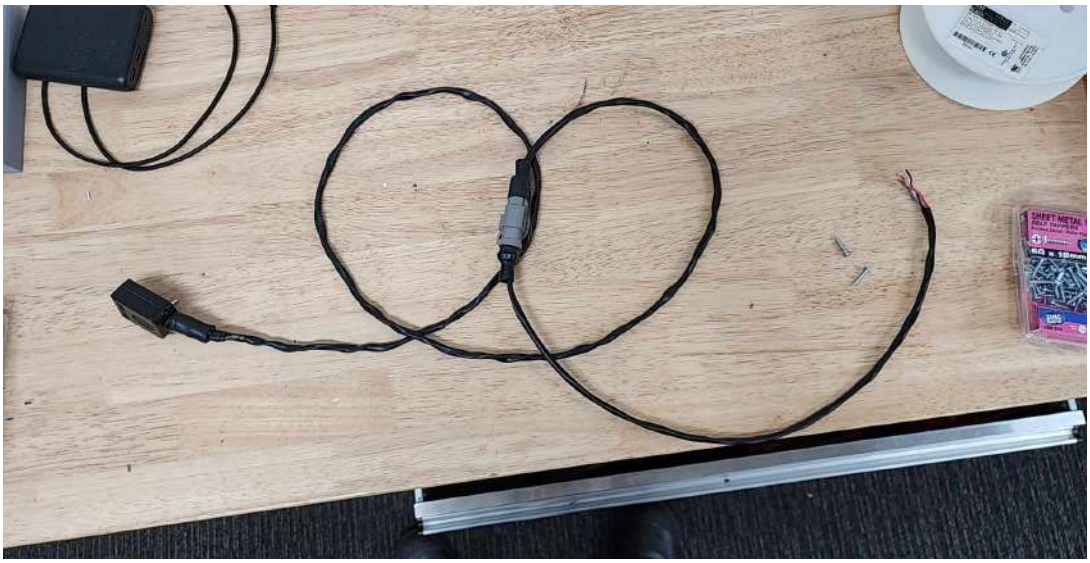


Figure 18: Fully assembled serial connector to 3 pin DT connector Loom

2.7.3 Commissioning and Testing

The Controller was received on the 4thDecember, 10 days before the planned date to travel to Melbourne for the Competition. Within 24 hours from receiving the controller and wiring, NU-Racing was able to integrate and validate its operation at SMSP. The track day was completed with the controller working exactly as intended with improved performance in comparison to the Bamocar, as seen between figures 19 and 20. These represent acceleration run times and data from the 2022 competition and the final SMSP track day This accomplishment underscores the team's adept understanding of NU23's design, the implementation of modular systems, and the proficiency in system integration, highlighting the team's comprehensive engineering capabilities.

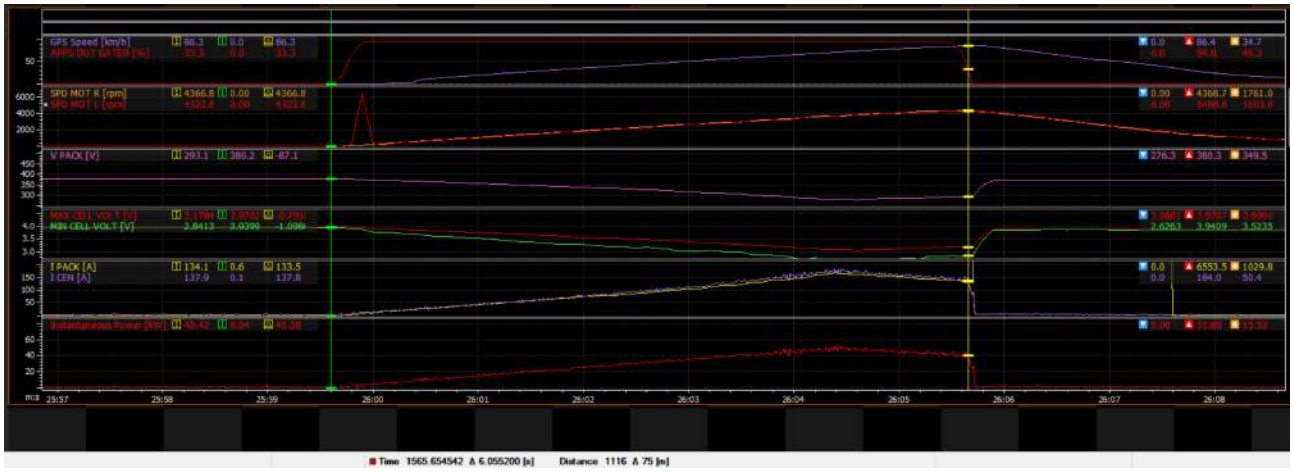


Figure 19: Acceleration Run from 2022 Competition

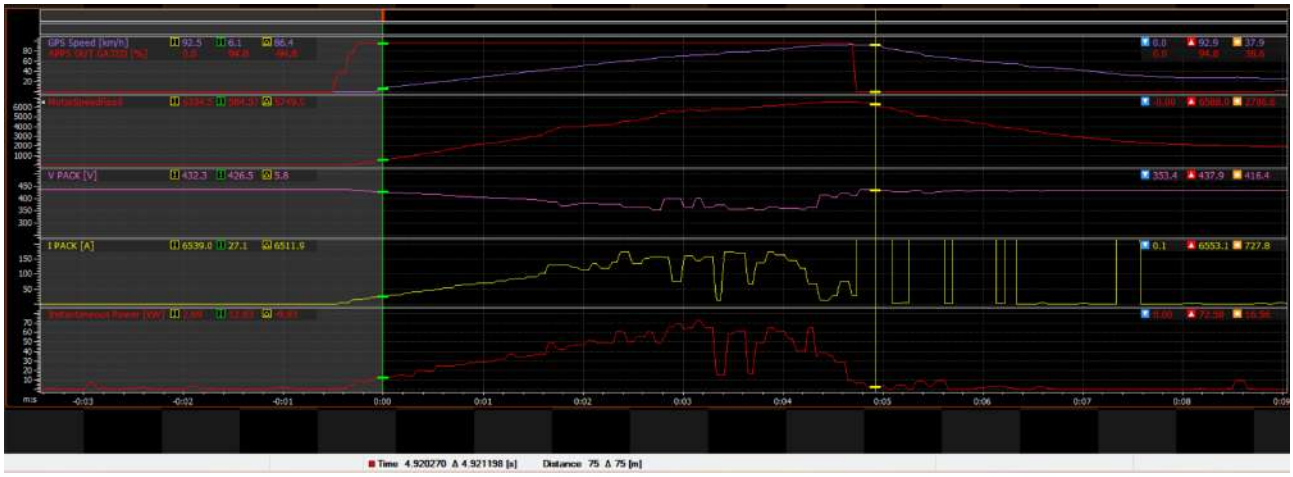


Figure 20: Acceleration Run from final SMSP track day

2.8 Competition Performance

At the 2023 FSAE Competition, the Motor Controller did not cause any issues or inhibit the performance of NU23. The performance of the controller was a significant factor to NU-Racing placing 2nd in the Skidpan and Autocross dynamic events. Our performance in the acceleration event was the best result ever achieved by a NU-Racing EV, 4.45 Seconds.

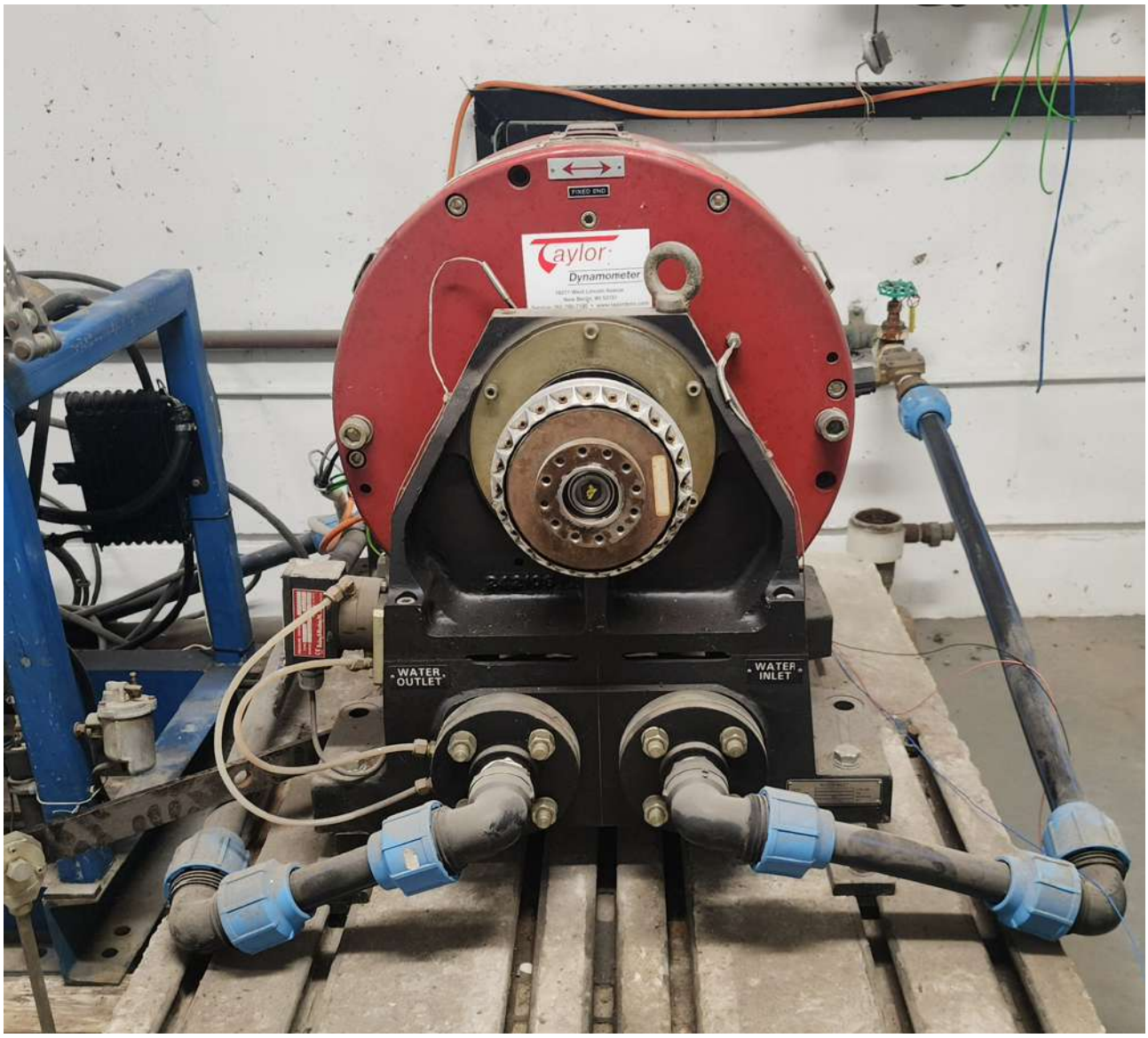
2.9 Recommendations

There will be overlap in recommendations between this report made by the author and those in J.Bush's and J.Hollier's reports pertaining to the controller design. All recommendations should be considered before making any design changes.

Design recommendations for the motor controller are as follows:

- Hardware:
  - The Molex pins used for the controller's LV loom are different than standard Micro and Mini Molex pins normally used. A procedure for correctly crimping these pins with the use of a standard Molex crimp tool should be created.
- Software:
  - The software implementation was very rushed to get the controller functional for the competition. It is entirely possible that there are several safety features that are not operational as we were unable to accurately test them. It is recommended that the software is completely revised, and commissioning test performed to ensure the controller is working as designed.
  - There are also a number of features that the controller is capable off that have not been used. The full capabilities of the controller should be explored for performance improvement.
  - The controller currently factors controller, motor and coolant temperature into its current de-rating algorithm. There are capabilities for the maximum battery temperature to be used in this equation, over CAN communication. Doing this will virtually make it not possible to trigger an over temperature hard fault. A downside of doing this could be that the controller starts de-rating too early, impacting overall performance but that is unlikely to be the case.

3 Implementation of an Eddy Current Dynamometer



3.1 Introduction

This section details the feasibility of using the existing Dynamometer in the basement of the EC workshop building, recommissioning it such that it would be usable for Nu Racing's needs and further recommendations with how to make it a useful tool.

There has been no method that allows for a simple, efficient way to test the HVTS under load system of any electric vehicle produced by the university of Newcastle, nor is there a way to safely discharge the accumulator in a timely manner without organising a track day or individually discharging unbalanced cell stacks using the fixed electronic load available in the NU-Teams Laboratory. The option of repurposing the existing Taylor DE150 Engine Dynamometer was proposed by Dr Alexander Gregg and Dr McBride, as it was previously used for internal combustion Engine (ICE) testing but had not seen consistent use for over 5 years.

3.2 Taylor DE150

The Taylor DE150 is an eddy current dynamometer. This class of dynamometer operates on the principle of electromagnetic induction to measure and absorb mechanical power produced by an engine or motor during testing. This dynamometer consists of two essential components: a rotor connected to the device being tested and a stator connected to a power source. The rotor, typically made of a conductive material, rotates within the magnetic field generated by the stator, inducing eddy currents in the rotor due to the changing magnetic flux. These eddy currents create a resistive force that opposes the rotor's motion, generating a braking effect. The magnitude of this force is proportional to the rotor's speed and the strength of the magnetic field. By varying the magnetic field intensity, the dynamometer can control the amount of resistance applied to the rotating rotor, allowing for precise measurement and adjustment of mechanical power output. The electrical power required to maintain the desired speed against the resistance provides a direct measure of the mechanical power produced by the tested device. The specifications of the Taylor DE150 are as follows:

Table 1: Taylor DE150 Specifications

Parameter	**Value**
Power	201 hp (150 kW)
Torque	369 lb-ft (500 Nm)
Speed	12,000 rpm
Water Use	28 gpm (106 lpm)
Inertia Value	2.21 lb-ft^2 (0.093 kg.m^2)
Shipping Weight	1753 lb (795 kg)
Rotation	Bi-directional

The above limits are well above the design limitations from the FSAE rules requirements, listed in table 1. Any future motors used will be able to be tested using this dynamometer.

The full specification sheet can be found at:

https://www.taylordyno.com/wp-content/uploads/pdfs/SMS2028-DE150.pdf

3.3 YourDyno Control Module

The YourDyno Control Module is an off the shelf device, manufactured by a small engineering company based out of Norway. It is designed to control the brake force applied to a dynamometer, take in generated data and save it in a format where it can be viewed and manipulated within the provided software interface. While this device is essentially just a micro-controller kit, the utility of the software interface does make it simple to graph and tabulate the parameters that are useful. The software is P.I.D. control based and is capable of programming the output to a Dynamometer as well as inputs to a motor, i.e. speed or load control.

Figure 21: YourDyno Module

The installation manual for the YourDyno module can be found at:

https://yourdyno.com/wp-content/uploads/2023/02/YourDyno-Standard-Instrument-kit-instructions.pdf

3.4 Setup Modifications

The Dynamometer room was setup for usage with I.C.E. systems where much of the hardware and cabling was either not needed or would need to be shortened. All the systems in place were designed so that the operator would be outside of the room during operation because of the fumes generated during the I.C.E. systems operation. A list of the tasks is as follows:

- The blue steel piping that was originally in place for fume extraction to overhead ventilation was removed.
- The ICE that was originally in place, setup on the Taylor, was safely demounted and was moved onto a pallet next to it, as seen in figure 23.
- The steel table that was originally used for a desktop computer to control the L.V. electronics was moved inside the room to accommodate anyone with a personal laptop, shown in figure 24.
- The YourDyno control module, seen in figure 25, was removed from its enclosure outside the room and setup inside on the table. As the maximum voltage running through the device is 5V, it does not require an enclosure as it doesn't pose any safety concerns.
- The H.V. power electronics enclosure that both powered and controlled the Taylor originally used a AC wave generator for its 5V input outside the room. This enclosure was moved inside the room and planned to be mounted to the concrete wall using its original brackets, above the table for ease of access as it is connected to the control module that requires a computer to run. This can be seen in figures 30 and 31. The HV transformer is also planned to be mounted to the wall beside the HV enclosure, shown in figure 29.
- The cabling for all previous ICE modules was removed and only the minimum required wiring is left, as seen in figures 26 and 27.
- The RPM sensor, shown in figure 28, had to be tested in isolation to determine wiring polarity before being connected to YourDyno module. this was completed by connecting a power supply, spinning the freely rotating shaft of the dynamometer and measuring pins with DMM until the correct pin was determined.

All electronics and electrical connections where high voltage was present were checked by Malcolm Sidney and were verified to be of a safe standard.

Figure 22: Outside of the dynamometer room post clean up.

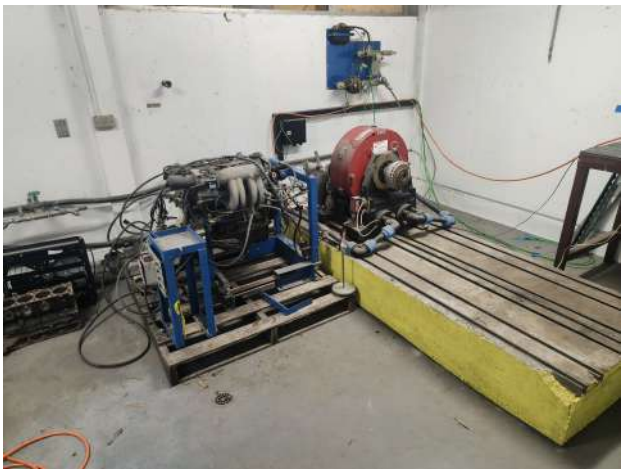


Figure 23: Engine placement.

Figure 24: Steel table and placement of components.

Figure 25: YourDyno Module with wiring attached.

Figure 26: Wiring setup of the dynamometer system.

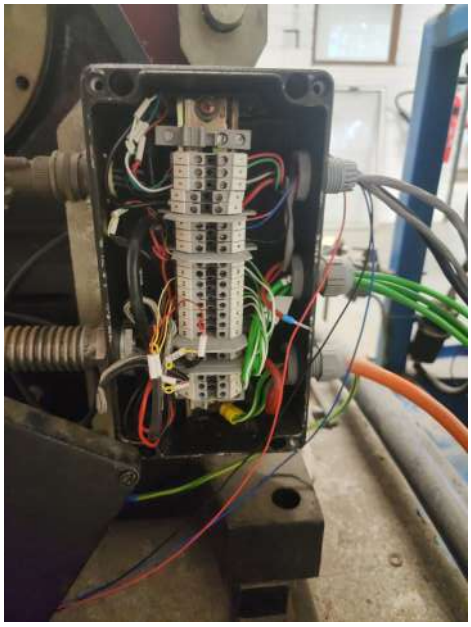


Figure 27: Additional view of the wiring configuration in the dynamometer system.

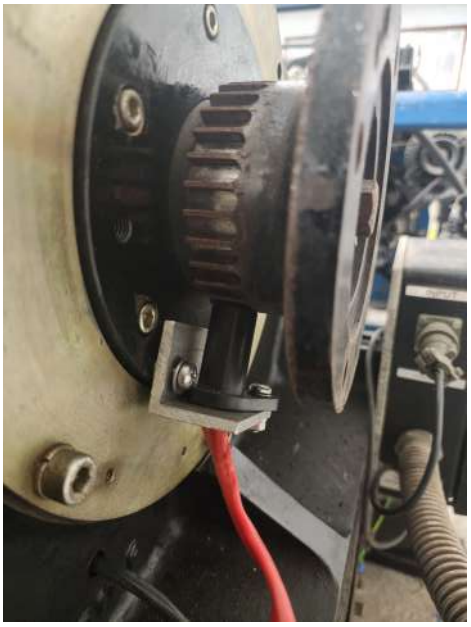


Figure 28: RPM sensor used on the dynamometer.

Figure 29: High-voltage transformer for the HV electronics enclosure.

Figure 30: HV electronics enclosure.

Figure 31: Internal components of the HV electronics enclosure.

Figure 32: Emergency stop enclosure.

Figure 33: Internal view of the enclosure.

3.5 Dynamometer Run Experiment

A electrical safety check was performed before running the dynamometer for the first time. The testing was carried out in conjunction with Malcolm Sidney as a safety supervisor. The order of testing was as follows:

Functional Test	**Pass/Fail**
Earth signal is not connected to any power input or output lines. all enclosures, including the Dynamometer, pass earth continuity.	Pass
All connections inside the high voltage electronics enclosure are securely fastened.	Pass
All connections on the YourDyno module are securely fastened.	Pass
While mains power is turned on, check earthing on all enclosures	Pass

Table 2: Functional Testing Procedure

3.6 Calibration

On an eddy current dynamometer, a load cell is used to regulate the amount of brake force applied to the output shaft that is connected the motor's output shaft on top of the inherent static inertia of the Dynamometer itself. This regulation is not fixed and therefore requires a control loop to regulate to a desired brake force. The YourDyno control module uses a P.I.D. controller for this and requires the P.I.D parameters to be tuned. To ensure the greatest accuracy for the brake force, the parameter tuning should be tested at an RPM roughly equivalent to the average rpm of the motor being used. For the Dynamometer to be as modular as possible, creating an informative and accurate tuning procedure when this process is undertaken will exponentially decrease the tuning time between different load testing experiments. Keeping an accurate log of tuning parameters for different load and RPM conditions will be very useful.

