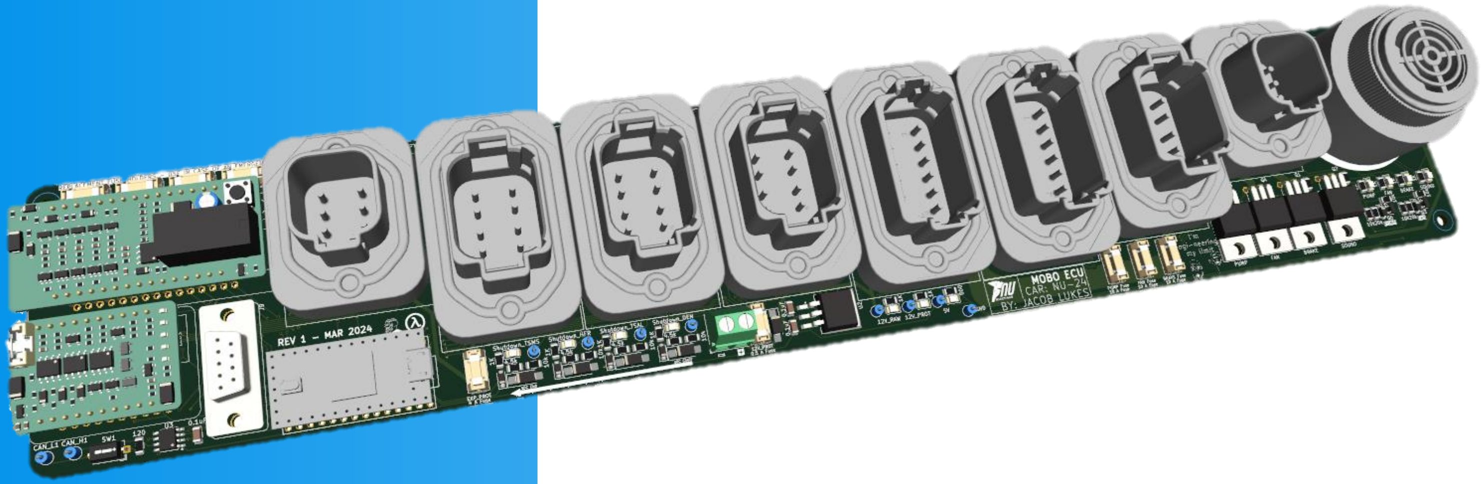
ENGG3200 Report

Directed Reading

NU24



Jacob Lukes

2024

Contents

List of Figures

Introduction

In the first half of 2024, a task was scoped to design, manufacture, and commission an iteration of NU23's Central Electric Node (CEN) Printed Circuit Board (PCB), created by Imel Munday. This iteration would improve upon the complexities and short comings of the board. The board, named the MOTHER board, would also take on the functionality of NU23's Rear Electronic Node (REN), created by Nick Lyall, thus eliminating the need for this PCB on NU24.

The goal of this project was to gain an in-depth understanding to how NU Racing and its internal systems operate. Entailed in this project was designing, manufacturing, commissioning and error testing the MOTHER board, helping other team members error test their PCBs and help with the set up and management of test days.

The creation process of the PCB was greatly aided by the help of Alexander Gregg, Malcom Sidney, Jacob Bush, and Alec Chapman.

Initial System Design

Background

NU23’s CEN distributed low voltage power and routed signals throughout the car. It managed the Hard Fault Reset (HFR) functionality, Brake System Plausibility Device (BSPD), Tractive System Active Light (TSAL) Driver and the Discharge Circuit. The existing system was cumbersome and required significant manufacturing time to make small changes and fixes to any of these subsystems. The final board is shown in Figure 1 and Figure 2, with a labelled schematic shown in Figure 3 and Figure 4.