3.7 Water Cooling System

Eddy current dynamometers generate heat during their operation that needs to be continuously cooled. There is an existing pump system that is still attached to the Taylor however it has not been testing if it is functional, as seen in figures 34 and 35. There is still water from the building going to this outlet as it was leaking. There is a 12V pump that is powered through the Taylor's electronics node; it should be possible to control it with the YourDyno control module on a G.P.I.O. pin. Caution should be taken when attempting to turn the pump on as there is potential for leaks; appropriate measures should be taken.

Figure 34: 12V pump motor

Figure 35: Water piping

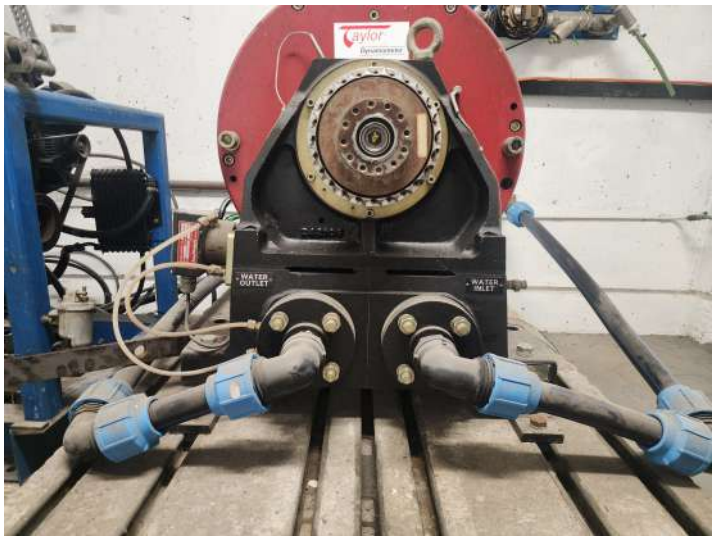


Figure 36: Piping to the front of the Dynamometer

3.8 Recommendations

3.8.1 Safety requirements

The following is a list of safety requirements that should be followed while implementing any solution

- The motor's emergency stop functionality needs to deactivate the dynamometer as well. This could be done through coding but the most ideal scenario would be having the same signal being used for both the dynamometer and the motor. This could be achieved through a shutdown circuit connecting multiple e-stop switches in series, similar to NU23.
- There is a 'blast shield' that, in the event of a severe mechanical failure, ensures protection for individuals in the vicinity by preventing the release of rotating components that could pose a risk of injury. This needs to be factored into the motor mount design by either using the current one or designing a new one to accommodate.
- the mounting system for the powerbox and attachment to the dynamometer shaft should always be alignment checked before it is in operation. the potential risk for misalignment with regular switching is high. designing a self aligning mounting system would be ideal
- the dynamometer should never be left to run by itself. high power devices should always be watched while in operation.

It is highly recommended to complete a risk assessment at the concept stage of development as the recommendations in this section may not cover all possible risks and hazards.

3.8.2 Motor Mount

to attach any E.V. motor to the dynamometer's output shaft, a modular mounting system needs to be created. This mounting system needs to account for the following:

- modular Powerbox mounting alignment that allows for changes in future design without having to redesign the mounting frame.
- given that there is only a single operational electrical system for load testing and running an E.V. the powerbox will be moved between load testing and vehicle operation. creating the alignment system in such a way that it is self aligning upon installing or a well documented alignment method that is simple and short to complete will be the optimal solution.
- all electrical cabling from the powerbox to the controller and accumulator, whether inside or outside the room.

3.8.3 Load Testing Node

Currently the only way for the motor to operate is having it fully implemented on the NU23 system with all nodes attached and operational. It is possible to spoof all the electrical safety parameters and only use the C.E.N.; this would still require removing it from the car as there is no spare at the moment. A more long term solution would be creating a new Node that is designed with all the current safety features and high voltage paths of the C.E.N while modifying its current functionality

to be useful for load testing inputs. The following is a list of suggestions for functionality and a general concept of how it could be implemented:

- Keep the current low voltage, TSMS and HFR switches in place for safety
- the L.V. and H.V. Accumulator ports as well at least one, if not more, of the expansion ports should remain for CAN probing.
- Add extra switches and/or potentiometers as inputs for throttle, throttle enable, etc

The existing enclosure for the Dynamometer E-stop and 12V transformer could be modified to fit this electrical node.

4 NU23 Charger

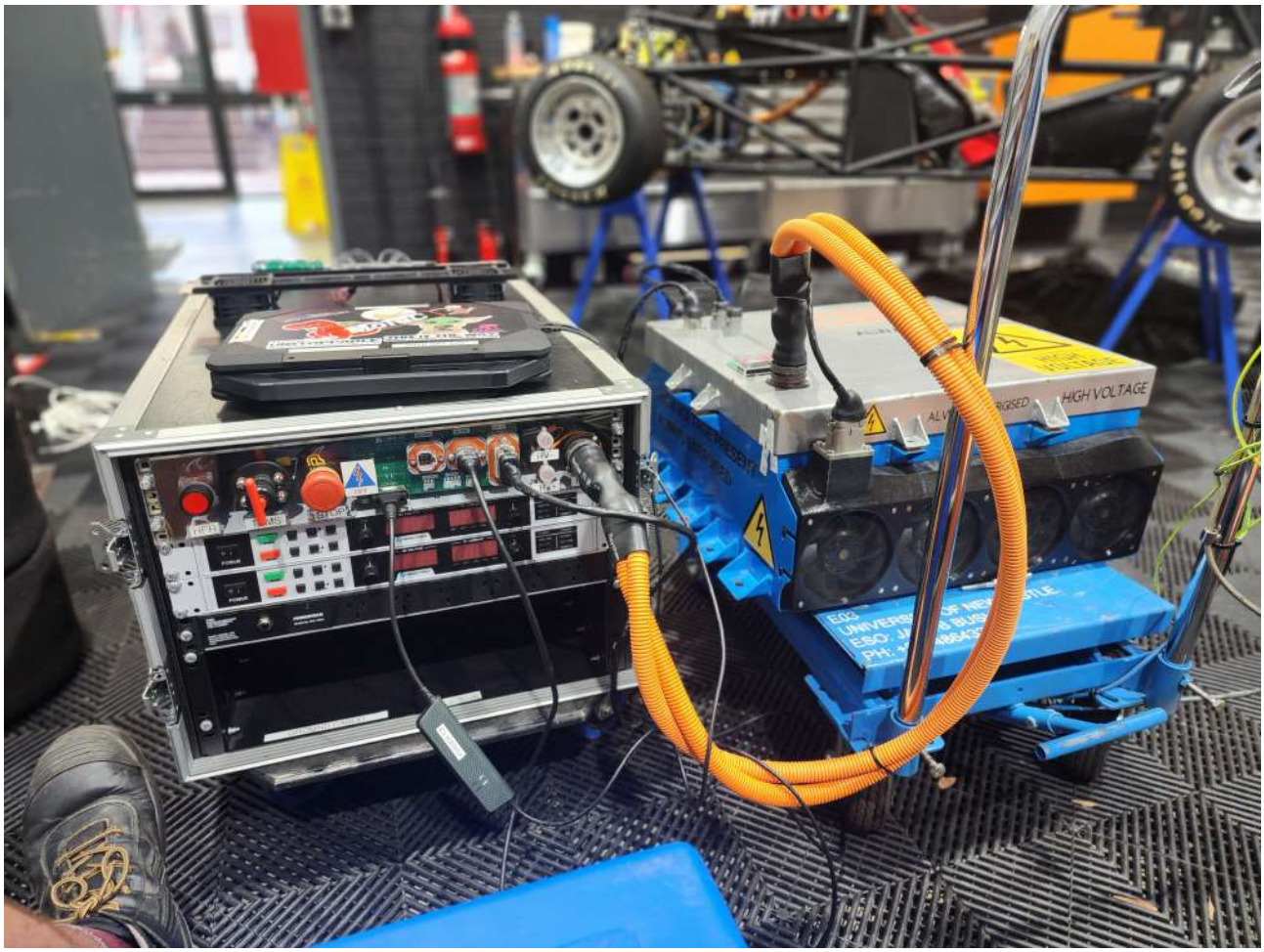


Figure 37: Charger Setup for charging

4.1 Abstract

The EV3 charger, while completely functional, lacked easy troubleshooting capabilities and was limited in its Low Voltage (LV) power supply. The modifications to the accumulator cooling design drastically increased the LV power requirements; this could not be supported by the current Meanwell 3 phase AC to DC converter at full power. The EV3 charger also lacked any capability to charge a device from a standard single phase 240V outlet. To be able to accomplish this, the 3-phase connection needed to able to use single phase power; requiring a 5 pin, 3-phase connection over the current 4 pin connection. To accomplish these goals while complying with the 2023 FSAE - Australasia rules the charger was modified to accommodate these changes while retaining its functional HV capabilities.

4.2 EV3 Charger

The EV3 charger was designed by X.Young. For more details about the EV3 charger, please refer to X. Young's thesis entitled *Formula SAE - HV and LV Systems Implementation*[7]. The notable design features include the following:

- Dual HV Magna-Power SL series power supplies in power capable of 500V, 16A, 8kW.
- Meanwell WDR120-12 120W 415VAC to 12VDC converter, seen in figure 39.
- The Charger ECU was designed by D.Birdsall, seen in figure 38, featuring an automated charging functionality. For more information about the automated functionality, please refer to D.Birdsall's thesis entitled *Energy Storage and Tractive System Design for an Electric Vehicle*[1].
- Aluminium face plate with panel mounted switches, LV and HV connectors and, HV test points

This design was compliant in accordance with the 2022 competition rules, however changes to the rules were made for the 2023 competition.

Figure 38: D.Birdsall Charger ECU

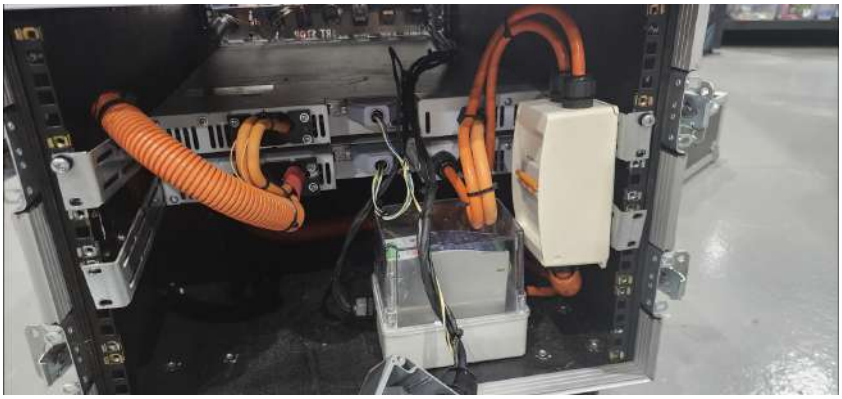


Figure 39: Open Rear view of EV3 Charger

4.3 FSAE Rule Requirements

In prior years of the FSAE competition the tech inspection regarding accumulator and charger design was not as stringent. Teams that were sufficiently prepared with all relevant supporting material such as datasheets and visual aids to portray compliance passed without any issues. The rules were updated this year to require tech inspectors to go into meticulous detail requiring teams to open and visually inspect components for compliance. The rules requirements pertaining to the charger design are as follows:

AI.4.1 Charger needs to be professionally built. All connections must be insulated and covered. No open connections allowed.

AI.4.2 Battery Charger Power lead must be free of damage, nicks, cuts, grease, or grime. Plug shall be suitably rated and conform to EN 60309-2, AS/NZS 3123. Charger must have a valid test and tag sticker or tag attached.

AI.4.3 HV wires must be marked with gauge, temperature rating and voltage rating, serial number or norm is also sufficient, if the team shows the datasheet in printed form.

AI.4.4 Wire temperature and voltage rating must be suitable for use in the charger.

AI.4.5 Using only insulating tape or rubber-like paint for insulation is prohibited.

AI.4.6 The AC power supply to the battery charger and other associated devices must include a residual current device (RCD) with over current protection (fuses or an appropriate circuit breaker) or residual current circuit breaker (RCBO). The RCD or RCBO device must act to disconnect both the active and neutral supplies. The trip sensitivity of the RCD must not exceed 30mA. Where possible 10mA is preferred.

AI.4.7 Traction System charging leads must be orange.

AI.4.8 Two charging system voltage measuring points must be provided on the charger output lines.

AI.4.9 The measuring points must be protected from being touched with the bare hand/fingers once the housing is opened.

AI.4.10 4mm shrouded banana jacks rated to a voltage higher than the maximum tractive system voltage must be used.

AI.4.11 The Charger System Measurement Points must be protected by body protection resistors.

AI.4.12 The Charger System Measuring Points must be marked with HV+ and HV-.

AI.4.13 The charger must include a push-type emergency stop button which has a minimum diameter of 25mm and must be clearly labelled. The Pushbutton must be easy to reach with the accumulator in place.

AI.4.14 When the E-stop is pressed, the output from the charger must be electrically disconnected from the battery, and the output from the charger must fall to less than 60 V in 5 seconds.

AI.4.15 The international electrical symbol consisting of a red spark on a white-edged blue triangle must be affixed in close proximity to this switch.

AI.4.16 The battery charger, accumulator and accumulator charging trolley, and any associated metallic components must be equipotentially bonded and connected to the AC power supply earth such

that the RCD function is not impeded. Cables used for earthing must be yellow/green stripped and at least 2.5mm^2 cross-section.

AI.4.17 The Charger DC output must be galvanically isolated from the AC input.

The rules AI.4.1, AI.4.2, AI.4.3, AI.4.4, AI.4.5, AI.4.7, AI.4.8, AI.4.11, AI.4.12, AI.4.13, AI.4.15, AI.4.16, and AI.4.17 remain compliant from EV3's charger design.

The remaining rules, specifically AI.4.6, AI.4.9, AI.4.10, and AI.4.14, will be addressed in the following sections.

The full list of 2023 tech inspection rules can be found at:

https://www.saea.com.au/content.cfm?page_id=2201117&current_category_code=21462

4.4 Proposed Modifications

The requirements for the proposed modifications were:

- Comply with all rule changes for the 2023 competition.
- Accommodate the increased current draw on the LV power supply from the accumulator cooling.
- modify the current Autosport connector CAN communication to a DT CAN expansion port.
- Implement a visualisation of hard fault and shutdown circuit errors.
- Add a front mounted $240V_{AC}$ outlet
- Add an Isolation Relay (IR) to immediately stop the flow of HV in the event of an emergency.

A 5 pin 3 phase HV connection would be implemented as it is not possible to use a single phase on a 4 pin connection due to the lack of a neutral pin. This allows for a wider variety of AC to DC converters to be considered. In accordance with AI.4.6 separate RCBO devices will be used to protect the 240V outlet, 320W LV supply and the Magna-Power supplies. A Hager ADM416T-c16 was used for the 3 phase connection and 2 Hager ADC910T-c10 were used for the single phase connections. The datasheets for these can be found in appendix 7.

Figure 40: Hager ADC416T RCBO

Figure 41: Hager ADC910T RCBO

Figure 42: 5 Pin 3 Phase Cable

The LV power demand from the accumulator is up to 27A @12V with the fans at full speed. The original current limit of 10A allowed the fans to be run at 40% of their capacity with no headroom. To accommodate this the Meanwell RSP-320W-12V 26.7A Power Supply was used. The high current capacity allows the fans to run at near full capacity if required. The datasheet can be found in Appendix 7.

Figure 43: Meanwell RSP-320-12 Supply

The power supply will be mounted under the existing HV supplies on the side wall of the charger. A 3D printed polymer enclosure was designed and printed by J.Wenham to encapsulate the open connections, acting as strain relief and minimising the possibility of an electrical accident.

The autonomous functions of the charger had not been utilised for the entire year. To keep in line with the goal of simplifying design decisions and minimising part count, the autonomous functionality was removed and a new CHARGER ECU was designed, seen in . The goal of this design was to have all the functionality of the Hard fault circuitry, found on the CEN ECU, and a streamlined connection to CAN communication through a DT expansion port. Indicator lights would also be added for power, hard fault conditions and each stage of the shutdown circuit. a ground connection would be added in accordance with AI.4.16, that is equipotentially connected to the metallic components of the charger.

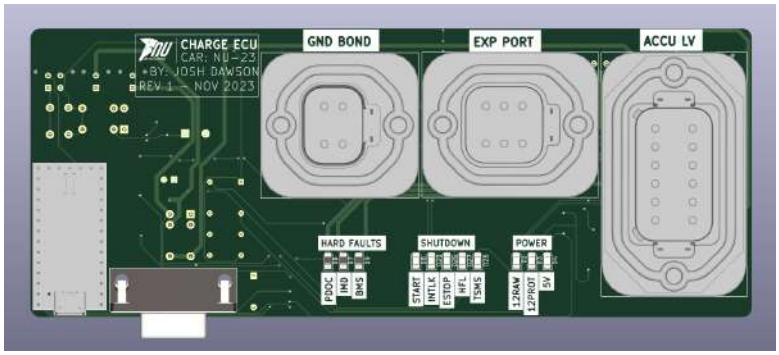


Figure 44: Charger ECU 3D Render - Front View

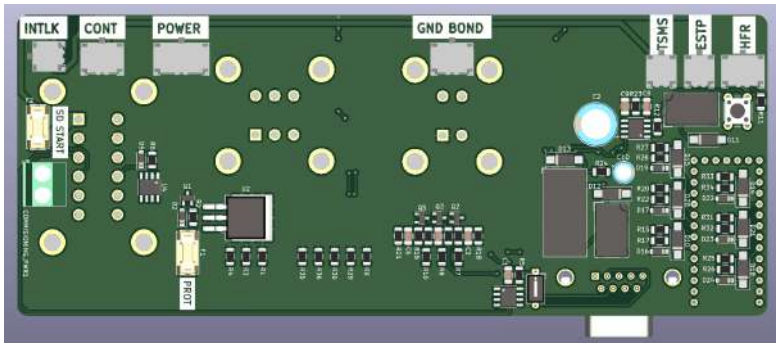


Figure 45: Charger ECU 3D Render - Rear View

The modified mounting plate, seen in figure 46, had the goal of simplifying the design while accommodating the changes in connectors. The following changes were made:

- LV connection changed from an Autosport to a DT.
- E-stop and HFR buttons changed location.
- mounting position was added for the CHARGER ECU.
- ERR and CHRG buttons were removed.
- Start switch changed from a momentary switch to a TSMS, same as what is used on the CEN enclosure.
- Panel material is clear acrylic due to its insulative properties and being transparent allows the user to see the hard fault indicator lights.

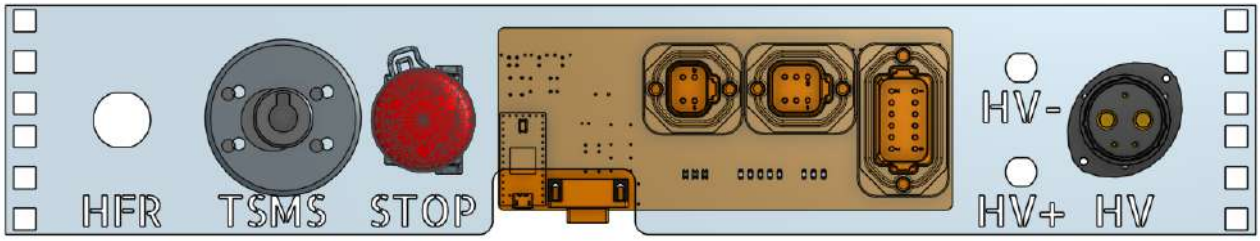


Figure 46: 3D model of Front Mounting Plate

The charger was wired as shown in the charger block diagram in appendix 13.

An Isolation relay would be connected in series between the HV output of the Magna Power Supplies and the HV auto sport connector. This relay is the same type that is used in the accumulator. This satisfies compliance for AI.4.14.

Shrouded banana Jacks with plastic test point covers where also purchased to replace the current normal banana jack test points. This satisfies compliance for AI4.9 and AI.4.10.

4.5 CAD/Schematic/PCB/Code

The CAD models and drawings for the charger (primarily the front panel, LV Supply and PCB) can be found at:

`https://newcastle.onshape.com/ > NU Racing > NU-23 > Charger-23`

The Code used for the charger was completed by Josh Wenham. A general outline of its functions are as follows:

- Every 100 ms:
  - Send a charge command over CAN causing the Precharge ECU to allow for fast charging.
  - Poll and update the shutdown circuit and hard faults messages over CAN using NUCAN_write().
- Run NUCAN_heartbeat() to send messages over the CAN network.

A copy of the code has been included in appendix 14 and can also be found, along with the Kicad project for the PCB and the block diagram at:

https://github.com/NU-Teams/Racing-NU23/tree/main/Charger

The Bill Of Materials for the charger V1 ECU is as follows:

Item	References	Value	Quantity
1	C1, C3, C4, C6, C8	100nf	5
2	C2	220uf	1
3	C9	10n	1
4	C10	470u	1
5	R1, R2, R3, R7, R8, R9, R10, R23, R29, R30, R35, R36	1K	12
6	R13, R14, R15, R16, R20, R25, R27, R31, R33	10k	9
7	R17, R22, R26, R28, R34	4.99K	5
8	R18, R19, R21	7.15K	3
9	R4	500	1
10	R5	120	1
11	R6	100k	1
12	R11	400	1
13	R12	12.4K	1
14	R24	100	1
15	R32	24.3K	1
16	D	D	1
17	D_Zener	D_Zener	1
18	12V_RAW	12V_RAW	1

Table 3: Charger ECU BOM

Item	References	Value	Quantity
19	MMSZ5231B	MMSZ5231B	1
20	12V_PROT	12V_PROT	1
21	5V	5V	1
22	5V1	5V1	1
23	PRE	PRE	1
24	IMD	IMD	1
25	SD_START	SD_START	1
26	BMS	BMS	1
27	INTERLOCK	INTERLOCK	1
28	ESTOP	ESTOP	1
29	TSMS	TSMS	1
30	HFL	HFL	1
31	Teensy4.0-NU-Teams	Teensy4.0-NU-Teams	1
32	LM7805_DPAK	LM7805_DPAK	1
33	TJA1051T-3	TJA1051T-3	1
34	MCP6002-xSN	MCP6002-xSN	1
35	LM555xM	LM555xM	1
36	Branding_Block_Racing	Branding_Block_Racing	1
37	SMDFuse	SMDFuse	1
38	3A	3A	1
39	DSIC01THGET	DSIC01THGET	1
40	HFR	HFR	3
41	2N7002	2N7002	2
42	G5V-1	G5V-1	1
43	COMMISIONING_PWR	COMMISIONING_PWR	1
44	G5V-2	G5V-2	1
45	PZXM61P03FTA	PZXM61P03FTA	1
46	ACC LV	ACC LV	1
47	EXP 1	EXP 1	1
48	DE9_Receptacle	DE9_Receptacle	1
49	Input Supply	Input Supply	1
50	GND Sense	GND Sense	1
51	HFR	HFR	1
52	INTERLOCKS	INTERLOCKS	1
53	SD_ESTOP	SD_ESTOP	1
54	Contactors	Contactors	1
55	TSMS	TSMS	1
56	GND Cart	GND Cart	1

Table 4: Charger ECU BOM Continued

4.6 Manufacturing

4.6.1 HV upgrade

All HV circuitry was mounted and checked by Malcolm Sidney to be in compliance with standard electrical practices.

The RCBO, 3 phase 5 pin connector and cable, aluminium mounting plate were all installed by Malcolm Sidney. The body protection resistors, required by AI.4.11, are attached in series with the HV output of the Magna-Power supplies. Earthing was installed on all relevant points around the charger and connected to the RCBO.

Figure 47: 5-Pin Cable for Charger

Figure 48: RCBO (Residual Current Breaker with Overcurrent Protection) in Charger

Figure 49: Charger Earthing point

Figure 50: Charger Rear panel Wiring

The isolation relay was mounted with HV lugs and Glenloch nuts inside a polymer enclosure with glands for strain relief. This was completed and tested by Malcolm Sidney.

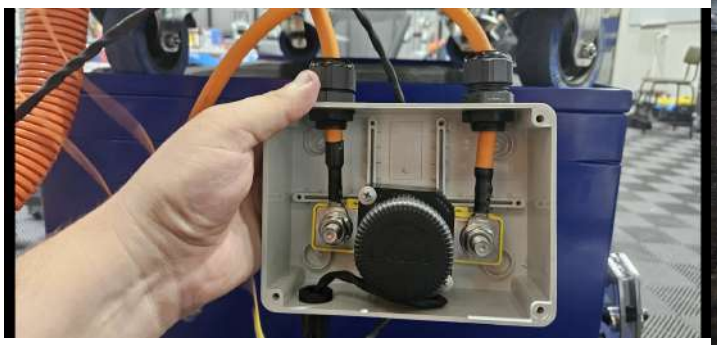


Figure 51: IR enclosure internal view

Figure 52: IR enclosure mounted in Charger

The 1u $240V\_AC$ outlet rack was wired into the relevant RCBO and mounted below the Power supplies. This was completed and tested by Malcolm Sidney.

Figure 53: Front view of the Charger

4.6.2 LV upgrade

Wiring looms were created by Josh Wenham. The LV supply and enclosure were mounted in the charger and tested by Malcolm Sidney.

Figure 54: LV Supply mounted inside the charger

4.6.3 Charger V1 PCB

The charger V1 PCB was assembled by Josh Wenham.

Figure 55: Charger ECU Front View

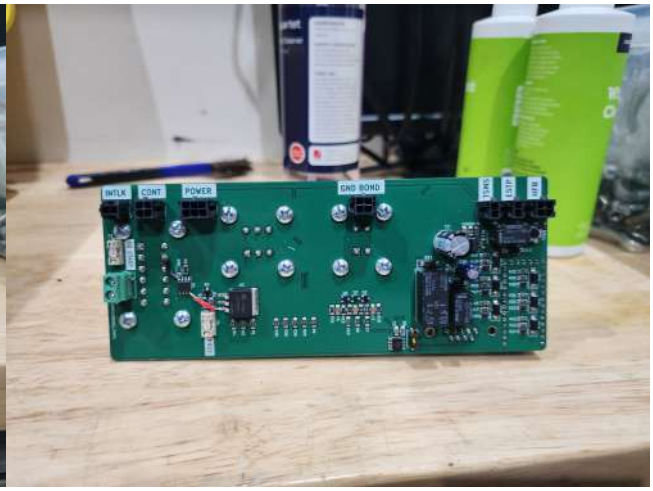


Figure 56: Charger ECU Rear View

4.6.4 Mounting Plate

The mounting plate was laser cut out of a 10mm clear acrylic panel. The mounting holes for the E-stop and HFR button were not large enough in diameter and were machined out to the correct diameter with a drill press. All components were mounted, and cabling routed, as seen in figure 53. The shrouded banana jacks and test point covers were mounted.

4.7 Verification

4.7.1 ECU

The charger PCB was tested in isolation. The hard fault signals and indicator lights were tested by spoofing the signals using 2 power supplies. The indicator lights became active as each hard fault signal was connected.

All stages of the shutdown circuit were tested in order of priority:

- Start: This checks one side of the interlock cable in the HV connector.
- INTLK: This checks the other side of the interlock connector. Both being closed implies that a cable is properly secured to the panel.
- E-STOP: This checks if the E-stop is currently active.
- HFL: Initially this is setup to be closed on startup. once in a fault state, the system can set to a 'healthy' state by pressing the HFL button if there are currently no hard faults active.
- TSMS: Switch for turning the tractive system on or off.

As each stage closed, the indicator light became active.

There was an issue with the U4 operational amplifier where the IC's power and GND were not included into the schematic and therefore did not get routed in the PCB editor. This was fixed by attaching jumper wires between the 5V regulator, U2 and the power in of the IC, U4 and grounding the IC to the ground pin of the LV accumulator connector, J1, as seen in figure 57.

Figure 57: view of jumper wires on the Charger ECU

4.7.2 HV and LV

All HV systems were tested by Malcolm Sidney. All passed testing and the HV 3 phase cable was tagged to be in compliance with AI.4.2.

4.8 Magna-Power supply fault

On 6/12/2023, the final track day at Sydney Motorsport Park (SMSP) was completed. The goal of the day was testing the newly implemented Cascadia Motions CM200 DZ controller. As normal, once the first charge was depleted NU23 was brought to the charging area along with the also newly modified charger. This was not the first time charging with the new modifications, that had been tested the day before in preparation for this track day.

The charging procedure was followed as normal. The only deviation from previous track days being that the ambient temperature was around 30°c. While data is not logged during charging, the following parameters were noted:

- Voltage limit = 445V
- Current limit = 16A
- Accumulator segment temperatures ¡ ambient + 5°c
- SOC ≈ 80%
- Magna-Power power supplies fault state appeared as an overcurrent error

These settings and conditions do not indicate the reason for an overcurrent error to cause a power trip.

While investigating the condition of the charger, the underside of the front-facing display of the power supplies was of a higher surface temperature than others around it, based on touch sensation. Unfortunately there was no way to accurately determine this temperature and it would also not be very helpful as the majority of heat dissipation from the supplies is at the rear end of the charger.

While following the procedure to restart the charging process, the top mounted Magna-Power was not able to be restarted. The rest of the charge cycle was carried out with only a single power supply in use to continue on with testing.

The power supply was inspected by Malcolm Sidney, following the Magna-Power troubleshooting guidelines. The following were the parameters that the supply faulted under:

- Voltage limit $\approx 28 - 30V$
- Current limit = 0.3A

It should also be noted that there is a repeating audible clicking/latching noise coming from the supply, which is believed to be from a relay or relays failing to close/open. This testing indicated an internal fault, resulting in de-mounting the supply to inspect it for damage.

Upon removing the screws from the supply's enclosure and lifting it off, the capacitors inside were tested to confirm that any potential HV present had dissipated using a Fluke 177 digital multi-meter in DC voltage mode. A visual inspection of all wiring and components did not indicate any component failure or loose cabling. As the fault was not found the supply was rebuilt and mounted back into the charger due to time constraints. Re-mounting the supply was completed in order to stop any debris from compromising the HV connections inside.

A further modification for cooling was added to the rear of the charger, as seen in figure 58. This was to minimise the decrease in airflow caused by the new mounting plate for the RCBO box and fixed 3 phase cable. The 2 Delta Electronics PFR0812XHE-SP00 are 12V fans with a maximum current draw

of 4.9A; The datasheet can be found in appendix 7. They are powered from the LV Meanwell AC-DC converter, increasing the estimated current draw up to 22A. This is still well within the 26.7A limit, allowing for spikes in current draw on fan startup.

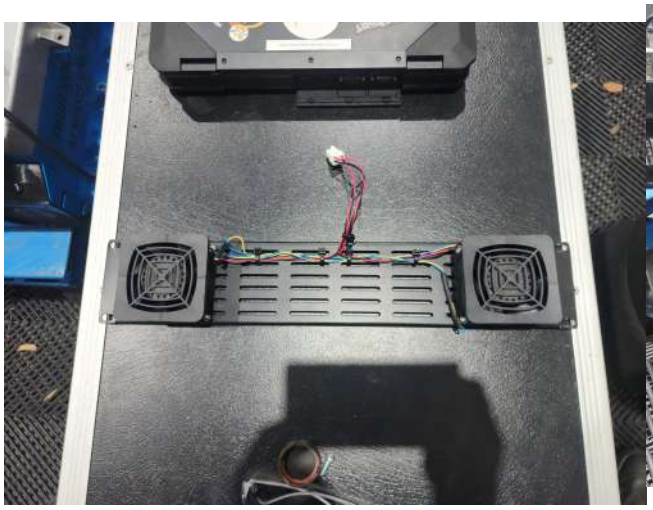


Figure 58: Fans attached to rear mounting plate

Figure 59: View of fans mounted on the charger rear

This issue was postponed until after the competition was finished. The planned course of action was to contact Magna-Power and ascertain if they had any idea what the issue could be and possible solutions; whether that could be accomplished by Malcolm Sidney or would need to be shipped to them was not determined.

4.9 Charger Tech Inspection

The charger tech inspection was completed almost without incident. The only issue being the datasheet that was used for the HV cabling was not the correct one. Upon visual inspection, the scrutineer was okay with the markings on the cabling indicating its insulative and heat ratings.

The comments made by the scrutineer were the following:

- Strain relief on all flying leads would be optimal. this is because the charger is moved around on wheels, causing impacts to all connections inside. strain relieving these connections would minimise the chance of failure.
- Greater access to the rear of the charger where the HV assembly is or better visual aids to portray that the connections are compliant would streamline the scrutineering process.

4.10 Charging Procedure

The charge procedure document has been updated to accommodate the changes made to the charger. this can be found in appendix 10 or at:
https://uoneduau.sharepoint.com/:w:/r/sites/StudentGroup-NURacing983/Shared%20Documents/

Electrical/NU-23/Charging%20Procedure%20-%20NU23.docx?d=wdc63050ad08f4ceca96803cb1d70c5bf&csf=1&web=1&e=6ZJkin

4.11 Recommendations

The following are recommendations for modifications to the current charger design:

- Update the Charger ECU schematic and PCB files to fix the IC not being powered correctly
- Either replace or modify the aluminium plate to allow for better airflow out the rear of the charger.
- The fans used to increase airflow are exceptionally loud, to the point that hearing protection should be worn when it is in use. These could be replaced with much lower current quieter fans.

5 Other Works

5.1 Introduction

This section covers the spectrum of work completed by the author that was not significant enough to warrant a section but enough to worth mentioning.

5.2 LVD Faults

5.2.1 Background

The LVD is an electronics node designed by J.Reeves. For more details than are mentioned here, please refer to Reeves' thesis entitled *Lvd/Accumulator Electronics*[6].

responsible for Low Voltage Distribution inside of the accumulator as well as the exterior mounted fans used to cool the accumulator. This involves power as well as signal distribution including:

- Shutdown circuitry for the IMD, AMS and PDOC hard faults.
- Fan power and speed control.
- Handling cell temperature data manipulation and communication over CAN.
- Board mounted thermistor for top plate temperature logging.

5.2.2 AMS Fault 1

Under various testing conditions in the workshop and out on track days, it was noted that a 'flickering' AMS fault was reoccurring at seemingly random time intervals. The code was completely re-written from the ground up and functionality was incrementally added until the source of the issue was found. These incremental versions of the LVD code are included in appendix 9. When implementing the signal frequency to send the temperature OKHS to the BMS it was noted that sending this signal in the range of 990-1050 ms would cause the AMS hard fault indicator light to flicker on and off. Consulting the Orion2 BMS manual indicated that a timeout function was implemented where if the BMS had not received an updated OKHS signal within 1s of receiving the last, it would cause an AMS fault. This showed there was a lack of understanding of communication requirements between hardware. The temperature OKHS was updated to be sent every 500 ms. This addressed the specific failure mode related to the AMS fault.

5.2.3 AMS Fault 2

After the CEN V3.1 was implemented an AMS hard fault became a reoccurring issue. as this fault originates from the LVD, the first actions taken were to test the software was working as intended. Each of the stages of the AMS signal generation were tested. The signal generated from the BMS is a 5V high signal that is pulled down, halving the amplitude of the signal, on the LVD board before being transmitted to the CEN via the Accumulator-CEN LV Loom.

The CEN also has a pull-up for this same signal that was not populated; if it was an unintended voltage divider would halve the amplitude again. This would leave the signal in a constant fault state. Probing different junctures of the traces connected to this node lead to an op-amp comparator IC

that was either faulty or had a small piece of solder shorting 2 of the pads together enough to cause a voltage divider between the pull down on the LVD and the comparator.

Desoldering, cleaning the pad and replacing the IC with a new component removed the voltage divider. Probing with a DMM confirmed the circuit was working as intended and running through the start-up procedure of NU23 verified this.

5.3 Throttle Fault

The throttle command is created by polling 2 pedal-actuated potentiometers that are then manipulated in the PEN ECU node where:

- The raw signals are polled.
- Predetermined offsets are added. Then each is multiplied by a predetermined scaling factor.
- The variables are bounded between 0-100.
- Both variables are averaged together.
- The 'raw torque' signal is run through a plausibility check against soft faults. Those being:
  - the brake and throttle being actuated at the same time can not allow any throttle request
  - The scaled raw signals must be a plausible number between -5 to 120 and the absolute difference must be greater than 10.
- Finally the generated 'gated' torque request (0-100% units) is checked against a ready to drive state variable, a condition set from a switch once the car is safe to drive, and sent via CAN to the Bamocar motor controller.

The reoccurring issue was the throttle was not linear in comparison to what the code had suggested. The mismatch understanding was actually that the motor controller expected a throttle request based on the setting, I max peak, 0xc4, in the controller's software. In this case, the standard setting was 47% as this corresponded to the $\approx 190A$ peak draw that the batteries could draw without damage. The range where the pedal could manipulate the throttle would then only be the initial 47% of its total range.

The most efficient solution to this was to bound the 0-100% generated by the pedal signal to the value of I max Peak. Testing this setting on a track day resulted in driver feedback that the throttle was smoother and much more manageable to control.

5.4 Track Day Involvement

Due to the nature of NU-Racing, track days are a vital component of testing new features and modifications. The following is a list of tasks involved with track days:

- NU23 was usually not in a functional state between track days from systems being modified and tested in isolation. This meant time had to be allocated to rebuilding the car and verifying its track readiness. If it did not work, which almost always happened, time had to be allocated to fault finding and getting it to a track ready state. This ranged from 30 minutes to several hours depending on the severity of the issue.
- Packing equipment used on track days such as:
  - Electrical testing equipment
  - Spare sub-system nodes and components
  - Pelican cases of driver gear, track gear, electrical and mechanical parts
  - Cones for track layout
  - Toolbox
  - Charger
  - Accumulator trolley
  - Gazebos
- Track setup.
- Marshalling.
- Charging the accumulator between testing.
- unpacking equipment.
- debriefing at the end of the day and receiving future work.

In total 24 track days, 15 SMSP and 9 university car park, were completed with the NU23 chassis from the 30th of July. This amounted to 442km of testing, exceeding our goal of 300km by a wide margin. On average time that went into each track day amounted to:

- University car park $\approx 10 - 12$ hours
- SMSP $\approx 12 - 14$ hours

This totalled $\approx 270 - 318$ hours over the course of the year.

5.5 Cost Report

The Cost report is a section of the cost event that is designed to itemise and give a cost to every material, component and process that goes into building an EV. It was completed by the entirety of the 2023 NU-Racing team, especially Drew Bender for designing the cost report template. With the help of Jacob Searle for all mechanical materials and processes, we completed the majority of electrical components in the cost report. A shortlist of those documents is the following:

- Electrical Part Template
- Loom Part Template
- Emrax 188 Motor
- Bamocar D3-700/400 motor controller
- Majority of LV and HV looms. Examples of which are the following:
  - LV Loom, DEN-PEN
  - LV Loom, CEN-DEN
  - LV Loom, CEN-REN
  - LV Loom, CEN-UEN
  - LV Loom, CEN-Accumulator LV
  - HV Loom, CEN-ACC
  - LV Loom, CEN-DCDC LV
  - LV Loom, CEN-DCDC Sig
  - HV Loom, CEN-DCDC
  - HV Loom, CEN-ACC
  - LV Loom, REN-Motec
  - LV Loom, LV Battery-DCDC
  - LV Loom, DEN-Dash
  - LV Loom, PEN-Pedals
  - LV Loom, LVD-AIL
  - LV Loom, DEN-Dash
- UEN Assembly and associated TSAL and UEN E-stop Parts

In total this amounted to ≈ 50 hours of work. Examples of these spreadsheets have been included in Appendix 11. the cost report in its entirety is located at:

5.6 BMS Segment Looms

The previous version of the looms connecting the accumulator segments to the BMS, shown in figure 60, do not have heat shrink protecting them and were very difficult to manage due to the number of connections. With the 9-segment accumulator having different positioning for the connection to the segment, new looms were constructed. the process involved the following:

- Measure the length required.
- Cut a new segment loom to length.
- Strip, crimp and number each connection as they are read in a specific order. connecting these in the wrong order could damage the internals of the BMS module.
- Cut Raychem heatshrink to length.
- insert each connection into the correct position on the segment's Micro-Molex connector side.
- Verify each connection is correct with a DMM.
- Shrink Raychem with a Makita heat gun.

There was a total of 108 connection over 9 looms constructed. The layout, seen in figure 61, is tidier and easier to route where required.

Figure 60: EV3 Accumulator Top Plate

Figure 61: NU23 Accumulator Top Plate

5.7 Mecha Lab and Workshop Maintenance

During the year, the workspaces in the TA building, especially after weeks with high workloads, became very cluttered with tools and equipment being left out of place, scrap wire and consumables not sorted and thrown away. This resulted in multiple workshop clean ups to keep the building in a satisfactory working condition.