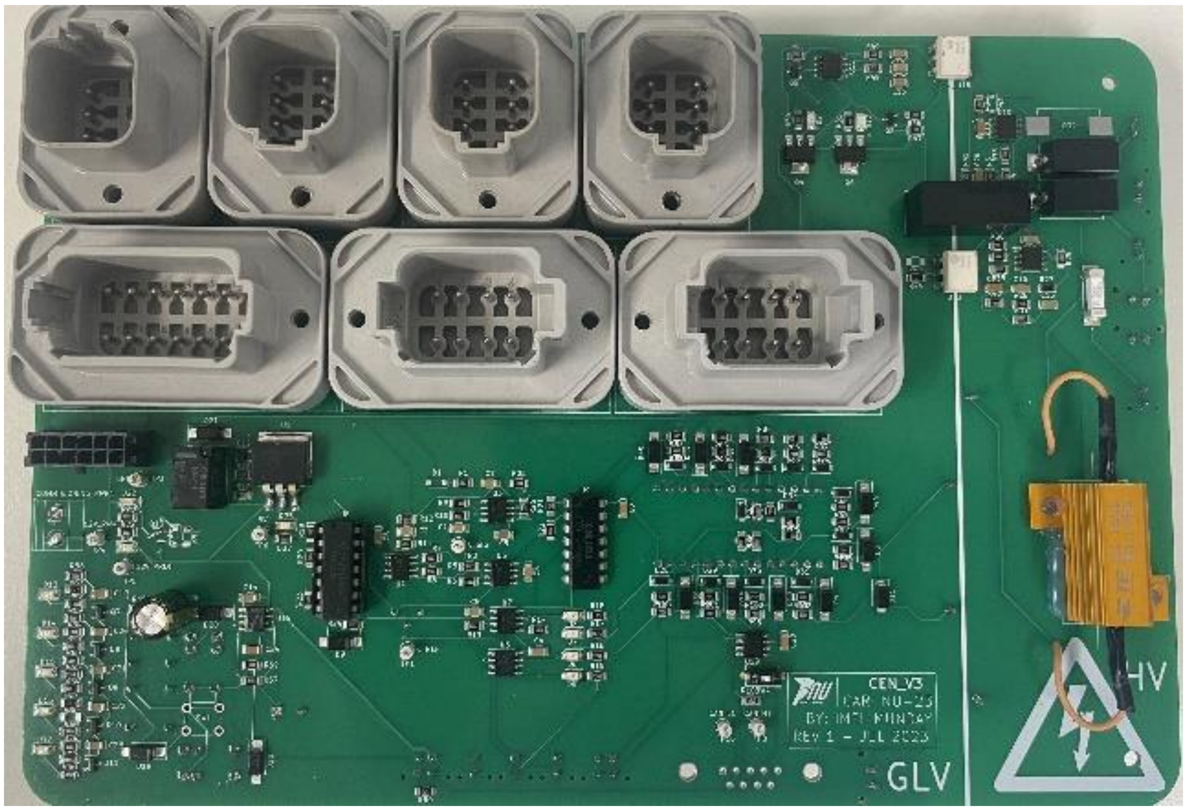


Figure 1 - 2023 CEN Board Front

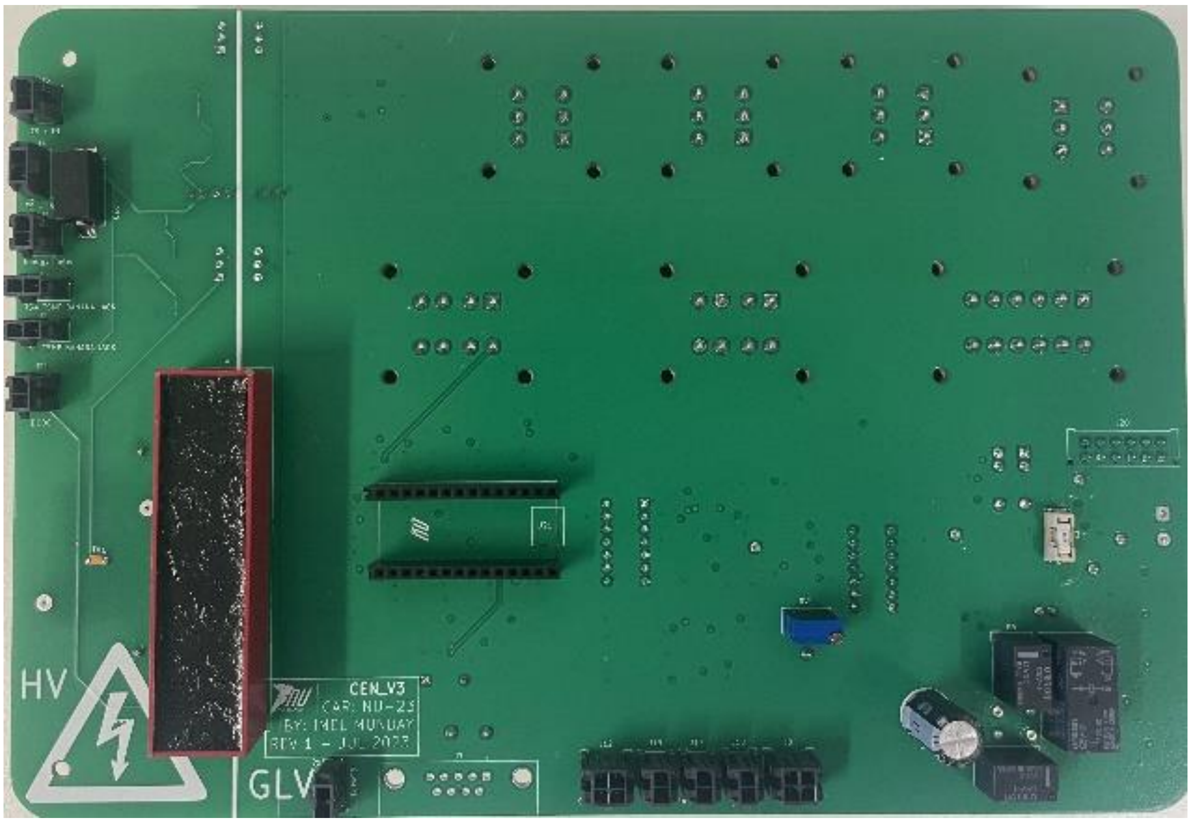


Figure 2 - 2023 CEN Board Back

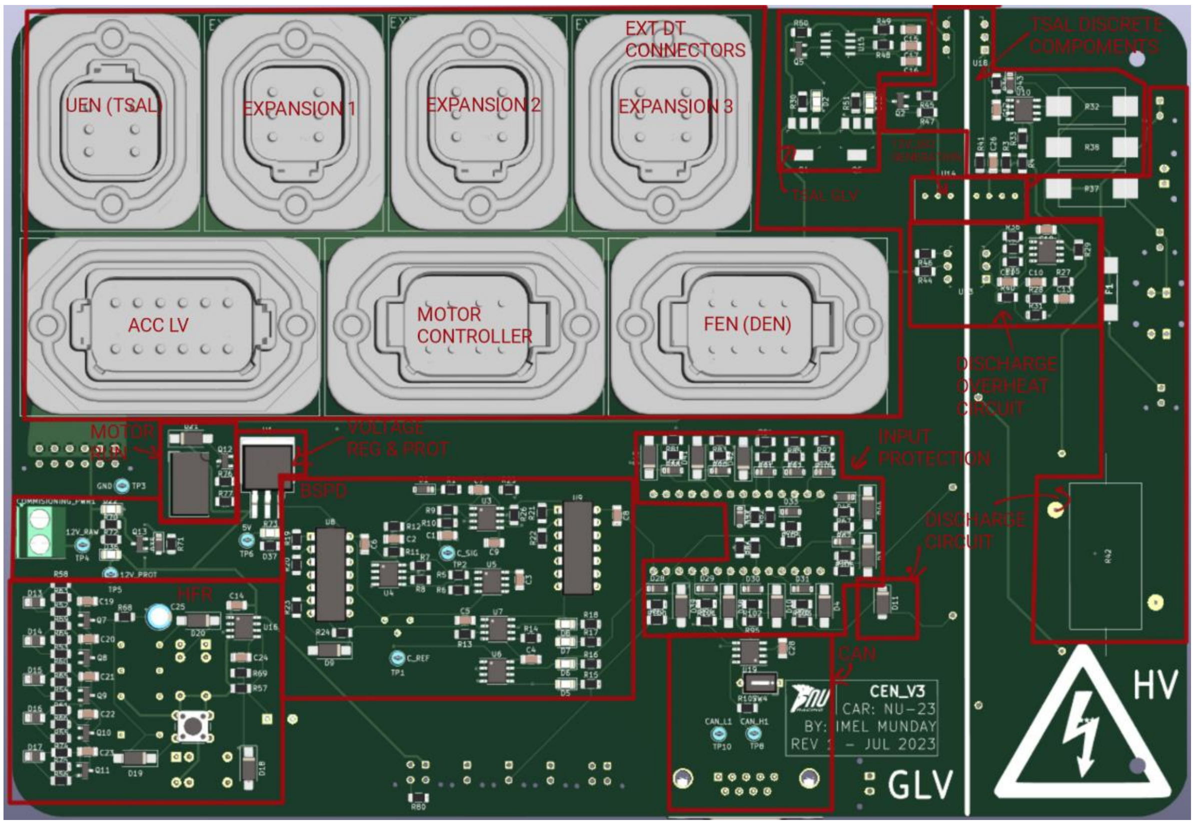


Figure 3 - 2023 CEN Board Annotated Front

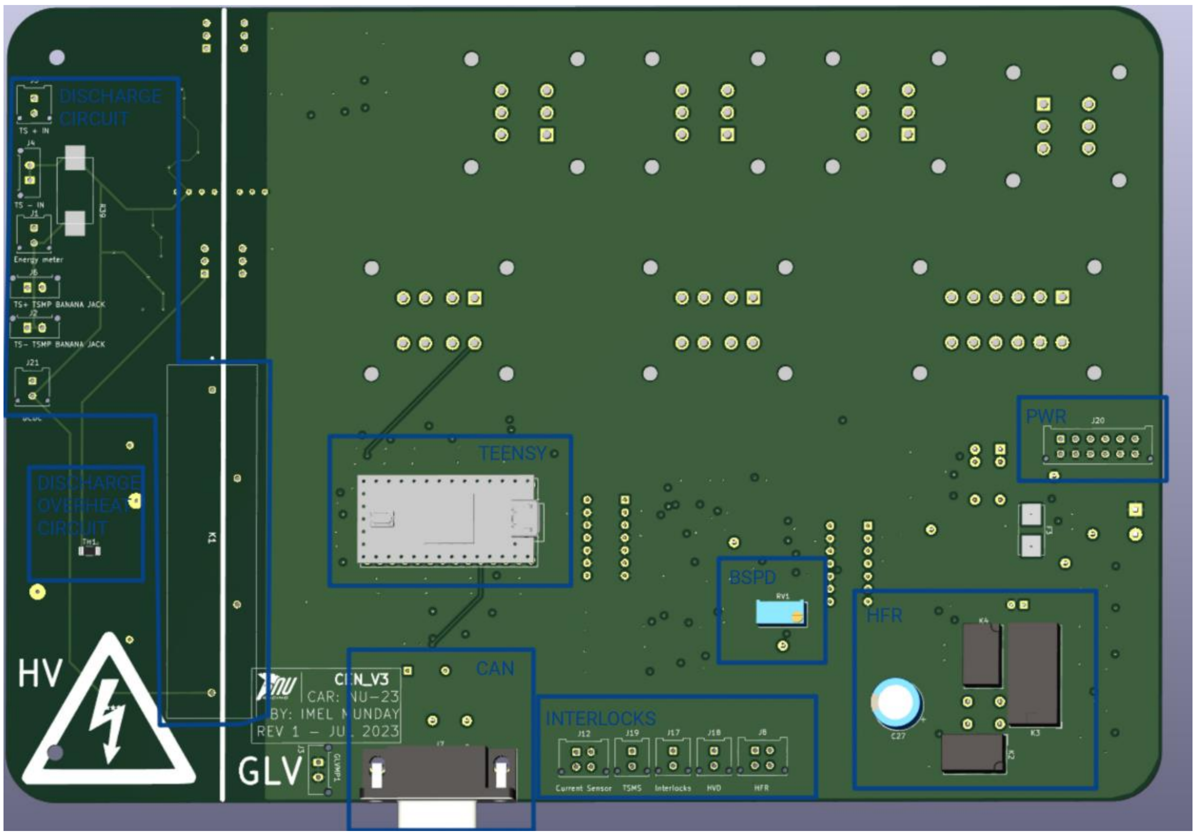


Figure 4 - 2023 CEN Board Annotated Back

Scope

NU24’s CEN has been split into two main sections, one section created by the author which kept the name CEN and the other section created by Marisa McLean, called Human Interface Panel (HIP). The CEN contained the MOTHER board PCB which handled the distribution of low voltage (LV) power and signals.

The MOTHER board contained a teensy microcontroller along with two smaller PCB breakout boards which attached through pin headers. These breakout boards managed the Hard Fault Reset (HFR) and Brake Sensor Plausibility Device (BSPD), both created by Jacob Bush, who has created a report and documentation on how these circuits work. The MOTHER board also managed the cooling system, brake light and sounder switching functionality of the REN.

The HIP handled the high voltage (HV) power, TSAL driver circuit, discharge circuit and the two master switches, LV and TSMS. This new design is shown in Figure 5. This report is based around the design, manufacture, and commission of the MOTHER board.