5.8 NU-Marine

During one of NU-Marine's testing days I attended to assist with retrieval of the WAMV vessel at Carrington boat ramp. This involved:

- Transporting my own vessel to the location.
- Unloading my vessel into the water.
- Assisting in unloading WAM23.
- Whenever WAM23 lost communication, it had to be retrieved with the help of another Marine team member while I maneuverer my vessel to safely retrieve and bring it back to dock.
- Assist in loading WAM23.
- Load my vessel onto my trailer.
- transport my vessel home.

5.9 LV Rescue Training

Being able to work on HV systems around the car required Low Voltage Rescue training. LV Rescue training was also a requirement to be an Electrical Safety Officer at the 2023 competition. This training was hosted by St. Johns NSW over a 6 hour period. The course covered:

- Legal issues
- Infection Control
- DRSABCD action plan
- Managing the unconscious breathing casualty
- CPR
- Defibrillation
- Low Voltage Rescue
- Safe performance of live electrical work
- Obligations of the Employer and Employee
- Risks & Hazards
- Electrical Isolation
- Rescue Kit

- Gases
- Burns

the certification has been included as figure 62.

St John

STATEMENT OF ATTAINMENT

A Statement of Attainment is issued by a Registered Training Organisation when an individual has completed one or more accredited units

This is a statement that

Joshua James John Dawson

has attained

HLTAID009	Provide cardiopulmonary resuscitation
UETDRRF004	Perform rescue from a live LV panel

RTO: 88041
Completion Date: 10 May 2023
Issue Date: 11 May 2023
In: ADAMSTOWN, NSW
Certificate No: CERT626329

To verify authenticity of this certificate please go to link or scan QR code

Brendan Maher
Chief Executive Officer
St John Ambulance Australia

NATIONALLY RECOGNISED TRAINING

Figure 62: Low Voltage Rescue training Certificate

5.10 Charging the Accumulator

I was chiefly responsible for ensuring the accumulator was charged and ready for track day testing as well as charging on track days. While other people did assist at times, Jacob Bush was the only other individual involved in a regular capacity for this process.

5.11 Accumulator Work

During the disassembly of the 8-segment accumulator and subsequent reassembly into the 9-segment accumulator, I assisted Michael Dalton and Jacob Searle during this process. This involved:

- Disconnecting all associated wiring looms.
- Removing the top plate assembly.
- Carefully extracting each segment housing and removing the cell stacks to be stored in a safe manner.
- Placing the cell stacks in their new segment housing and inserting them into the new accumulator container.
- Placing the top plate into the new accumulator container.
- Connecting all associated wiring looms.
- Verifying that all electrical components are integrated and working as intended.

5.12 Team Meetings

Every week, a time slot for a team meeting was allocated. This was usually a 1 hour meeting that involved:

- Review of previous track day performance.
- Brief description of where all team members are up to in their assigned work and if assistance is required.
- Assigning work that may take priority over current work due to time constraints or lack of manpower for more important tasks.
- Assigning deadlines.
- Informing members of upcoming events and whats expected from them for attendance.

5.13 Open Day Expo

During the University of Newcastle's Open Day, Nu Racing presented NU23 and NU16 in a side by side exhibition show. The work involved was the same as a normal track day as well as representing the NU Racing as a spokesperson to the general public that came up to ask questions about the team and what was involved in the project.

5.14 NU-Teams Career's Expo

The author participated in the NU-Teams Careers exposition day and helped in cleaning the workshop spaces for the event.

Figure 63: NU-Racing FYPs at the Career's Expo

5.15 Industry Partner Visits

The author participated in site visits to Safegroup and 3ME where NU23 had to be transported to and from their sites.

5.16 Mock Tech Inspection

In preparation for tech inspection at competition, multiple mock tech inspections were run to ensure that all systems involved with the competition were compliant and to increase the confidence of members involved with being answering questions about compliance to a satisfactory standard.

5.17 Replacement PCBs for CEN and LVD

In preparation for the competition, a spare assembled PCB for all the electronics nodes around the car was produced. The author contributed to the assembly process, specifically assisting in populating the spare LVD and CEN PCBs, which were completed by J.Reeves and I.Munday, respectively.

5.18 Design Event- EV Tractive System

In the 2023 competition Design event, the author served as a member of the EV tractive team. Responsibilities included presenting the design of the Inverter/Motor controller and motor implementation, along with providing a rationale for the chosen design decisions within the context of NU-Racing.

5.19 Competition Results

NU Racing achieved 7th place in the 2023 FSAE-EV class competition on top of placing 2nd in the business, skidpan and autocross events. A breakdown of points from the 2023 competition compared to 2022 is located in Appendix 12. This displayed that a simple, effective and robust system is capable of achieving as good or better results than teams that implemented more complex systems at the cost of reliability.

NU-Racing was first passing through tech inspection, an achievement continued on from the 2022 EV3 team.

For the cost event, scored out of 100, the entire team put in a combined effort over a 2 week period. This effort was reflected in our result by more than doubling the score from last year, 51.06 from 23.11. The areas where we lost most of our scoring was in lack of full detail in drawings and datasheets. This would be something to improve on further for the 2024 team.

For the business event, Jacob Searle and Justin Li did an amazing job acquiring 2nd–place for the team. Their efforts, while working only as a 2-man group compared to other top teams that had many more people involved, goes to show their innovative business plan. Their ideas are something to be followed and built upon for next year's team.

NU-Racing's design philosophy for NU23 was weight reduction, simplicity, and reliability. Continuing on from EV3's 4th-place performing platform, optimisation while retaining performance and increasing modularity played a key role in design decisions. Engaging in a mock design event prior to the real one played a key role in solidifying the knowledge base of the team and confidence in being questioned on design decisions.

For the skidpan event, the lack of an aero kit allowed for a greater margin of error for the drivers as they made their way around the circuit. Placing 2nd in the event shows the effort in optimisation of dynamic performance achieved by the team.

For the acceleration event, NU23 being a relatively small single motor driven vehicle impacted our results. Placing 7th in this event is reflective of our design maximising the performance of simple, optimised tractive system.

NU-Racing placing 2nd in autocross is a testament to NU23's dynamic performance as well as the reliability and consistency of the drivers on the day. Due to the efforts of the simulation team, Ethan Small and Ethan Pay creating a useful training tool to allow the drivers to have a practice track before competing in the actual event.

Due to an issue in the accumulator, NU23 faulted after 11/18 laps of the endurance event. The projected points for endurance based on our average lap time would have been ≈ 255, given that all other teams kept the same result, reflects the potential of NU23 capabilities. The lack of track time with the new motor controller implementation and overconfidence in NU23's abilities resulted in a DNF for endurance.

NU-Racing demonstrated an overall commendable performance in the 2023 competition, making an advancement in all static and most dynamic events from the preceding year. Despite an unfortunate outcome in the endurance event, there was substantial progress made in the right direction.

6 Conclusion

Over the duration of this project, I have worked on designing new modifications to old systems and conceptualising new ones. I have learnt a lot during this project, including the ability to work in and prioritize the goals of the team, not just the goals I would like to achieve. It has taught me a great deal about design engineering, hardware implementation/integration and all the seemingly tedious but truly satisfying solutions that come along with it.

Recommendations for future team members:

- Ask questions. Alex, Malcolm, your team leader, chief engineers, and senior members of the team are ready and willing to help.
- Prioritising course work alongside NU-Racing will be a struggle at times. Understand your work capabilities and prioritise accordingly.
- Take all the pictures. It can be very difficult to show where you have taken a design from the pits to where you have it at the end of the year if you can't show what state it was in when you started. Keep notes of what processes were completed and thinking behind design decisions because it can be difficult to write about if you can't remember why.

I hope that NU Racing continues its upward trend in both performance and work environment. I hope to continue design engineering in my future career.

Figure 64: Post competition Team Photo

References

[1] D. Birdsall. Energy Storage and Tractive System Design for an Electric Vehicle. Technical report, University of Newcastle, 2021.

[2] J. Bush. Lead Mechatronic Engineer. Technical report, University of Newcastle, 2023.

[3] P. Gleeson. Formula SAE - Low Voltage Systems Design, Manufacture, and Testing. Technical report, University of Newcastle, 2022.

[4] J. Hollier. Chief Engineer. Technical report, University of Newcastle, 2023.

[5] G. Horsnell. Formula SAE - Tractive System Design, Manufacturing and Testing. Technical report, University of Newcastle, 2022.

[6] J. Reeves. LVD/Accumulator Electronics. Technical report, University of Newcastle, 2023.

[7] X. Young. Formula SAE - HV and LV Systems Implementation. Technical report, University of Newcastle, 2022.

7 Appendix A: Datasheets

Magna Power SL500 datasheet:

https://uoneduau.sharepoint.com/:b:/r/sites/StudentGroup-NURacing983/Shared%20Documents/Electrical/EV3/Magna-Power%20Electronics%20Installe/Documentation/user_manual_sl.pdf?csf=1&web=1&e=ZGbhdA

6-Way PDU with Surge and Overload Protection ($240V - AC$ outlet):

https://www.jaycar.com.au/6-way-pdu-with-surge-and-overload-protection/p/MS4094

Mean Well RSP-320-12 (LV Power Supply) Datasheet:

https://www.meanwell.com/Upload/PDF/RSP-320/RSP-320-SPEC.PDF

Hager ADC416T RCBO datasheet: https://hager.com/au/products/product-information/adm416t-rcbo-4p

Hager ADC910T RCBO datasheet:

https://hager.com/au/products/product-information/adc910t-rcbo-1m-1pn-6ka-c-10a-30ma-a

Delta Electronics PFR0812XHE-SP00 Datasheet: https://www.delta-fan.com/Download/Spec/PFR0812XHE-SP00.pdf

CM200 INVERTER

Our newest inverter is the CM200, packing the punch of a PM150 but being smaller volume and lighter weight than a PM100. Also features HVIL, pluggable connectors and an EMI filter!

FEATURES

- 4 (0-5V) Analog Inputs
- 2 RTD inputs PT100/1000
- 4 Digital Inputs 4-STG
- 2 High Side Driver Outputs
- 2 Low Side Driver Outputs
- 1 Resolver Interface
- 1 Sin-Cos Encoder Interface (-SP Option)
- 2 CAN 2.0A/B Ports 125kb-1Mb adjustable rate and offset
- RS232 Programming and Diagnostic Connection
- Rosenberger Power Connectors
- Integrated DC-Link EMI Filter
- Designed to ISO16750 heavy vehicle climatic, mechanical, and environmental requirements
- ISO20653 high pressure wash rated IP6K9K / IP67
- Easy to use CAN-based network node
- Custom .dbc messaging
- J1939 compatible CAN messages available
- Extensive feedback broadcast messaging for datalogging
- PC-based setup and programming tools available free
- Robust, fault-tolerant IGBT power stage
- HVIL Interlock on connectors
- Command Safety Watchdog
- Variable PWM Rate

CM200	DX	DZ	Units
DC Voltage – operating	50-480	200-840	VDC
DC Overvoltage Trip	500	860	VDC
Maximum DC Voltage – non-operating	500	900	VDC
Motor Current Continuous	300	200	A
Motor Current Peak *	740	400	Arms
Output Power Peak (elect) *	225	225	kW
DC Bus Capacitance	650	255	µF
Size and Volume	330 x 188 x 97 / 3.9		mm / L
Weight	6.75		kg
Active Discharge via motor winding to <50V	< 1		sec
Passive Discharge (internal resistor) to <50V	< 120		sec
Vehicle System Power	9 .. 32 (12V & 24V systems)		VDC
Inverter PWM Frequency	12 (6 .. 16 variable)		kHz
Operating Temperature Range – coolant water	- 40 .. +80, (derate to zero 80 .. 100)		°C
Coolant Flow Rate	12 (3 GPM min)		LPM
Coolant Pressure Drop (60°C coolant / 12 LPM)	0.3 (30kPa / 4.3psi)		bar
Maximum Coolant Pressure (absolute)	3 (300kPa / 45psia)		bar
Operating Shock (ISO 16750-3, Test 4.2.2.2)	500 (50g), pending		m/s^2
Operating Vibration (ISO 16750-3, 4.1.2.7 Test VII)	57.9 (6grms), pending		m/s^2
EMC compatibility	IEC61000 / CISPR-25 pending		
Compatible Conductor Sizes	16, 25, 35, 50		mm^2

Ratings subject to change without notice—consult factory
* *Peak current is defined as a maximum of 30 seconds.*

Point Camera Here

To View Manuals

CASCADIA MOTION
cascadiamotion.com +1-503-344-5085

8 Appendix B: Motor Controller Fault Code

Listing 2: GABCAN modified File

```
// This script is dedicated to the one and only taylor swift #vigalante shit

#include "GABCAN.h"

FlexCAN_T4<CAN2, RX_SIZE_256, TX_SIZE_16> can2;
static CAN_message_t Message_out2, Message_in2;

/* ------------------------------------------------------------------------------*/

void GABCAN_init(void) {

  // Define CAN Message struct parameters
  Message_out2.flags.extended = 0;       // Standard Messages, Not Extended ID
  Message_out2.flags.remote   = 0;       // No Remote Transmission Requests (RTR)
  Message_out2.flags.overrun  = 0;       // Unsure what this does
  Message_out2.flags.reserved = 0;       // Reserved bit. Set to zero (dominant).

  // Initialise CAN BUS
  can2.begin();                         // Also setup can2, even though we don't use
  can2.setBaudRate(1000000);            // Set the Baud Rate to 1 Mb/s

  // Setup CAN Mailboxes
  int maxMB = 60;                        // Number of mailboxes to use. T4.0 has 64.
  can2.setMaxMB(maxMB);                  // Set number of mailboxes to use.

  for (int i = 0; i < maxMB; i++) {      // Assign mailboxes to TX or RX
    if (i<30) {
      can2.setMB(i,TX,STD);
    } else {
      can2.setMB(i,RX,STD);
    }
  }
}

void GABCAN_read(float **outputVar2) {

  if (can2.read(Message_in2)) {
  // these statements are for when its looking for motor controller messages on the can bus.
  // right controlller READ ID is 181 this is reading all the right controllor meassages
    if (Message_in2.id == 0x181) {
      // 0x49 is motor temperature
      if (Message_in2.buf[0] == 0x49) {

        // shifting it from little endian to one NUM (NUM has no meaningful value)
        float temp_num_R = GABCAN_Unpack(Message_in2.buf[2], Message_in2.buf[1]);
        //Serial.println("Motor Temp NUM"); Serial.println(temp_num_R);
        //Serial.println(temp);
         // converting the NUM to temp degrees celcius to be sent to the output variable
        *outputVar2[0] = ((0.0196*temp_num_R) - 187.75);
      }
```

```
  // 0x30 is actual speed
else if (Message_in2.buf[0] == 0x30) {

  // shifting it form little endian to one NUM (NUM has no meaningful value)
  float Speed_num_R = GABCAN_Unpack(Message_in2.buf[2], Message_in2.buf[1]);
  //Serial.println("Motor Speed Right NUM"); Serial.println(Speed_num_R);
  if (Speed_num_R < 32768) {
    // if spinning forward
    // converting from a NUM to a rpm to be sent over can
    *outputVar2[1] = Speed_num_R *6500/ 32767 ;
  }
  else {
    // if spinning Backwards
    *outputVar2[1] = (65534 - Speed_num_R)*6500 / 32767;
  }
}
// 4a is motor controllers temp
else if (Message_in2.buf[0] == 0x4a) {
  float Cont_temp_num_R = GABCAN_Unpack(Message_in2.buf[2], Message_in2.buf[1]); // shifting it f
  //Serial.println("Cont temp Right NUM"); Serial.println(Cont_temp_num_R);
  *outputVar2[2] = 0.0111*Cont_temp_num_R - 191.41;   // equation to convert the temperature to d
}
// eb is controllor voltage
else if(Message_in2.buf[0] == 0xeb){
  float voltage_num_R = GABCAN_Unpack(Message_in2.buf[2], Message_in2.buf[1]);
  //Serial.println("Cont volt Right NUM"); Serial.println(voltage_num_R);
  *outputVar2[3] = voltage_num_R*800/ 25200;;
}
//20 is controller current
else if(Message_in2.buf[0] == 0x20){
    // shifting it form little endian to one NUM (NUM has no meaningful value)
    float current_num_R = GABCAN_Unpack(Message_in2.buf[2], Message_in2.buf[1]);
    *outputVar2[4] = current_num_R; // converting from NUM to amps (A);
}
// C6 is device current
else if(Message_in2.buf[0] == 0xC6){
    float current_device_R = GABCAN_Unpack(Message_in2.buf[2], Message_in2.buf[1]);
    *outputVar2[5] = current_device_R;
}

else if(Message_in2.buf[0] == 0xD9){
    float current_PC_R = GABCAN_Unpack(Message_in2.buf[2], Message_in2.buf[1]);
    *outputVar2[6] = current_PC_R;
}
//Addition: Reads the fault codes for the current derating that the controller has been
// set to do
// will output the codes contained in the address 0x40- 4 bytes of data
// each bit corresponds to a fault state. only bits 5, 20-28 are relevant
// (10 total bits)
else if(Message_in2.buf[0] == 0x40){
// bits that you care about
int bits[10] = {5,20,21,22,23,24,25, 26, 27, 28};
// a place to store the 1s and 0s
```

```
      int vals[10];
      //       // loop over bits we care about
      for (int i=0; i<10; i++) {
        // printf("bit = %d, ",bits[i]);
        // find out which byte bit belongs to (0 indexed)
        int byte = bits[i] / 8; // integer division
        // printf("byte = %d, ",byte);
        // find out what position in the byte this corresponds to (0 indexed)
        int rel_pos = bits[i] - 8*byte-7;
        // printf("rel pos = %d, ",rel_pos);
        // get the bit value by bit shifting and masking
        vals[i] = (Message_in2.buf[byte] >> rel_pos) & 1;
        //  printf("bit val = %d, \n",vals[i]);
      }
      // Bit 5 of 0x40 - Icns
      *outputVar2[7] = vals[0];
      // Bit 20 of 0x40 - Ird-dig
      *outputVar2[8] = vals[1];
      // // Bit 21 of 0x40 -  Iuse_rchd
      *outputVar2[9] = vals[2];
      // Bit 22 of 0x40 -  Ird-N
      *outputVar2[10] = vals[3];
      // Bit 23 of 0x40 -  IrdN
      *outputVar2[11] = vals[4];
      // Bit 24 of 0x40 - IrdTiR
      *outputVar2[12] = vals[5];
      // Bit 25 of 0x40 - Great10Hz
      *outputVar2[13] = vals[6];
      // Bit 26 of 0x40 -  IrdTM
      *outputVar2[14] = vals[7];
      // Bit 27 of 0x40 -  IrdAna
      *outputVar2[15] = vals[8];
      // Bit 28 of 0x40 -  Iwcns
      *outputVar2[16] = vals[9];
      }
    }
  }
}

void GABCAN_SendTorque(float appsOutGated, float rtdState) {
  float torqueSig = appsOutGated*rtdState;
  //apply scale and offset
  float scaledTorque = (((torqueSig )/ 100.0) * 32767.0);
  //scaledTorque = scaledTorque; + 32768.0;
  float Controller_ID[] = {0x201, 0x202};
    Message_out2.len = 3;
    //pack message buffer
    Message_out2.buf[0] = 0x90; //reg
    Message_out2.buf[1] = (int)scaledTorque & 0xFF; //lsb (big or little endian?)
    Message_out2.buf[2] = (int)scaledTorque >> 8; //msb
  for (int i = 0; i < 2; i++) {
      //define message parameters for Flexcan message structure
      Message_out2.id = Controller_ID[i] ;
```

```
      //send message on bus 2
      can2.write(Message_out2);
    }
}

void GABCAN_RequestData() {
  // IDS of messages to send 0xeb,0x20
  float Register_ID[] = {0x30,0x49,0x4a,0xeb,0x20, 0x40, 0xC6, 0xD9};
  Message_out2.buf[2] = 100; // how often is this requested in ms
  Message_out2.len = 3; // length of the message
  Message_out2.buf[0] = 0x3d; // reg ID Telling Motor Controller what you want to do
    for (int j = 0; j < 6; j++) {
      Message_out2.id = 0x201; // address of motor controller
      Message_out2.buf[1] = Register_ID[j]; // Telling the motor controller to read
      can2.write(Message_out2);
      //delay(150);
    }
}

float GABCAN_Unpack(int highByte, int lowByte) {
  float NUM = (highByte << 8) | lowByte;
  return NUM;
}
```