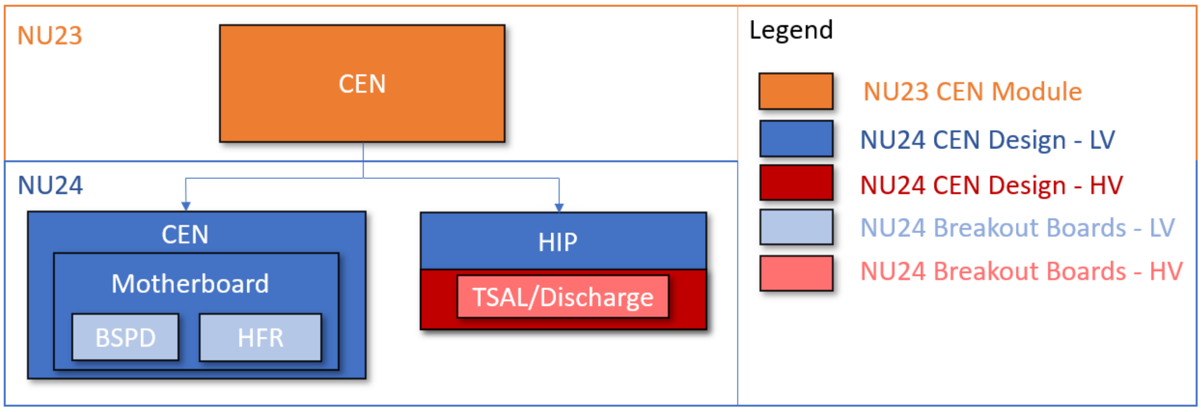


Figure 5 NU24's CEN breakdown

Initial Design

Multiple meetings were had with the Lead Mechatronics Engineer, Alec Chapman, detailing how the board will function, the role it will play and what component groups were required to be present. These were found to be:

- Teensy.
- Power input stage.
- Voltage input protection stage for logging the shutdown circuit.
- Fuses to protect circuits that would power components outside of the PCB.
- Various sized DT Connectors.
- Metal-Oxide Semiconductor Field-Effect Transistor (MOSFET) circuits to switch the signals for the cooling, sounder and brake light circuits.
- CAN Bus network integration.

Provided will be an explanation of the circuits that require more information and are not included in the NU Teams Standard Circuits KiCad Library or have been manipulated to what is given in the NU Teams KiCad Library, these include:

- The MOSFET circuits.
- Shutdown circuit voltage protection.
- Fuses to protect other circuits.
- The breakout boards.

Metal-Oxide Semiconductor Field-Effect Transistor (MOSFET) Switching Circuits

These MOSFET circuits came from the REN that was located at the rear of NU23. The base circuit topology was copied from the original REN PCB, all of which would switch the ground connection (referred to as low-side power switching). On this PCB, these circuits experienced a lot of heat generation, around 120 °C, in some cases getting so hot that one managed to de-solder itself from the PCB. In some situations, causing the MOSFETs to become unreliable and fail. Multiple changes were made to these circuits, detailed below.

Each of the circuits used an IRLB8721PBF MOSFET, which is a “normally open” MOSFET. This MOSFET was chosen due to multiple reasons, these being its high availability at NU Teams, that it was previously tested and used on NU23, and its greater reliability than the other option; a MOSFET called 2N7002.

In addition to the IRLB8721PBF MOSFET, the cooling circuit topologies used an extra 2N7002 MOSFET, which is also a “normally open” MOSFET. This was also chosen due to the same reasons as the IRLB8721PBF MOSFET.

Brake Light and Sounder Switching Circuit

The circuit topology that was used to power both the brake light and sounder on the REN was kept the same, due to both circuits not having a high amount of current running through them. This circuit consisted of a IRLB8721PBF MOSFET connected in a low-side switching configuration, with a Light-Emitting-Diode (LED) and resistor to indicate that the MOSFET works correctly. It also consisted of a pull-down resistor connected to the Gate and Source pins, used to eliminate the possibility of floating voltages.

The sounder that was chosen was the CPI-4232-92FST, due to its use and success on NU23's REN.