9 Appendix C: LVD Code Blocks

Listing 3: LVDCodeV4_0

```
// Include necessary libraries and define pins
// Code by Josh and Co
// Purpose: Minimal wokring code for LVD. N_milliseconds must be set below 1002 at most
// ( ideally x<600) or AMS flicker due to BMS update frequency at 2.2 seconds

#include <EV3_CAN.h> // Include the NU23_CAN library for CAN communication

// Pin Definitions
#define IN_PWM_PIN 11         // Pin for PWM input to control a fan
#define OUT_PWM_PIN 12        // Pin for PWM output to control a fan

// Cooling Control Parameters
float lvPowerStatus = 0;  // Temperature value representing power status
int fanPWM = 0;   // PWM speed
// Temperature Fault Parameters
float tempOKHS = 1;       // OKHS (Overheat Shutdown) status, HIGH when no temperature fault

// NUCAN Variables
// CAN messages for input and output
canmsg *inputmsgs[] = {&LV_POWER_STATUS};
float *outputVar[] = {&lvPowerStatus};
int numReceive = 1;  // Number of CAN messages to receive
int numSend = 2;     // Number of CAN messages to send

/* ------------------------------------------------------------------------------*/
// Setup Function

void setup() {
   Serial.begin(9600);  // Initialize serial communication at 9600 baud
   NUCAN_init(numSend, numReceive);  // Initialize the NUCAN communication

   // Set pin modes
   pinMode(IN_PWM_PIN, OUTPUT);  // Set the IN_PWM_PIN as an output for PWM
   pinMode(OUT_PWM_PIN, OUTPUT); // Set the OUT_PWM_PIN as an output for PWM
}

/* ------------------------------------------------------------------------------*/
// Main Loop

void loop() {
  // Read CAN messages and distribute to variables...
  NUCAN_read(outputVar, inputmsgs, numReceive); //Read desired CAN messages,
                                                // store results in outputVar
  // Rebroadcast
  NUCAN_heartbeat(&HB_LVD);

  EVERY_N_MILLISECONDS(999) {
    updateFanSpeed();
    NUCAN_write(&TEMP_OKHS, tempOKHS);
    serialOut();
```

```
  }
}


/* ------------------------------------------------------------------------------*/
void updateFanSpeed(void) {
  if (lvPowerStatus==0) {
      // Turn off the fans if TS is off
      fanPWM = 0;
  }else{
    // Set the fan to full beans
    fanPWM = 255;
  }
    analogWrite(IN_PWM_PIN, fanPWM);
    analogWrite(OUT_PWM_PIN, fanPWM);

}


void serialOut(void) {
  Serial.print("Temp OKHS = "); Serial.println(tempOKHS);
  Serial.print("LV Power Status = "); Serial.println(lvPowerStatus);
  Serial.print("Fan PWM = "); Serial.println(fanPWM);
}
```

Listing 4: LVDCodeV4_1

```
// Include necessary libraries and define pins
// Code by Josh and Co
// Purpose: Minimal wokring code for LVD. N_milliseconds must be set below 1002 at most
// ( ideally x<600) or AMS flicker due to BMS update frequency at 2.2 seconds

#include <EV3_CAN.h> // Include the NU23_CAN library for CAN communication

// Pin Definitions
#define IN_PWM_PIN 11         // Pin for PWM input to control a fan
#define OUT_PWM_PIN 12        // Pin for PWM output to control a fan

// Cooling Control Parameters
float lvPowerStatus = 0;  // Temperature value representing power status
int fanPWM = 0;   // PWM speed
// Temperature Fault Parameters
float tempOKHS = 1;       // OKHS (Overheat Shutdown) status, HIGH when no temperature fault

// NUCAN Variables
// CAN messages for input and output
canmsg *inputmsgs[] = {&LV_POWER_STATUS};
float *outputVar[] = {&lvPowerStatus};
int numReceive = 1;  // Number of CAN messages to receive
int numSend = 2;     // Number of CAN messages to send

/* ------------------------------------------------------------------------------------*/
// Setup Function

void setup() {
   Serial.begin(9600);  // Initialize serial communication at 9600 baud
   NUCAN_init(numSend, numReceive);  // Initialize the NUCAN communication

   // Set pin modes
   pinMode(IN_PWM_PIN, OUTPUT);  // Set the IN_PWM_PIN as an output for PWM
   pinMode(OUT_PWM_PIN, OUTPUT); // Set the OUT_PWM_PIN as an output for PWM
}

/* ------------------------------------------------------------------------------------*/
// Main Loop

void loop() {
  // Read CAN messages and distribute to variables...
  NUCAN_read(outputVar, inputmsgs, numReceive); //Read desired CAN messages,
                                                // store results in outputVar

  // Rebroadcast
  NUCAN_heartbeat(&HB_LVD);

  EVERY_N_MILLISECONDS(1000) {
    updateFanSpeed();
    serialOut();
  }
```

```
   EVERY_N_MILLISECONDS(500) {    //set to 500 such that the BMS timeout shouldnt occur
                                  // after 2.2 seconds
    NUCAN_write(&TEMP_OKHS, tempOKHS);
   }
}


/* ------------------------------------------------------------------------------*/
void updateFanSpeed(void) {
  if (lvPowerStatus==1) {
    // Set the fan to full beans
      fanPWM = 255;
  }else{
    // Turn off the fans if TS is off
    fanPWM = 0;
  }
    analogWrite(IN_PWM_PIN, fanPWM);
    analogWrite(OUT_PWM_PIN, fanPWM);

}


void serialOut(void) {
  Serial.print("Temp OKHS = "); Serial.println(tempOKHS);
  Serial.print("LV Power Status = "); Serial.println(lvPowerStatus);
  Serial.print("Fan PWM = "); Serial.println(fanPWM);
}
```

Listing 5: LVDCodeV4_2

```
// Include necessary libraries and define pins
// Code by Josh and Co
// Purpose: Minimal wokring code for LVD. N_milliseconds must be set below 1002ms at most
// ( ideally x < 600ms) or AMS flicker due to BMS update frequency at 2.2 seconds
//          Added charge case code that will trigger when charging. sets pwm to 60 PWM
#include <EV3_CAN.h> // Include the NU23_CAN library for CAN communication

// Pin Definitions
#define IN_PWM_PIN 11         // Pin for PWM input to control a fan
#define OUT_PWM_PIN 12        // Pin for PWM output to control a fan

// Cooling Control Parameters
float lvPowerStatus = 0;  // Temperature value representing power status
float charge = 0;     // charge fan speed control parameter
int fanPWM = 0;   // PWM speed
// Temperature Fault Parameters
float tempOKHS = 1;       // OKHS (Overheat Shutdown) status, HIGH when no temperature fault

// NUCAN Variables
// CAN messages for input and output
canmsg *inputmsgs[] = {&LV_POWER_STATUS, &CHARGE};
float *outputVar[] = {&lvPowerStatus, &charge};
int numReceive = 2;  // Number of CAN messages to receive
int numSend = 2;     // Number of CAN messages to send

/* ------------------------------------------------------------------------------------*/
// Setup Function

void setup() {
   Serial.begin(9600);  // Initialize serial communication at 9600 baud
   NUCAN_init(numSend, numReceive);  // Initialize the NUCAN communication

   // Set pin modes
   pinMode(IN_PWM_PIN, OUTPUT);  // Set the IN_PWM_PIN as an output for PWM
   pinMode(OUT_PWM_PIN, OUTPUT); // Set the OUT_PWM_PIN as an output for PWM
}

/* ------------------------------------------------------------------------------------*/
// Main Loop

void loop() {
  // Read CAN messages and distribute to variables...
  NUCAN_read(outputVar, inputmsgs, numReceive); //Read desired CAN messages,
                                                // store results in outputVar
  // Rebroadcast
  NUCAN_heartbeat(&HB_LVD);

  EVERY_N_MILLISECONDS(1000) {
    updateFanSpeed();
    serialOut();
  }
```

```
   EVERY_N_MILLISECONDS(500) {    // set to 500 such that the BMS timeout
                                  // shouldnt occur after 2.2 seconds
    NUCAN_write(&TEMP_OKHS, tempOKHS);
   }
}


/* ------------------------------------------------------------------------------*/
void updateFanSpeed(void) {
  if (lvPowerStatus == 1) {
    // Set the fan to full beans
    fanPWM = 255;
  } else if (charge == 1) {
    // Charge speed based on LV supply from charger (12V, 10A)
    fanPWM = 60;
  } else {
    // Turn off the fans if DCDC is off
    fanPWM = 0;
  }
  analogWrite(IN_PWM_PIN, fanPWM);
  analogWrite(OUT_PWM_PIN, fanPWM);
}


void serialOut(void) {
  Serial.print("Temp␣OKHS␣=␣"); Serial.println(tempOKHS);
  Serial.print("LV␣Power␣Status␣=␣"); Serial.println(lvPowerStatus);
  Serial.print("Fan␣PWM␣=␣"); Serial.println(fanPWM);
}
```

Listing 6: LVDCodeV4_3

```
// Include necessary libraries and define pins
// Code by Josh and Co
// Purpose: Minimal wokring code for LVD. N_milliseconds must be set below 1002ms at most
// ( ideally x < 600ms) or AMS flicker due to BMS update frequency at 2.2 seconds
//          Added charge case code that will trigger when charging. sets pwm to 60 PWM
//          Added CAN2 HB rebroadcasting
#include <EV3_CAN.h> // Include the NU23_CAN library for CAN communication

// Pin Definitions
#define IN_PWM_PIN 11         // Pin for PWM input to control a fan
#define OUT_PWM_PIN 12        // Pin for PWM output to control a fan

// CAN 2 setup block
FlexCAN_T4<CAN2, RX_SIZE_256, TX_SIZE_16> can2; // Initialize CAN2 communication
CAN_message_t msg;         // Define an incoming/Outgoing CAN message for pass through

 // Define HB IDs for Canamons to Rebroadcast
int heartBeatIDs[] = {1001, 1002, 1003, 1004, 1005, 1006, 1007, 1008, 1009};


// Cooling Control Parameters
float lvPowerStatus = 0;  // Temperature value representing power status
float charge = 0;     // charge fan speed control parameter
int fanPWM = 0;   // PWM speed
// Temperature Fault Parameters
float tempOKHS = 1;

// NUCAN Variables
// CAN messages for input and output
canmsg *inputmsgs[] = {&LV_POWER_STATUS, &CHARGE};
float *outputVar[] = {&lvPowerStatus, &charge};
int numReceive = 2;  // Number of CAN messages to receive
int numSend = 2+9;     // Number of NUCAN messages

/* ------------------------------------------------------------------------------------*/
// Setup Function

void setup() {
  Serial.begin(9600);  // Initialize serial communication at 9600 baud
  NUCAN_init(numSend, numReceive);  // Initialize the NUCAN communication
  can2.begin();  // Start CAN communication
  can2.setBaudRate(1000000);  // Set the CAN bus baud rate to 1 Mbps

  // Set pin modes
  pinMode(IN_PWM_PIN, OUTPUT);  // Set the IN_PWM_PIN as an output for PWM
  pinMode(OUT_PWM_PIN, OUTPUT); // Set the OUT_PWM_PIN as an output for PWM
}

/* ------------------------------------------------------------------------------------*/
// Main Loop

void loop() {
```

```
  // Read CAN messages and distribute to variables...
  NUCAN_read(outputVar, inputmsgs, numReceive);
  CAN2_read();
  // Rebroadcast
  NUCAN_heartbeat(&HB_LVD);

  EVERY_N_MILLISECONDS(1000) {
    updateFanSpeed();
    serialOut();
  }

   EVERY_N_MILLISECONDS(500) {
    NUCAN_write(&TEMP_OKHS, tempOKHS); //set to 500 such that the BMS timeout
                                       // shouldnt occur after 2.2 seconds
   }
}


/* ---------------------------------------------------------------------*/
void updateFanSpeed(void) {
  if (lvPowerStatus == 1) {
    // Set the fan to full beans
    fanPWM = 255;
  } else if (charge == 1) {
    // Charge speed based on LV supply from charger (12V, 10A)
    fanPWM = 60;
  } else {
    // Turn off the fans if DCDC is off
    fanPWM = 0;
  }
  analogWrite(IN_PWM_PIN, fanPWM);
  analogWrite(OUT_PWM_PIN, fanPWM);
}

void CAN2_read(void){
  if (can2.read(msg)){
    for (int i = 0; i < 9; i++){
      if (msg.id == heartBeatIDs[i]){
      NUCAN_direct_write(msg);
      }
    }
  }
  //more on that later...
}

void serialOut(void) {
  Serial.print("Temp OKHS = "); Serial.println(tempOKHS);
  Serial.print("LV Power Status = "); Serial.println(lvPowerStatus);
  Serial.print("Fan PWM = "); Serial.println(fanPWM);
}
```

Listing 7: LVDCodeV4_4

```
// Include necessary libraries and define pins
// Code by Josh and Co
// Purpose: Minimal wokring code for LVD. N_milliseconds must be set below 1002ms at most
// ( ideally x < 600ms) or AMS flicker due to BMS update frequency at 2.2 seconds
//          Added charge case code that will trigger when charging. sets pwm to 60 PWM
//          Added CAN2 HB rebroadcasting
//          Added Max Temp and index extraction from CAN2 and rebroadcasting to NUCAN
#include <EV3_CAN.h> // Include the NU23_CAN library for CAN communication

// Pin Definitions
#define IN_PWM_PIN 11         // Pin for PWM input to control a fan
#define OUT_PWM_PIN 12        // Pin for PWM output to control a fan

// CAN 2 setup block
FlexCAN_T4<CAN2, RX_SIZE_256, TX_SIZE_16> can2; // Initialize CAN2 communication
CAN_message_t msg;         // Define an incoming/Outgoing CAN message for pass through
// Define HB IDs for Canamons to Rebroadcast
int heartBeatIDs[] = {1001, 1002, 1003, 1004, 1005, 1006, 1007, 1008, 1009};
//IDs for minMaxAvg segment temps
int minMaxAvgIDs[] = {1210, 1220, 1230, 1240, 1250, 1260, 1270, 1280, 1290};
float maxTemps[] = {1, 1, 1, 1, 1, 1, 1, 1, 1}; // initialise segment max temps

// Cooling Control Parameters
float lvPowerStatus = 0;  // Temperature value representing power status
float charge = 0;     // charge fan speed control parameter
int fanPWM = 0;   // PWM speed
// Temperature Fault Parameters
float tempOKHS = 1;

// NUCAN Variables
// CAN messages for input and output
canmsg *inputmsgs[] = {&LV_POWER_STATUS, &CHARGE};
float *outputVar[] = {&lvPowerStatus, &charge};
int numReceive = 2;  // Number of CAN messages to receive
int numSend = 4+9;     // Number of NUCAN messages

/* ------------------------------------------------------------------------------*/
// Setup Function

void setup() {
  Serial.begin(9600);  // Initialize serial communication at 9600 baud
  NUCAN_init(numSend, numReceive);  // Initialize the NUCAN communication
  can2.begin();  // Start CAN communication
  can2.setBaudRate(1000000);  // Set the CAN bus baud rate to 1 Mbps

  // Set pin modes
  pinMode(IN_PWM_PIN, OUTPUT);  // Set the IN_PWM_PIN as an output for PWM
  pinMode(OUT_PWM_PIN, OUTPUT); // Set the OUT_PWM_PIN as an output for PWM
}

/* ------------------------------------------------------------------------------*/
// Main Loop
```

```
void loop() {
  // Read CAN messages and distribute to variables...
  NUCAN_read(outputVar, inputmsgs, numReceive); //Read desired CAN messages,
                                                //store results in outputVar
  CAN2_read();
  // Rebroadcast
  NUCAN_heartbeat(&HB_LVD);

  EVERY_N_MILLISECONDS(1000) {
    updateFanSpeed();
    serialOut();
    // Finds the maximum segment temp and writes to NUCAN
    NUCAN_write(&MAX_SEG_TEMP, findMax(maxTemps));
    // Finds the maximum segment temp INDEX and writes to NUCAN. +1 as its 1 indexed not 0.
    NUCAN_write(&MAX_SEG_TEMP_IND, findMaxInd(maxTemps)+1);
  }

   EVERY_N_MILLISECONDS(500) {    //set to 500 such that the BMS timeout shouldnt occur
                                  // after 2.2 seconds
    NUCAN_write(&TEMP_OKHS, tempOKHS);
   }
}


/* ---------------------------------------------------------------------------*/
void updateFanSpeed(void) {
  if (lvPowerStatus == 1) {
    // Set the fan to full beans
    fanPWM = 255;
  } else if (charge == 1) {
    // Charge speed based on LV supply from charger (12V, 10A)
    fanPWM = 60;
  } else {
    // Turn off the fans if DCDC is off
    fanPWM = 0;
  }
  analogWrite(IN_PWM_PIN, fanPWM);
  analogWrite(OUT_PWM_PIN, fanPWM);
}

void CAN2_read(void){
  if (can2.read(msg)){
    for (int i = 0; i < 9; i++){
      if (msg.id == heartBeatIDs[i]){
      NUCAN_direct_write(msg);
      } else if (msg.id == minMaxAvgIDs[i]) {
        maxTemps[i] = msg.buf[0]*0.4; // read temp*tempscale
      }
    }
  }
  //more on that later...
}
```

```
float findMax(float values[]) {
  int size = sizeof(values) / sizeof(values[0]); // Calculate the size of the array

  float max = values[0]; // Assume the first element is the maximum

  // Iterate through the array to find the maximum value
  for (int i = 1; i < size; i++) {
    if (values[i] > max) {
      max = values[i]; // Update the maximum value if a larger value is found
    }
  }

  return max; // Return the maximum value
}

float findMaxInd(float values[]) {
  int size = sizeof(values) / sizeof(values[0]); // Calculate the size of the array
  int maxIndex = 0; // Assume the first element is the maximum
  // Iterate through the array to find the index of the maximum value
  for (int i = 1; i < size; i++) {
    if (values[i] > values[maxIndex]) {
      // Update the index of the maximum value if a larger value is found
      maxIndex = i;
    }
  }
  return maxIndex; // Return the index of the maximum value
}

void serialOut(void) {
  Serial.print("Temp OKHS = "); Serial.println(tempOKHS);
  Serial.print("LV Power Status = "); Serial.println(lvPowerStatus);
  Serial.print("Fan PWM = "); Serial.println(fanPWM);
}
```

Listing 8: LVDCodeV4_5

```
// Include necessary libraries and define pins
// Code by Josh and Co
// Purpose: Minimal wokring code for LVD. N_milliseconds must be set below 1002ms at most
// ( ideally x < 600ms) or AMS flicker due to BMS update frequency at 2.2 seconds
//          Added charge case code that will trigger when charging. sets pwm to 60 PWM
//          Added CAN2 HB rebroadcasting
//          Added Max Temp and index extraction from CAN2 and rebroadcasting to NUCAN
//          Added Max seg temps and warning for loss of communication with a segment

#include <EV3_CAN.h>  // Include the NU23_CAN library for CAN communication

// Pin Definitions
#define IN_PWM_PIN 11                  // Pin for PWM input to control a fan
#define OUT_PWM_PIN 12                 // Pin for PWM output to control a fan
#define SEG_TEMP_MILLIS_TIMEOUT 10000
// CAN 2 setup block
FlexCAN_T4<CAN2, RX_SIZE_256, TX_SIZE_16> can2;
CAN_message_t msg;
int heartBeatIDs[] = { 1001, 1002, 1003, 1004, 1005, 1006, 1007, 1008, 1009 };
int minMaxAvgIDs[] = { 1210, 1220, 1230, 1240, 1250, 1260, 1270, 1280, 1290 };
float maxTemps[] = { 1, 1, 1, 1, 1, 1, 1, 1, 1 };
int LastUpdate[] = { 0, 0, 0, 0, 0, 0, 0, 0, 0 };
// Cooling Control Parameters
float lvPowerStatus = 0;  // Temperature value representing power status
float charge = 0;         // charge fan speed control parameter
int fanPWM = 0;           // PWM speed
// Temperature Fault Parameters
float tempOKHS = 1;  // OKHS (Overheat Shutdown) status, HIGH when no temperature fault

// NUCAN Variables
// CAN messages for input and output
canmsg *inputmsgs[] = { &LV_POWER_STATUS, &CHARGE };
float *outputVar[] = { &lvPowerStatus, &charge };
int numReceive = 2;       // Number of CAN messages to receive
int numSend = 5 + 2 * 9;  // Number of NUCAN messages )

// serial variables
float segsPresentOKHS = 1;  // serial out for SEG_PRESENT_OKHS state
float maxIndex = 1; // segment number that has the highest temp
float maxSegTempInd = 1; // number of segment with max temperature
float maxSegTemp = 1; // maximum segment temperature
/* ------------------------------------------------------------------------------*/
// Setup Function

void setup() {
  Serial.begin(9600);               // Initialize serial communication at 9600 baud
  NUCAN_init(numSend, numReceive);  // Initialize the NUCAN communication
  can2.begin();                     // Start CAN communication
  can2.setBaudRate(1000000);        // Set the CAN bus baud rate to 1 Mbps

  // Set pin modes
  pinMode(IN_PWM_PIN, OUTPUT);   // Set the IN_PWM_PIN as an output for PWM
```

```
  pinMode(OUT_PWM_PIN, OUTPUT);  // Set the OUT_PWM_PIN as an output for PWM
}

/* ---------------------------------------------------------------------------*/
// Main Loop

void loop() {
  // Read CAN messages and distribute to variables...
  NUCAN_read(outputVar, inputmsgs, numReceive);
  CAN2_read();
  // Rebroadcast
  NUCAN_heartbeat(&HB_LVD);

  EVERY_N_MILLISECONDS(1000) {
    updateFanSpeed();
    lostSegCheck();
    findMaxInd();
    findMax();
    serialOut();
  }
  EVERY_N_MILLISECONDS(500) {
    NUCAN_write(&TEMP_OKHS, tempOKHS);
  }
}
/* ---------------------------------------------------------------------------*/
void updateFanSpeed(void) {
  if (lvPowerStatus == 1) {
    // Set the fan to full beans
    fanPWM = 255;
  } else if (charge == 1) {
    // Charge speed based on LV supply from charger (12V, 10A)
    fanPWM = 60;
  } else {
    // Turn off the fans if DCDC is off
    fanPWM = 0;
  }
  analogWrite(IN_PWM_PIN, fanPWM);
  analogWrite(OUT_PWM_PIN, fanPWM);
}
/* ---------------------------------------------------------------------------*/
void CAN2_read(void) {
  if (can2.read(msg)) {
    for (int i = 0; i < 9; i++) {
      if (msg.id == heartBeatIDs[i]) {
        NUCAN_direct_write(msg);  //  write the heartbeat of each segment
      } else if (msg.id == minMaxAvgIDs[i]) {
        maxTemps[i] = msg.buf[0] * 0.4;  // read temp*tempscale
        NUCAN_direct_write(msg);         //  write the minMaxAvg temps of each segment
        LastUpdate[i] = millis();
      }
    }
  }
}
```

```
/* ------------------------------------------------------------------------*/
void lostSegCheck(void) {
  segsPresentOKHS = 1;
  for (int i = 0; i < 9; i++) {
    int timeSinceLastUpdate = millis() - LastUpdate[i];


    if (timeSinceLastUpdate > SEG_TEMP_MILLIS_TIMEOUT) {
      segsPresentOKHS = 0;
    }
  }
  NUCAN_write(&SEGS_PRESENT_OKHS, segsPresentOKHS);
}
/* ------------------------------------------------------------------*/
void findMax(void) {
  int size = sizeof(maxTemps) / sizeof(maxTemps[0]);

  maxSegTemp = maxTemps[0];  // Assume the first element is the maximum

  // Iterate through the array to find the maximum value
  for (int i = 1; i < size; i++) {
    if (maxTemps[i] > maxSegTemp) {
      maxSegTemp = maxTemps[i];
    }
  }

  NUCAN_write(&MAX_SEG_TEMP, maxSegTemp);
}
/* ------------------------------------------------------------------*/
void findMaxInd(void) {
  int size = sizeof(maxTemps) / sizeof(maxTemps[0]);
  maxSegTempInd = 0;
  // Iterate through the array to find the index of the maximum value
  for (int i = 1; i < size; i++) {
    if (maxTemps[i] > maxTemps[(int)maxSegTempInd]) {
      // Update the index of the maximum value if a larger value is found
      maxSegTempInd = i;
    }
  }
  maxSegTempInd = maxSegTempInd + 1 ; //0 indexing to 1 indexing
  NUCAN_write(&MAX_SEG_TEMP_IND,maxSegTempInd);
}
/* ------------------------------------------------------------------------*/
void serialOut(void) {
  Serial.print("Temp OKHS = ");
  Serial.println(tempOKHS);
  Serial.print("LV Power Status = ");
  Serial.println(lvPowerStatus);
  Serial.print("Fan PWM = ");
  Serial.println(fanPWM);
  Serial.print("segsPresentOKHS = ");
  Serial.println(segsPresentOKHS);
  Serial.print("maxSegTemp = ");
```

```
  Serial.println(maxSegTemp);
  Serial.print("maxSegTempInd␣=␣");
  Serial.println(maxSegTempInd);
}
```

Listing 9: LVDCodeV4_6

```
// Include necessary libraries and define pins
// Code by Josh Dawson and Co 1/11/2023
// Purpose: Minimal wokring code for LVD. N_milliseconds must be set below 1002ms at most
// ( ideally x < 600ms) or AMS flicker due to BMS update frequency at 2.2 seconds
//          Added charge case code that will trigger when charging. sets pwm to 60 PWM
//          Added CAN2 HB rebroadcasting
//          Added Max Temp and index extraction from CAN2 and rebroadcasting to NUCAN
//          Added Max seg temps and warning for loss of communication with a segment
//          Added Temp over 60 degrees, TEMP_OKHS fault state
#include <EV3_CAN.h>  // Include the NU23_CAN library for CAN communication

// Pin Definitions
#define IN_PWM_PIN 11   // Pin for PWM input to control a fan
#define OUT_PWM_PIN 12  // Pin for PWM output to control a fan

// defines for
#define SEG_TEMP_MILLIS_TIMEOUT 10000
#define FAULT_TEMP 60                   // Fault temperature for segments
#define DELAY_TIME 5000                 // Fault delay time for overTemp
#define BUSPEED 250000                  // Set the Buspeed for CAN
// CAN 2 setup block
FlexCAN_T4<CAN2, RX_SIZE_256, TX_SIZE_16> can2;
CAN_message_t msg;
int heartBeatIDs[] = { 1001, 1002, 1003, 1004, 1005, 1006, 1007, 1008, 1009 };
int minMaxAvgIDs[] = { 1210, 1220, 1230, 1240, 1250, 1260, 1270, 1280, 1290 };
float maxTemps[] = { 1, 1, 1, 1, 1, 1, 1, 1, 1 };
int LastUpdate[] = { 0, 0, 0, 0, 0, 0, 0, 0, 0 };
// Cooling Control Parameters
float lvPowerStatus = 0;  // Temperature value representing power status
float charge = 0;         // charge fan speed control parameter
int fanPWM = 0;           // PWM speed

// Temperature Fault Parameters
float tempOKHS = 1;   // OKHS (Overheat Shutdown) status, HIGH when no temperature fault
int tempSetTime = 0;  // Timer for overtemp beyond 5 seconds over 60 degrees C

// NUCAN Variables
// CAN messages for input and output
canmsg *inputmsgs[] = { &LV_POWER_STATUS, &CHARGE};
float *outputVar[] = { &lvPowerStatus, &charge};
int numReceive = 2;       // Number of CAN messages to receive
int numSend = 5 + 2 * 9;  // Number of NUCAN messages )

// Temperature Fault State Machine parameters
enum STATEVAR {
  STATE_GOOD,  // GOOD state
  STATE_WAIT,  // WAIT state
  STATE_BAD    // BAD state
};
STATEVAR state = STATE_GOOD;  // Initialize the state as GOOD

// serial variables
```

```
float segsPresentOKHS = 1;  // serial out for SEG_PRESENT_OKHS state
float maxIndex = 1;         // segment number that has the highest temp
float maxSegTempInd = 1;    // number of segment with max temperature
float maxSegTemp = 1;       // maximum segment temperature
/* ------------------------------------------------------------------------*/
// Setup Function

void setup() {
  Serial.begin(9600);               // Initialize serial communication at 9600 baud
  NUCAN_initialise(numSend, numReceive, BUSPEED);
  can2.begin();                     // Start CAN communication
  can2.setBaudRate(250000);        // Set the CAN bus baud rate to 1 Mbps

  // Set pin modes
  pinMode(IN_PWM_PIN, OUTPUT);   // Set the IN_PWM_PIN as an output for PWM
  pinMode(OUT_PWM_PIN, OUTPUT);  // Set the OUT_PWM_PIN as an output for PWM
}

/* ------------------------------------------------------------------------*/
// Main Loop

void loop() {
  // Read CAN messages and distribute to variables...
  NUCAN_read(outputVar, inputmsgs, numReceive);
  // Read CAN2 messages and rebroadcast
  CAN2_read();
  // Rebroadcast LVD Heartbeat
  NUCAN_heartbeat(&HB_LVD);

  EVERY_N_MILLISECONDS(1000) {
    updateFanSpeed();  // Update the fan speed based on charge and LV power status
    lostSegCheck();    // Check for loss of segment communication
    findMaxInd();      // Find the index of the maximum segment temperature
    findMax();         // Find the maximum segment temperature
    // serialOut();   // uncomment to output variables to serial monitor for debugging
  }
  // Write tempOKHS to NUCAN every 500ms to prevent BMS timeout(BMS timeout is 1 second)
  EVERY_N_MILLISECONDS(500) {
    NUCAN_write(&TEMP_OKHS, tempOKHS);
  }
  updateTempOKHS();  // Update the temperature fault state machine
}
/* ------------------------------------------------------------------------*/
// This function updates the fan speed based on the charge and LV power status
void updateFanSpeed(void) {
  if (lvPowerStatus == 1) {
    // Set the fan to full beans
    fanPWM = 255;
  } else if (charge == 1) {
    // Charge speed based on LV supply from charger (12V, 10A)
    fanPWM = 60;
  } else {
    // Turn off the fans if DCDC is off
```

```
    fanPWM = 0;
  }
  analogWrite(IN_PWM_PIN, fanPWM);   // Write the fan speed to the fan
  analogWrite(OUT_PWM_PIN, fanPWM);  // Write the fan speed to the fan
}
/* ---------------------------------------------------------------------------*/
//This function reads CAN2 and rebroadcasts the messages
void CAN2_read(void) {
  if (can2.read(msg)) {
    for (int i = 0; i < 9; i++) {
      if (msg.id == heartBeatIDs[i]) {
        NUCAN_direct_write(msg);  //  write the heartbeat of each segment
      } else if (msg.id == minMaxAvgIDs[i]) {
        maxTemps[i] = msg.buf[0] * 0.4;  // read temp*tempscale
        NUCAN_direct_write(msg);         //  write the minMaxAvg temps of each segment
        LastUpdate[i] = millis();
      }
    }
  }
}
/* ---------------------------------------------------------------------*/
// This function checks for loss of segment communication
void lostSegCheck(void) {
  segsPresentOKHS = 1;
  for (int i = 0; i < 9; i++) {
    int timeSinceLastUpdate = millis() - LastUpdate[i];


    if (timeSinceLastUpdate > SEG_TEMP_MILLIS_TIMEOUT) {
      segsPresentOKHS = 0;
    }
  }
  NUCAN_write(&SEGS_PRESENT_OKHS, segsPresentOKHS);
}
/* ---------------------------------------------------------------------------*/
// This function finds the maximum segment temperature
void findMax(void) {
  int size = sizeof(maxTemps) / sizeof(maxTemps[0]);

  maxSegTemp = maxTemps[0];  // Assume the first element is the maximum

  // Iterate through the array to find the maximum value
  for (int i = 1; i < size; i++) {
    if (maxTemps[i] > maxSegTemp) {
      maxSegTemp = maxTemps[i];
    }
  }

  NUCAN_write(&MAX_SEG_TEMP, maxSegTemp);
}
/* ---------------------------------------------------------------------*/
// This function finds the index of the maximum segment temperature
void findMaxInd(void) {
```

```
  int size = sizeof(maxTemps) / sizeof(maxTemps[0]);
  maxSegTempInd = 0;
  // Iterate through the array to find the index of the maximum value
  for (int i = 1; i < size; i++) {
    if (maxTemps[i] > maxTemps[(int)maxSegTempInd]) {
      maxSegTempInd = i;
    }
  }
  maxSegTempInd = maxSegTempInd + 1;
  NUCAN_write(&MAX_SEG_TEMP_IND, maxSegTempInd);
}
/* ---------------------------------------------------------------*/
// This function updates the temperature fault state machine
void updateTempOKHS(void) {
  switch (state) {
    // GOOD state
    case STATE_GOOD:
      tempOKHS = 1;  // Set the temperature fault to LOW
      if (checkTempGood() == 0) {
        state = STATE_WAIT;
        tempSetTime = millis();  // Set the timer for the WAIT state
      }
      break;
    // WAIT state
    case STATE_WAIT:
      if (checkTempGood() == 1) {
        state = STATE_GOOD;                                    // Go back to the GOOD state
      } else if ((millis() - tempSetTime) >= DELAY_TIME) {
        state = STATE_BAD;                                     // Go to the BAD state
      }
      break;
      // BAD state
    case STATE_BAD:
      tempOKHS = 0;  // Set the temperature fault to HIGH
      if (checkTempGood() == 1) {
        state = STATE_GOOD;  // Go back to the GOOD state
      }
      break;
  }
}
/* ---------------------------------------------------------------------------*/
int checkTempGood(void) {
  int tempCheck;
  if (maxSegTemp > FAULT_TEMP) {
    tempCheck = 0;
  } else {
    tempCheck = 1;
  }
  return tempCheck;
}
/* ---------------------------------------------------------------------------*/
// This function outputs the variables to the serial monitor for debugging
void serialOut(void) {
```

```
  Serial.print("Temp␣OKHS␣=␣");
  Serial.println(tempOKHS);
  Serial.print("LV␣Power␣Status␣=␣");
  Serial.println(lvPowerStatus);
  Serial.print("Fan␣PWM␣=␣");
  Serial.println(fanPWM);
  Serial.print("segsPresentOKHS␣=␣");
  Serial.println(segsPresentOKHS);
  Serial.print("maxSegTemp␣=␣");
  Serial.println(maxSegTemp);
  Serial.print("maxSegTempInd␣=␣");
  Serial.println(maxSegTempInd);
  Serial.print("Charge␣=␣");
  Serial.println(charge);
  Serial.print("Test␣PEN␣=␣");
  Serial.println(hbPen);
}
```

Listing 10: LVDCodeV4_7

```
// Include necessary libraries and define pins
// Code by Josh Dawson and Co 1/11/2023
// Purpose: Minimal wokring code for LVD. N_milliseconds must be set below 1002ms at most
// ( ideally x < 600ms) or AMS flicker due to BMS update frequency at 2.2 seconds
//          Added charge case code that will trigger when charging. sets pwm to 60 PWM
//          Added CAN2 HB rebroadcasting
//          Added Max Temp and index extraction from CAN2 and rebroadcasting to NUCAN
//          Added Max seg temps and warning for loss of communication with a segment
//          Added Temp over 60 degrees, TEMP_OKHS fault state
//          Added FindTempData
#include <EV3_CAN.h>  // Include the NU23_CAN library for CAN communication

// Pin Definitions
#define IN_PWM_PIN 11   // Pin for PWM input to control a fan
#define OUT_PWM_PIN 12  // Pin for PWM output to control a fan
#define BUSPEED 250000  // Set the Buspeed for CAN

// defines for
#define SEG_TEMP_MILLIS_TIMEOUT 10000
#define FAULT_TEMP 60                  // Fault temperature for segments
#define DELAY_TIME 5000                // Fault delay time for overTemp
#define NSEGS 9                        // number of segments
#define RAMPTIME 5                     // Ramp time for fans full beans IN SECONDS

// CAN 2 setup block
FlexCAN_T4<CAN2, RX_SIZE_256, TX_SIZE_16> can2;
CAN_message_t msg;
int heartBeatIDs[] = { 1001, 1002, 1003, 1004, 1005, 1006, 1007, 1008, 1009 };
int minMaxAvgIDs[] = { 1210, 1220, 1230, 1240, 1250, 1260, 1270, 1280, 1290 };
float averageTemps[] = { 1, 1, 1, 1, 1, 1, 1, 1, 1 };
float maxTemps[] = { 1, 1, 1, 1, 1, 1, 1, 1, 1 };
int LastUpdate[] = { 0, 0, 0, 0, 0, 0, 0, 0, 0 };

// Cooling Control Parameters
float lvPowerStatus = 0;  // Temperature value representing power status
float charge = 0;         // charge fan speed control parameter
int fanPWM = 0;           // PWM speed
int coolingFlag = 0;      // cooling on flag when lvPowerStatus is on

// Temperature Fault Parameters
float tempOKHS = 1;         // OKHS (Overheat Shutdown) status
int tempSetTime = 0;        // Timer for overtemp beyond 5 seconds over 60 degrees C
float avgSegTemp = 0;       // Average Seg Temp
float segsPresentOKHS = 1;  // serial out for SEG_PRESENT_OKHS state
float maxIndex = 0;         // segment number that has the highest temp
float maxSegTempInd = 0;    // number of segment with max temperature
float maxSegTemp = 0;       // maximum segment temperature
// NUCAN Variables
// CAN messages for input and output
canmsg *inputmsgs[] = { &LV_POWER_STATUS, &CHARGE, &AVG_SEG_TEMP };
float *outputVar[] = { &lvPowerStatus, &charge, &avgSegTemp };
int numReceive = 2;       // Number of CAN messages to receive
```

```
int numSend = 6 + 2 * 9;  // Number of NUCAN messages)

// Temperature Fault State Machine parameters
enum STATEVAR {
  STATE_GOOD,  // GOOD state
  STATE_WAIT,  // WAIT state
  STATE_BAD    // BAD state
};
STATEVAR state = STATE_GOOD;  // Initialize the state as GOOD


/* ------------------------------------------------------------------*/
// Setup Function

void setup() {
  Serial.begin(9600);                 // Initialize serial communication at 9600 baud
  NUCAN_initialise(numSend, numReceive, BUSPEED); // Initialize the NUCAN communication
  can2.begin();                                   // Start CAN communication
  can2.setBaudRate(BUSPEED);                    // Set the CAN bus baud rate to 1 Mbps

  // Set pin modes
  pinMode(IN_PWM_PIN, OUTPUT);   // Set the IN_PWM_PIN as an output for PWM
  pinMode(OUT_PWM_PIN, OUTPUT);  // Set the OUT_PWM_PIN as an output for PWM
}

/* ------------------------------------------------------------------*/
// Main Loop

void loop() {
  // Read CAN messages and distribute to variables...
  NUCAN_read(outputVar, inputmsgs, numReceive);
  // Read CAN2 messages and rebroadcast
  CAN2_read();
  // Rebroadcast LVD Heartbeat
  NUCAN_heartbeat(&HB_LVD);

  EVERY_N_MILLISECONDS(1000) {
    updateFanSpeed();  // Update the fan speed based on charge and LV power status
    lostSegCheck();    // Check for loss of segment communication
    // findMaxInd();          // Find the index of the maximum segment temperature
    // findMax();             // Find the maximum segment temperature
    findTempData();  //Find Temp data
    serialOut();     // uncomment to output variables to serial monitor for debugging
  }
  // Write tempOKHS to NUCAN every 500ms to prevent BMS timeout(BMS timeout is 1 second)
  EVERY_N_MILLISECONDS(500) {
    NUCAN_write(&TEMP_OKHS, tempOKHS);
  }
  updateTempOKHS();  // Update the temperature fault state machine
}
/* ------------------------------------------------------------------*/
// This function updates the fan speed based on the charge and LV power status
void updateFanSpeed(void) {
```

```
  if (lvPowerStatus == 1) {
    if (coolingFlag < RAMPTIME) {
      fanPWM = 255 *coolingFlag / RAMPTIME;
      coolingFlag = coolingFlag + 1;
    } else {
      // Set the fan to full beans
      fanPWM = 255;
    }
  } else if (charge == 1) {
    // Charge speed based on LV supply from charger (12V, 10A)
    fanPWM = 60;
  } else {
    // Turn off the fans if DCDC is off
    fanPWM = 0;
    coolingFlag = 0;
  }
  NUCAN_write(&FAN_SPD_ACC, fanPWM/255); // Write fan PWM to NUCAN
  analogWrite(IN_PWM_PIN, fanPWM);   // Write the fan speed to the fan
  analogWrite(OUT_PWM_PIN, fanPWM);  // Write the fan speed to the fan
}
/* ------------------------------------------------------------------------*/
//This function reads CAN2 and rebroadcasts the messages
void CAN2_read(void) {
  if (can2.read(msg)) {
    for (int i = 0; i < 9; i++) {
      if (msg.id == heartBeatIDs[i]) {
        NUCAN_direct_write(msg);  //  write the heartbeat of each segment
      } else if (msg.id == minMaxAvgIDs[i]) {
        maxTemps[i] = msg.buf[0] * 0.4;      // read maxtemp*tempscale
        averageTemps[i] = msg.buf[2] * 0.4;  // read averagetemp * tempscale
        NUCAN_direct_write(msg);
        LastUpdate[i] = millis();
      }
    }
  }
}
/* ------------------------------------------------------------------------*/
// This function checks for loss of segment communication
void lostSegCheck(void) {
  segsPresentOKHS = 1;
  for (int i = 0; i < 9; i++) {
    int timeSinceLastUpdate = millis() - LastUpdate[i];

    if (timeSinceLastUpdate > SEG_TEMP_MILLIS_TIMEOUT) {
      segsPresentOKHS = 0;
    }
  }
  NUCAN_write(&SEGS_PRESENT_OKHS, segsPresentOKHS);
}
/* ------------------------------------------------------------------------*/
// This function finds the maximum segment temperature
void findTempData(void) {
  maxSegTemp = maxTemps[0];      // Assume the first element is the maximum
```

```
  avgSegTemp = averageTemps[0];  // Assume the first element is the average
  // Iterate through the array to find the maximum value
  for (int i = 1; i < NSEGS; i++) {
    if (maxTemps[i] > maxSegTemp) {
      // Update the maximum value if a larger value is found
      maxSegTemp = maxTemps[i];
      maxSegTempInd = i;
    }
    avgSegTemp = avgSegTemp + averageTemps[i];
  }
  //  divide by 9 for all the segments
  avgSegTemp = avgSegTemp / NSEGS;
  // Finds the average segment temp and writes to NUCAN
  NUCAN_write(&AVG_SEG_TEMP, avgSegTemp);
  // Finds the maximum segment temp and writes to NUCAN
  NUCAN_write(&MAX_SEG_TEMP, maxSegTemp);
   // Finds the segment with the maximum temp and writes to NUCAN
  NUCAN_write(&MAX_SEG_TEMP_IND, maxSegTempInd);
}
/* ---------------------------------------------------------------------------*/
// This function updates the temperature fault state machine
void updateTempOKHS(void) {
  switch (state) {
    // GOOD state
    case STATE_GOOD:
      tempOKHS = 1;  // Set the temperature fault to LOW
      if (checkTempGood() == 0) {
        state = STATE_WAIT;
        tempSetTime = millis();  // Set the timer for the WAIT state
      }
      break;
    // WAIT state
    case STATE_WAIT:
      if (checkTempGood() == 1) {
        state = STATE_GOOD;       // Go back to the GOOD state
      } else if ((millis() - tempSetTime) >= DELAY_TIME) {  //Check if the timer has ended
        state = STATE_BAD;        // Go to the BAD state
      }
      break;
      // BAD state
    case STATE_BAD:
      tempOKHS = 0;  // Set the temperature fault to HIGH
      if (checkTempGood() == 1) {
        state = STATE_GOOD;  // Go back to the GOOD state
      }
      break;
  }
}
/* ---------------------------------------------------------------------------*/
int checkTempGood(void) {
  int tempCheck;
  if (maxSegTemp > FAULT_TEMP) {
    tempCheck = 0;
```

```
  } else {
    tempCheck = 1;
  }
  return tempCheck;
}
/* ------------------------------------------------------------------------*/
// This function outputs the variables to the serial monitor for debugging
void serialOut(void) {
  Serial.print("Temp␣OKHS␣=␣");
  Serial.println(tempOKHS);
  Serial.print("LV␣Power␣Status␣=␣");
  Serial.println(lvPowerStatus);
  Serial.print("Fan␣PWM␣=␣");
  Serial.println(fanPWM);
  Serial.print("segsPresentOKHS␣=␣");
  Serial.println(segsPresentOKHS);
  Serial.print("maxSegTemp␣=␣");
  Serial.println(maxSegTemp);
  Serial.print("maxSegTempInd␣=␣");
  Serial.println(maxSegTempInd);
  Serial.print("Charge␣=␣");
  Serial.println(charge);
  Serial.print("Average␣Temp␣=␣");
  Serial.println(avgSegTemp);
}
```

10 Appendix D: Charge procedure document

POWER UP Procedure

1. Install the Service Handle plug between accumulator segments.

   WARNING: 450 V potential now exists between + and – poles.

2. Fasten accumulator lid using M6 bolts w/ nylock nuts and washers on top and bottom.

3. Take off both ends of the charger.

4. Connect the 6 Pin DT to Dual CAN Kvaser and BMS CAN adapter to the Panel Mount PCB port labelled EXP. Both a kvaser cord and BMS CAN adapter are needed to read the cell voltage, check for any BMS or cell issues and, to check the difference between the highest and lowest cell voltages. This is called the cell delta.

5. Connect LV and HV looms between accumulator and charger front panel. It is not possible to put these in the wrong way.

6. Ensure the 3 Phase power point is turned OFF.

7. Connect the 3 Phase input power extension to a 3-phase power point.

   WARNING: a 415V 32A 3 phase connection is needed for the charger to work. While the connector should not be able to go on a smaller sized connector, the capacity of a new outlet should be checked.

8. Ensure Magna supplies power switch is in off position.

9. Ensure there is no obstruction in the way of fan blades on both sides of the accumulator and the charger's rear.

10. Turn 3-Phase power point to the ON position.

11. Verify that onboard 12 V output AC/DC converter is working by:

    - Seeing and hearing the charger cooling fans spool up, and/or
    - Seeing and hearing the accumulator cooling fans spool up (Assuming a CHARGING command is sent to and received by the LVD ECU resulting in a charging fan speed %, and/or
    - Observing the RED 12V RAW, PROT and 5V LEDs indicating power to the CHARGE ECU and/or
    - Observing the Teensy Heartbeat LED on the CHARGE ECU board.

12. Open Orion BMS 2 software and Connect to BMS and confirm there's no fault codes active.

    WARNING: Bluetooth must be disabled for the software to work correctly

13. Connect to the appropriate COM port, this should be found automatically by the GUI, at a 250 kb/s BUSPEED. Continue through all prompts by indicating yes.

14. Verify the BMS is functioning by:

- Navigate to the Diagnostic Trouble codes tab and observe no fault codes appearing. This will also be seen in red text at the bottom of the interface as Trouble Codes Detected if one is present.
- Navigate to the Live Text Data Tab. Observe the SOC %, cell delta, highest voltage cell, lowest voltage cell and the cell internal resistance.

WARNING: The cell internal resistance being different to 4mΩ could indicate that the cell/cells are not working correctly. Possible reasons are:

- cell damage
- Faulty wiring – loose connector, not crimped correctly, plug is loose on either end.

15. Open Matlab, then CAN explorer app. Use the KVASER LEAF LITE V2 option. Ensure the following settings are used:
    - In Device Channel, the CANBUS speed is set to 250 kb/s. (this can change if the buspeed is changed in the future).
    - In Databases, the correct DBC file is being used. The current file for charging is the NU23 Charging DBC in the NUCAN library on GIT.
    - In Signals, Move the following signals to the signals table:
        - V_PACK
        - MAX_SEGx_TEMP where x corresponds to segment 1-9.
16. Start sniffing. Observe:
    - The Cell temperatures are appropriate for the conditions.
    - Monitor the cell temperatures, SOC%, and pack voltage while charging.

      WARNING: If these are not monitored this could result in CATASTROPHIC failure of the system (probably not). The MAGNA-POWER Supplies are designed to fault and trip before anything drastic happens, (NOTE: designed to and NOT WILL fault).
17. Ensure the E stop is in the released position, not pressed.
18. Observe:
    - The shutdown circuit is closed up to the TSMS. This will be indicated by 4 green lights in the SHUTDOWN section on the CHARGE ECU and the 5th TSMS light being unlit.
    - Hard fault signals are working, as indicated by the blue lights in the HARD FAULTS section of the CHARGE ECU.
19. Power on the MAGNA-POWER supplies using the toggle switch power button.

    IMPORTANT: DO NOT ENABLE THE SUPPLIES (GREEN START BUTTON) YET
20. Check if the lower power supply has the word 'ROTARY' lit up. (The upper supply should always be set to external programming)
21. If yes, continue to step 15
22. If no, follow the changing modes procedure

23. Set the voltage and current limits on the lower MAGNA-POWER supply by:
    - Holding the UP (V/I DIS) button while turning the voltage and then current knobs
    - Set the voltage limit to Pack Voltage + within 4 V

      This prevents the AIRs from welding when they are closed
    - Set the current limit to 0.15 A

      This is good practice. The current limit will be raised after the AIRs are closed.
24. Enable the supplies using the Green START Buttons

    WARNING: High Voltage will now be present at the HV Autosport.
25. Verify that the Accumulator Indicator is in the red region to indicate the presence of High Voltage at the HV Autosport. Also Verify that the MAGNA-POWER cooling fans are working. If not, begin the POWER DOWN procedure.
26. Turn the TSMS to the closed (right) position.

    This will close the shutdown circuit and, via a CAN message, the precharge board will start 'charging', this message tells the precharge board to allow for fast charging.

    This is needed as there is no capacitance from a motor controller/capacitors to precharge here.
27. The AIRs will close in the accumulator and the AIR inside the Charger, which you can hear.

    If not, there is likely a Hard Fault present, or a current or voltage limit issue. Check the voltage is Pack+ 4V and the current is 0.15A-0.3A
28. The pack should begin drawing current from the MAGNA-POWER supplies, which will current limit at 0.15 A.
29. Increase voltage limit to 445V. This ensures that the cells won't be able to be overcharged.

    WARNING: The Pack should only be charged to 95% SOC during regular testing, charging beyond this can damage the cells.

    WARNING: When charging to above 95% SOC the voltage limit can be increased to 100%, i.e. 453V. EXTREME care should be taken so that the cells never reach this voltage as it is possible to overload the cells causing a chemical fire.

    WARNING: DO NOT CHARGE THE CELLS ABOVE 4.2V per CELL. Lithium fires are self-sustaining. This will be expensive and could result in injury or death.
30. Increase current limit according to SOC and temps. With the current cooling it is unlikely for temperature to ever be a limiting factor unless ambient is > 35°c.
31. If SOC $< 90\%$, and:
    - Temp < 37°c, set current limit to max (approx. 8 A on the power supply, this is 16 A between both)
    - 37°c $<$ Temp < 40°c, current limit = 6 A (Total 12 A)
    - 40°c $<$ Temp < 43°c, current limit = 4 A (Total 8 A)

- Temp > 43°c, set current limit to 0 A, TOO HOT.

32. If SOC is between 90% - 95%, and:
    - Temp < 43°c, current limit = 4 A (Total 8 A)
    - Temp > 43°c, current limit = 0 A, TOO HOT.
33. If SOC > 95%, set current limit to 0 A and follow power down procedure.
34. Monitor cell voltages with Orion Software and CANamon cell temperatures over CAN and verify that no anomalies are present:
    - Min Safe Cell Voltage = 2.5 V
    - Max Safe Cell Voltage = 4.2 V
    - Max Safe Cell Temperature = 45°c

POWER DOWN Procedure

1. Set current to zero.
2. Open the shutdown circuit by turning the TSMS to the off (up) position.
3. Observe:
    - AIRS open in both the charger and accumulator.
    - SHUTDOWN section on CHARGER ECU indicate the TSMS is open.
4. Power down the MAGNA-POWER supplies using the Red STOP button on each
5. Verify that potential drops via:
    - On board displays

      NOTE: There should not be any external HV present. This can be tested via the HV test points and/or the Accumulator indicator being in the white region at 0V.
6. Turn off the MAGNA-POWER supplies using the toggle switch on the front of the panel.
7. Power off the charge cart by turning the 3 Phase power point to the OFF position.
8. Observe:
    - The CHARGE ECU no longer indicates any presence of power
    - All fans power down. Silence is bliss.
9. Remove the LV DT connector.
10. Remove the HV AUTOSPORT.

Changing Modes Procedure (Taken from page 24 of user_manual_sl.pdf, 3.2.3. Configuration Commands):

Figure 65: Magna Supply Control Panel

1. Press `MENU` key. (The over voltage trip LED will initially flash)
2. Press `item` key 2 times. (The voltage display will flash `conF` (configure).)
3. Press the `enter` key to select configure commands. (The `REM SEN` (remote sense) LED will initially flash.)
4. Press the `item` key until the relevant LED flashes. `ROTARY` (manual control) or `EXT PGM` (external program)
5. Press the `enter` key.
6. The power supply may need to be turned off and on again for the change to trigger.

Magna Supplies User Manual:

https://uoneduau.sharepoint.com/:b:/r/sites/StudentGroup-NURacing983/Shared%20Documents/Electrical/EV3/Magna-Power%20Electronics%20Installe/Documentation/user_manual_sl.pdf?csf=1&web=1&e=ZGbhdA

11 Appendix E: Cost Report Documents

University	University of Newcastle	**P/N**	20008	**Part Cost**	$ 3,704.57
Competition Code	FSAE-A	**System**	Motor & Drivetrain	**QTY**	1
Year	2023	**Assembly**	Motor	**Extended Cost**	$ 3,704.57
Car #	E03	**Part**	Emrax 188 HV Motor		

Materials

ID	Material	Description	Size 1	Unit 1	Size 2	Unit 2	Area Name	Area (mm^2)	Length (mm)	Density (kg/m^3)	QTY	Unit Cost	Sub Total
	Wire, Power	costed in part: CONTROLLER L	0.6	m		-					0	$ 3.00	$ -
	Wire, Signal	costed in part: CONTROLLER L	0.6	m		-					0	$ 1.00	$ -
	Connector, Single Wire	soldered connections inside m	1	wire		-					8	$ 0.05	$ 0.40
	Motor, Tractive AC	Emrax 188 Motor	37	kW		-					1	$ 100.00	$ 3,700.00
Total													**$ 3,700.40**

Processes

Process Multipliers

ID	Process	Use	Unit	QTY	Multiplier	Mult. Val.	Multiplier 2	Mult. Val. 2	Unit Cost	Sub total
	Attach Wire, Solder wire, not be	solder connections inside mot	unit	8	-	1	-	1	$ 0.52	$ 4.17
Total										**$ 4.17**

Fasteners

ID	Fasteners	Use	Size 1	Unit 1	Size 2	Unit 2	QTY	Unit Cost	Sub Total
Total									**0**

Tooling

ID	Tooling	Use	Unit	QTY	PVF	Frac. Incl	Unit Cost	Sub Total
Total								**0**

University	University of Newcastle	**P/N**	20001	**Part Cost**	$ 6,153.82
Competition Code	FSAE-A	**System**	Motors & Drivetrain	**QTY**	1
Year	2023	**Assembly**	Motor Controller	**Extended Cost**	$ 6,153.82
Car #	E03	**Part**	Bamocar D3 400/700		

Materials

ID	Material	Description	Size 1	Unit 1	Size 2	Unit 2	Area Name	Area (mm^2)	Length (mm)	Density (kg/m^3)	QTY	Unit Cost	Sub Total
	Chassis Control Module, + Motor Controller AC		135	kW		-					1	$ 6,075.00	$ 6,075.00
	Chassis Control Module, Baseline Enclosure		1	unit		-					1	$ 40.00	$ 40.00
	Chassis Control Module, + HV Housing		1	unit		-					1	$ 25.00	$ 25.00
	Hose, Rubber	3/8" Hose to Motor	17	mm	0.4	m					1	$ 3.06	$ 3.06
Total													**$ 6,143.06**

Processes

Process Multipliers

ID	Process	Use	Unit	QTY	Multiplier	Mult. Val.	Multiplier 2	Mult. Val. 2	Unit Cost	Sub total
	Attach Wire, Terminated wire with screw and nut	3 Phase terminals	unit	3	-	1	-	1	$ 0.52	$ 1.56
	Attach Wire, Terminated wire with screw and nut	CEN HV Terminals	unit	2	-	1	-	1	$ 0.52	$ 1.04
	Reaction Tool <= 25.4 mm	tighten nuts	unit	5	-	1	-	1	$ 0.25	$ 1.25
	Hand, Loose <= 25.4 mm	hand loose nuts	unit	5	-	1	-	1	$ 0.50	$ 2.50
	Shrink Tube	Insulation for cabling	cm	5	-	1	-	1	$ 0.15	$ 0.75
	Tube cut	start nuts	cm	5	-	1	-	1	$ 0.15	$ 0.75
	Tube cut	Cut Hose to length	cm	1.7	-	1	-	1	$ 0.15	$ 0.26
	Assemble, 1 kg, Interference	Attach Hose onto Barbs	unit	1	-	1	Repeat 2	2	$ 0.19	$ 0.38
	Assemble, 1 kg, Loose	Slide on Hose Clamps	unit	1	-	1	Repeat 2	2	$ 0.06	$ 0.13
	Screwdriver > 1 Turn	Tighten Hose Clamps	unit	1	-	1	Repeat 2	2	$ 0.50	$ 1.00
Total										**$ 9.61**

Fasteners

ID	Fasteners	Use	Size 1	Unit 1	Size 2	Unit 2	QTY	Unit Cost	Sub Total
	Hose Clamp, Worm Drive	Fasten Hose	19	mm		-	2	$ 0.58	$ 1.15
Total									**$ 1.15**

Tooling

ID	Tooling	Use	Unit	QTY	PVF	Frac. Incl	Unit Cost	Sub Total
Total								**0**

University	University of Newcastle	**A/N**	A4002	**Part Cost**	N/A
Competition Code	FSAE-A	**System**	Electrical	**QTY**	N/A
Year	2023	**Assembly**	UEN Assembly	**Extended Cost**	$ 42.08
Car #	E03	**Part**	N/A		

Parts

ID	P/N	Description	Part Cost	QTY	Sub Total
	TSAL		$ 15.14	1	$ 15.14
	UEN E-Stop		$ 12.65	2	$ 25.30
Total					**$ 40.44**

Processes

Process Multipliers

ID	Process	Use	Unit	QTY	Multiplier	Mult. Val.	Multiplier 2	Mult. Val. 2	Unit Cost	Sub total
	Install Tie Wrap (Zip Tie, Cable Clamp)	Zip tie TSAL to chassis	unit	1	-	1	Repeat 2	2	$ 0.09	$ 0.19
	Assemble, 1 kg, Loose	Place E-stop through chassis tab	unit	1	-	1	Repeat 2	2	$ 0.06	$ 0.13
	Hand, Tight <= 6.35 mm	Hand tighten E-Stop together	unit	1	-	1	Repeat 2	2	$ 0.50	$ 1.00
	Assemble, 1 kg, Line-on-Line	Clip back of E-Stop on	unit	1	-	1	Repeat 2	2	$ 0.13	$ 0.25
Total										**$ 1.56**

Fasteners

ID	Fasteners	Use	Size 1	Unit 1	Size 2	Unit 2	QTY	Unit Cost	Sub Total
	Tie Wrap	Zip Tie TSAL to chassis	1	unit		-	2	$ 0.04	$ 0.08
Total									**$ 0.08**

University	University of Newcastle	**P/N**		**Part Cost**	$ 15.14
Competition Code	FSAE-A	**System**	Electrical	**QTY**	1
Year	2023	**Assembly**	UEN	**Extended Cost**	$ 15.14
Car #	E03	**Part**	TSAL		

Materials

ID	Material	Description	Size 1	Unit 1	Size 2	Unit 2	Area Name	Area (mm^2)	Length (mm)	Density (kg/m^3)	QTY	Unit Cost	Sub Total
	Plastic, Polyethelene		0.086	kg		-					1	$ 3.30	$ 0.28
	Connector, OEM Quality		4	pin(s)		-					1	$ 0.50	$ 2.00
	Lamp, LED	LED Strip	1	unit		-					8	$ 0.50	$ 4.00
	Wire, Power	Power wires for TSAL	0.3	m		-					1	$ 3.00	$ 0.90
Total													**$ 7.18**

Processes

Process Multipliers

ID	Process	Use	Unit	QTY	Multiplier	Mult. Val.	Multiplier 2	Mult. Val. 2	Unit Cost	Sub total
	Rapid Prototype - Plastic	TSAL Top	kg	0.047	-	1	-	1	$ 32.00	$ 1.50
	Rapid Prototype - Plastic	TSAL Clear	kg	0.008	-	1	-	1	$ 32.00	$ 0.26
	Rapid Prototype - Plastic	TSAL Bottom	kg	0.031	-	1	-	1	$ 32.00	$ 0.99
	Cut wire	Cut power wire to length	unit	2	-	1	-	1	$ 0.08	$ 0.17
	Strip Wire	Strip each end of wire	unit	2	-	1	Repeat 3	3	$ 0.08	$ 0.50
	Crimp Wire	Crimp one end of each wire	unit	3	-	1	-	1	$ 0.17	$ 0.50
	Lay Wire - Signal	Lay LED Strip	m	0.19		2		1	$ 0.02	$ 0.01
	Attach Wire, Solder wire, bent	Solder wires on LED Strip	unit	3		1		1	$ 0.35	$ 1.06
	Connector Assembley, Crimp	Attach Wire to connector	contacts	3		1		1	$ 0.36	$ 1.08
	Assemble, 1 kg, Loose	Put bolts through enclosure and DT connector	unit	2		1		1	$ 0.06	$ 0.13
	Hand - Start Only	Hand start nuts to secure connector to enclosure	unit	1	-	1	Repeat 2	2	$ 0.12	$ 0.24
	Reaction Tool <= 6.35 mm	Tighten nuts to secure connector to enclosure	unit	1	-	1	Repeat 2	2	$ 0.25	$ 0.50
	Liquid Apply - Spot	Attach bottom middle and top together	unit	8		1		1	$ 0.10	$ 0.80
	Assemble, 1 kg, Loose	Assemble all	unit	2		1		1	$ 0.06	$ 0.13
Total										**$ 7.86**

Fasteners

ID	Fasteners	Use	Size 1	Unit 1	Size 2	Unit 2	QTY	Unit Cost	Sub Total
	Bolt, Grade 12.9		3	mm	13	mm	2	$ 0.02	$ 0.04
	Nut, Grade 12.9		3	mm		mm	2	$ 0.03	$ 0.06
Total									**$ 0.10**

Tooling

ID	Tooling	Use	Unit	QTY	PVF	Frac. Incl	Unit Cost	Sub Total
Total								**$ -**

University	University of Newcastle	**P/N**		**Part Cost**	$ 12.65
Competition Code	FSAE-A	**System**	Electrical	**QTY**	2
Year	2023	**Assembly**	UEN	**Extended Cost**	$ 25.31
Car #	E03	**Part**	E-Stop		

Materials

ID	Material	Description	Size 1	Unit 1	Size 2	Unit 2	Area Name	Area (mm^2)	Length (mm)	Density (kg/m^3)	QTY	Unit Cost	Sub Total
	Switch, Kill		1	unit		-					1	$ 3.00	$ 3.00
	Plastic, Polyethelene		0.053	kg		-	Filament Roll				1	$ 3.30	$ 0.17
	Connector, OEM Quality	DT connector	4	pin(s)		-					1	$ 0.50	$ 2.00
	Connector, OEM Quality		2	pin(s)		-					1	$ 0.50	$ 1.00
	Wire, Signal		0.1	m		-					1	$ 3.00	$ 0.30
Total													**$ 6.47**

Processes

Process Multipliers

ID	Process	Use	Unit	QTY	Multiplier	Mult. Val.	Multiplier 2	Mult. Val. 2	Unit Cost	Sub total
	Rapid Prototype - Plastic	3D print housing	kg	0.053	-	1	-	1	$ 32.00	$ 1.70
	Cut wire	Cut wires to length	unit	1	-	1	-	1	$ 0.08	$ 0.08
	Strip Wire	Strip each end of wires	unit	2	-	1	Repeat 2	2	$ 0.08	$ 0.33
	Crimp Wire	Crimp each end of wire	unit	2	-	1	Repeat 2	2	$ 0.17	$ 0.67
	Attach Wire, Wire to terminal block	Attach wire to screw terminal	unit	1	-	1	Repeat 2	2	$ 0.35	$ 0.71
	Connector Assembley, Crimp	Insert crimped into DT	contacts	2	-	1	-	1	$ 0.36	$ 0.72
	Assemble, 1 kg, Loose	Place bolts through enclosure and DT connector	unit	1	-	1	Repeat 4	4	$ 0.06	$ 0.25
	Hand - Start Only	Hand start nuts to secure DT to enclosure	unit	1	-	1	Repeat 4	4	$ 0.12	$ 0.48
	Reaction Tool <= 6.35 mm	Tighten nuts to secure DT to enclosure	unit	1	-	1	Repeat 4	4	$ 0.25	$ 1.00
Total										**$ 5.94**

Fasteners

ID	Fasteners	Use	Size 1	Unit 1	Size 2	Unit 2	QTY	Unit Cost	Sub Total
	Bolt, Grade 12.9	Securing Connector	4	mm	12	mm	4	$ 0.03	$ 0.12
	Nut, Grade 12.9		4	mm		mm	4	$ 0.03	$ 0.12
Total									**$ 0.24**

Tooling

ID	Tooling	Use	Unit	QTY	PVF	Frac. Incl	Unit Cost	Sub Total
Total								**0**

University	University of Newcastle	P/N		Part Cost	#N/A
Competition Code	FSAE-A	System		QTY	
Year	2023	Assembly		Extended Cost	#N/A
Car #	E03	Part			

Materials

ID	Material	Description	Size 1	Unit 1	Size 2	Unit 2	Area Name	Area (mm^2)	Length (mm)	Density (kg/m^3)	QTY	Unit Cost	Sub Total
	Wire, Power		1	m		-					1	$ 3.00	$ 3.00
	Wire, Signal		1	m		-					1	$ 1.00	$ 1.00
	Heat Shrink Tubing		1	m		-					1	$ 0.50	$ 0.50
	Connector, Single Wire		1	wire		-					1	$ 0.05	$ 0.05
	Connector, OEM Quality		1	pin(s)		-					1	$ 0.50	$ 0.50
	Connector, Computer Type		1	pin(s)		-					1	$ 1.00	$ 1.00
	Connector, HC-HV incl. Interlock		1	pin		-					1	$ 30.00	$ 30.00
	Connector, HC-HV Lug Type		1	pin		-					1	$ 1.00	$ 1.00
	Connector, High Power, > 2 Amps		1	pin(s)		-					1	$ 2.00	$ 2.00
	Heat Shrink Tubing			m		-						$ 0.50	$ -
	Connector, HC-HV incl. Interlock			pin		-						$ 30.00	$ -
	Wiring HV (> 12 mm²)			m		-						$ 15.00	$ -
	Connector, HC-HV Lug Type			pin		-						$ 1.00	$ -
				#N/A		#N/A						#N/A	#N/A
				#N/A		#N/A						#N/A	#N/A
Total													**#N/A**

Processes

Process Multipliers

ID	Process	Use	Unit	QTY	Multiplier	Mult. Val.	Multiplier 2	Mult. Val. 2	Unit Cost	Sub total
	Attach Wire, Quick connect terminal		unit	1	-	1	-	1	$ 0.10	$ 0.10
	Attach Wire, Terminated wire with screw		unit	1	-	1	-	1	$ 0.35	$ 0.35
	Shrink Tube		cm		-	1	-	1	$ 0.15	$ -
	Strip Wire		unit		#N/A		#N/A	#N/A	$ 0.08	#N/A
	Cut wire		unit		#N/A		#N/A	#N/A	$ 0.08	#N/A
	Crimp Wire		unit		#N/A		#N/A	#N/A	$ 0.17	#N/A
	Connector Assembley, Crimp		contacts		#N/A		#N/A	#N/A	$ 0.36	#N/A
	Tin Wire		unit		#N/A		#N/A	#N/A	$ 0.13	#N/A
	Install Tie Wrap (Zip Tie, Cable Clamp)		unit		#N/A		#N/A	#N/A	$ 0.09	#N/A
	Insert Bundle Into Tube or Sleeve		m		#N/A		#N/A	#N/A	$ 0.02	#N/A
Total										**#N/A**

Fasteners

ID	Fasteners	Use	Size 1	Unit 1	Size 2	Unit 2	QTY	Unit Cost	Sub Total
	Tie Wrap			unit		-		$ 0.04	$ -
Total									**$ -**

Tooling

ID	Tooling	Use	Unit	QTY	PVF	Frac. Incl	Unit Cost	Sub Total
Total								**$ -**

University	University of Newcastle	P/N		Part Cost	$ 81.19
Competition Code	FSAE-A	System	Electrical	QTY	1
Year	2023	Assembly	Looms	Extended Cost	$ 81.19
Car #	E03	Part	LV Loom, PEN-Pedals		

Materials

ID	Material	Description	Size 1	Unit 1	Size 2	Unit 2	Area Name	Area (mm^2)	Length (mm)	Density (kg/m^3)	QTY	Unit Cost	Sub Total
	Wire, Power	Power wires	3.2	m		-					1	$ 3.00	$ 9.60
	Wire, Signal	Signal wires	3.2	m		-					1	$ 1.00	$ 3.20
	Heat Shrink Tubing	Heat shrink for Looms	0.95	m		-					1	$ 0.50	$ 0.48
	Heat Shrink Tubing	heat shrink for boots	0.8	m		-					1	$ 0.50	$ 0.40
	Connector, Single Wire	Crimps for connectors	1	wire		-					1	$ 0.05	$ 0.05
	Connector, OEM Quality	DT Connector- PEN	12	pin(s)		-					1	$ 0.50	$ 6.00
	Connector, OEM Quality	DT Connector-Pedals	3	pin(s)		-					4	$ 0.50	$ 6.00
	Connector, OEM Quality	Pedal connector	6	pin(s)		-						$ 0.50	$ -
	Connector, OEM Quality	brake sensor connector	3	pin(s)		-					2	$ 0.50	$ 3.00
	Connector, Single Wire	spade terminals	1	wire		-					2	$ 0.05	$ 0.10
Total													**$ 28.83**

Processes

Process Multipliers

ID	Process	Use	Unit	QTY	Multiplier	Mult. Val.	Multiplier 2	Mult. Val. 2	Unit Cost	Sub total
	Attach Wire, Quick connect terminal		unit	1	-	1	-	1	$ 0.10	$ 0.10
	Attach Wire, Terminated wire with screw		unit	1	-	1	-	1	$ 0.35	$ 0.35
	Shrink Tube		cm	175	-	1	-	1	$ 0.15	$ 26.25
	Strip Wire		unit	42	-	1	-	1	$ 0.08	$ 3.50
	Cut wire		unit	-		1	-	1	$ 0.08	$ -
	Crimp Wire		unit	42	-	1	-	1	$ 0.17	$ 7.00
	Connector Assembley, Crimp		contacts	42	-	1	-	1	$ 0.36	$ 15.12
	Insert Bundle Into Tube or Sleeve		m	1.75	-	1	-	1	$ 0.02	$ 0.04
Total										**$ 52.36**

Fasteners

ID	Fasteners	Use	Size 1	Unit 1	Size 2	Unit 2	QTY	Unit Cost	Sub Total
Total									**$ -**

Tooling

ID	Tooling	Use	Unit	QTY	PVF	Frac. Incl	Unit Cost	Sub Total
Total								**$ -**

University	University of Newcastle	P/N		Part Cost	$ 26.08
Competition Code	FSAE-A	System		QTY	1
Year	2023	Assembly		Extended Cost	$ 26.08
Car #	E03	Part	HV Loom, CEN-DCDC		

Materials

ID	Material	Description	Size 1	Unit 1	Size 2	Unit 2	Area Name	Area (mm^2)	Length (mm)	Density (kg/m^3)	QTY	Unit Cost	Sub Total
	Wire, Power	Power wires for CEN-DCDC	0.6	m		-					1	$ 3.00	$ 1.80
	Connector, High Power, > 2 Amps	2 pin connector (DCDC)	2	pin(s)		-					2	$ 2.00	$ 8.00
	Connector, High Power, > 2 Amps	3 pin connector (CEN)	3	pin(s)		-					1	$ 2.00	$ 6.00
	Heat Shrink Tubing	Heat shrink for loom	0.3	m		-					1	$ 0.50	$ 0.15
	Heat Shrink Tubing	Heat shrink boots	0.1	m		-					2	$ 0.50	$ 0.10
Total													**$ 16.05**

Processes

Process Multipliers

ID	Process	Use	Unit	QTY	Multiplier	Mult. Val.	Multiplier 2	Mult. Val. 2	Unit Cost	Sub total
	Cut wire	Cut power wires	unit	1	-	1	-	1	$ 0.08	$ 0.08
	Strip Wire	Strip each end of wire	unit	2	-	1	Repeat 2	2	$ 0.08	$ 0.33
	Crimp Wire	Crimp each end of wire	unit	2	-	1	Repeat 2	2	$ 0.17	$ 0.67
	Insert Bundle Into Tube or Sleeve	Insert loom into heat shrink	m	0.5	-	1	-	1	$ 0.02	$ 0.01
	Connector Assembley, Crimp	Insert crimped ends of wires into connectors	contacts	2	-	1	Repeat 2	2	$ 0.36	$ 1.44
	Shrink Tube	Shrink heat shrink for loom and boots	cm	50	-	1	-	1	$ 0.15	$ 7.50
Total										**$ 10.03**

Fasteners

ID	Fasteners	Use	Size 1	Unit 1	Size 2	Unit 2	QTY	Unit Cost	Sub Total
Total									**0**

Tooling

ID	Tooling	Use	Unit	QTY	PVF	Frac. Incl	Unit Cost	Sub Total
Total								**0**

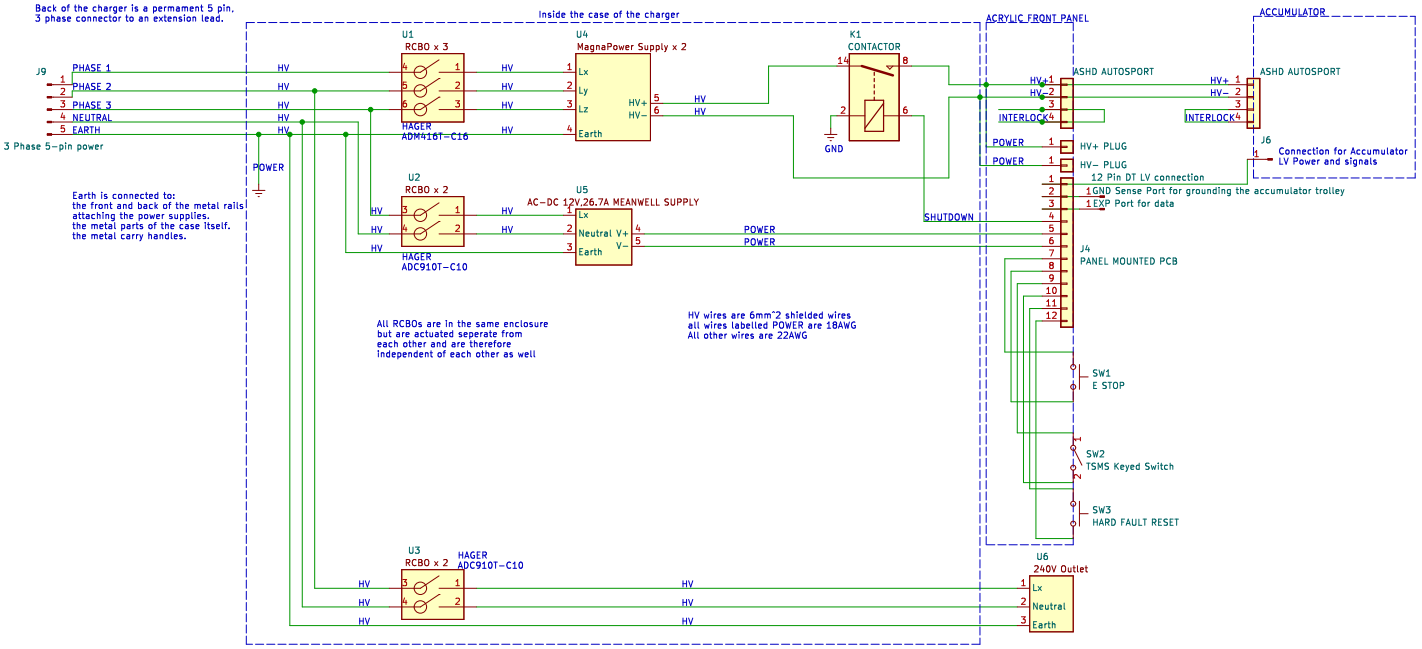
12 Appendix F: Competition Point Analysis

FSAE COMP RESULTS

Event	2023 (7th)	2022 (4th)	Delta	Out of
Presentation	72.55	33.16	+39.39	75
Cost	51.06	23.11	+27.95	100
Design	130.33	105	+25.33	150
Skid pad	65.77	56.06	+9.71	75
Acceleration	50.92	15.13	+35.79	100
Autocross	120.99	93.44	+27.55	125
Endurance*	9	159.99	-150.99	275
Efficiency	52.52	85.26	-32.74	100
Total	**553.14**	**571.15**	**-18.01**	**1000**

* Projected Points from Endurance based on our average lap time would be approximately 255.

13 Appendix G: Charger Block Diagram


Back of the charger is a permament 5 pin, 3 phase connector to an extension lead.
Inside the case of the charger
ACRYLIC FRONT PANEL
ACCUMULATOR
J9
PHASE 1
PHASE 2
PHASE 3
NEUTRAL
EARTH
3 Phase 5-pin power
HV
POWER
U1
RCBO x 3
HAGER
ADM416T-C16
U2
RCBO x 2
HAGER
ADC910T-C10
U3
RCBO x 2
HAGER
ADC910T-C10
U4
MagnaPower Supply x 2
Lx
Ly
Lz
HV+
HV-
Earth
U5
AC-DC 12V,26.7A MEANWELL SUPPLY
Lx
Neutral V+
V-
Earth
K1
CONTACTOR
GND
SHUTDOWN
ASHD AUTOSPORT
INTERLOCK
HV+ PLUG
HV- PLUG
12 Pin DT LV connection
GND Sense Port for grounding the accumulator trolley
EXP Port for data
J4
PANEL MOUNTED PCB
SW1
E STOP
SW2
TSMS Keyed Switch
SW3
HARD FAULT RESET
J6
Connection for Accumulator LV Power and signals
U6
240V Outlet
Lx
Neutral
Earth
Earth is connected to:
the front and back of the metal rails attaching the power supplies.
the metal parts of the case itself.
the metal carry handles.
All RCBOs are in the same enclosure but are actuated seperate from each other and are therefore independent of each other as well
HV wires are 6mm^2 shielded wires
all wires labelled POWER are 18AWG
All other wires are 22AWG

14 Appendix H: Charger ECU Code

Listing 11: Charger ECU V1 Code

```
// CHARGER ECU V1
// Joshua Wenham

// Include necessary libraries and define pins
#include <EV3_CAN.h>
#define IMD_PIN 4
#define BMS_PIN 6
#define PRECHARGE_PIN 7
#define HFR_LED 9
#define SHUTDOWN_TSMS_PIN 17
#define SHUTDOWN_INTERPOSE_PIN 19
#define SHUTDOWN_INTERLOCK_PIN 21

// Initialise and declare variables
float imdOKHS = 0;              // [boolean] CAN output for logging and dash lights
float bmsOKHS = 0;              // [boolean] CAN output for logging and dash lights
float prechargeOKHS = 0;        // [boolean] CAN output for logging
float sdTSMS = 0;            // [boolean] CAN output shutdown logging
float sdInterpose = 0;
float sdInterlock = 0;

//CAN 1 variables
//inputmsgs defines message, outputVar defines location for incoming data
int numreceive = 0;
int numsend = 7;

void setup()
{
  Serial.begin(9600);
 // Set pin modes
  pinMode(IMD_PIN, INPUT);
  pinMode(BMS_PIN, INPUT);
  pinMode(PRECHARGE_PIN, INPUT);
  pinMode(HFR_LED, OUTPUT);
  pinMode(SHUTDOWN_TSMS_PIN, INPUT);
  pinMode(SHUTDOWN_INTERPOSE_PIN, INPUT);
  pinMode(SHUTDOWN_INTERLOCK_PIN, INPUT);

 //CAN bus initialisation
  NUCAN_initialise(numsend, numreceive, 250000);
}

void loop()
{
  EVERY_N_MILLIS(100){
    updateCharge();
    updateLogged();
    serialOut();
  }
```

```
   NUCAN_heartbeat(&HB_CHG);
}

void updateCharge(void){
      NUCAN_write(&CHARGE, 1);
}

void updateLogged(void){
      imdOKHS = digitalRead(IMD_PIN);// read the input from the IMD
      NUCAN_write(&IMD_OKHS, imdOKHS);

      bmsOKHS = digitalRead(BMS_PIN);// read the input from the BMS
      NUCAN_write(&AMS_OKHS, bmsOKHS);

      prechargeOKHS = digitalRead(PRECHARGE_PIN);// read the input from the Precharge
      NUCAN_write(&PRECHARGE_OKHS, prechargeOKHS);

      sdTSMS = digitalRead(SHUTDOWN_TSMS_PIN);// read the input from the TSMS
      NUCAN_write(&SD_TSMS, sdTSMS);

      sdInterpose = digitalRead(SHUTDOWN_INTERPOSE_PIN);// read the input from the Interpose
      NUCAN_write(&SD_INTERPOSE, sdInterpose);

      sdInterlock = digitalRead(SHUTDOWN_INTERLOCK_PIN);// read the input from the Interlock
      NUCAN_write(&SD_CEN_INTERLOCK, sdInterlock);
}

void serialOut(void){
    Serial.println("---------------");
    Serial.print("HARD␣FAULTS:␣PDOC␣=␣"); Serial.println(prechargeOKHS);
    Serial.print("HARD␣FAULTS:␣IMD␣=␣"); Serial.println(imdOKHS);
    Serial.print("HARD␣FAULTS:␣BMS␣=␣"); Serial.println(bmsOKHS);
    Serial.print("TSMS␣=␣"); Serial.println(sdTSMS);
    Serial.print("Interpose␣=␣"); Serial.println(sdInterpose);
    Serial.print("Interlock␣=␣"); Serial.println(sdInterlock);
    Serial.println("---------------");


}
```