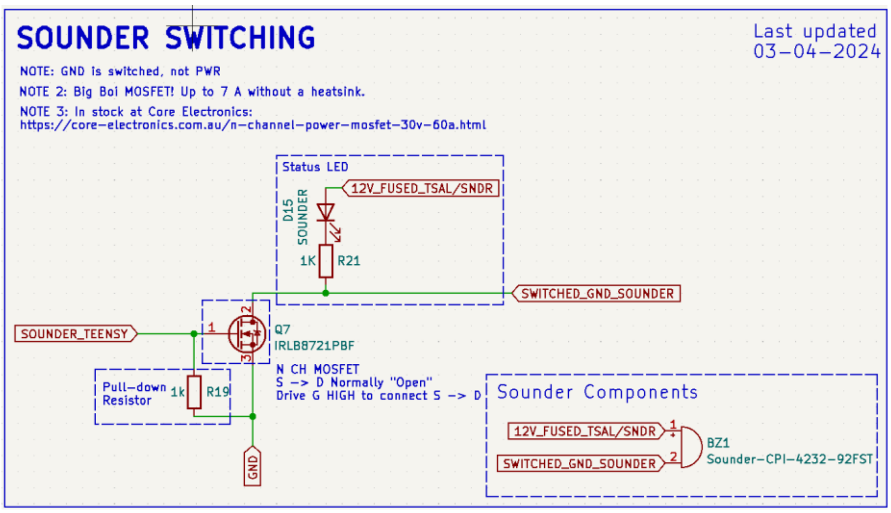


Figure 6 Sounder Switching Circuit Topology

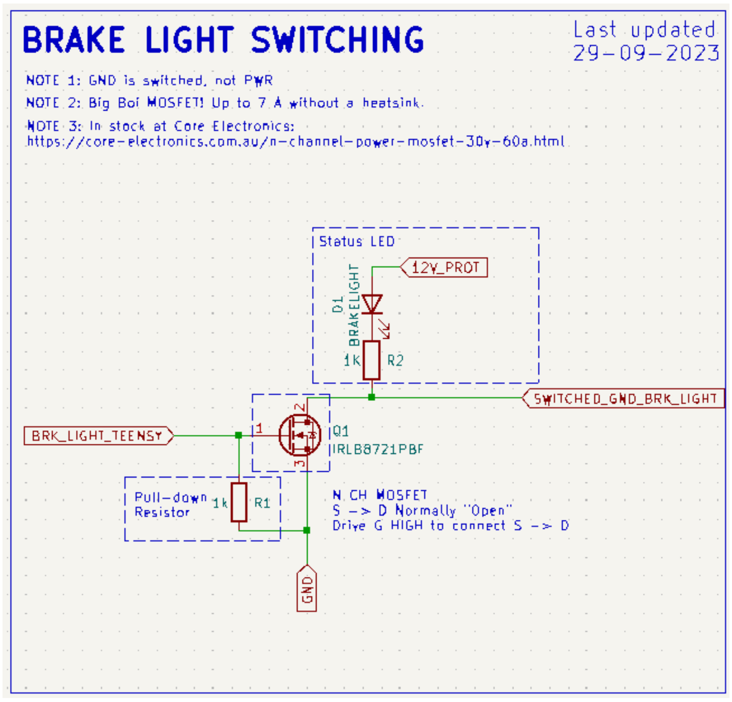

BRAKE LIGHT SWITCHING
Last updated 29-09-2023
NOTE 1: GND is switched, not PWR
NOTE 2: Big Boi MOSFET! Up to 7 A without a heatsink.
NOTE 3: In stock at Core Electronics: https://core-electronics.com.au/n-channel-power-mosfet-30v-60a.html
Status LED
12V_PROT
D1 BRAKELIGHT
1K R2
SWITCHED_GND_BRK_LIGHT
BRK_LIGHT_TEENSY
Q1 IRLB8721PBF
N CH MOSFET
S -> D Normally "Open"
Drive G HIGH to connect S -> D
Pull-down Resistor
1k R1
GND

Figure 7 Brake Light Switching Circuit Topology

Cooling Switching Circuit

The cooling switching circuit powered the pumps and radiator fan attached to the back of NU23, which consisted of a single IRLB8721PBF, an LED and a pull-down resistor. It was the circuit that generated the most heat due to the high current draw of the connected pumps and radiator fan. Through testing, the total current that these devices drew was found to be around 12.54 A: around 6.24 A from the pumps and around 6.28 A from the radiator fan.

To improve this circuit, Jacob Bush suggested that the high temperatures occurred due to a combination of two factors. One being the high current that the circuit was switching and the other being that the voltage that drove the Gate pin of the MOSFET was too low (a teensy 3.3 V signal), compared to the voltage connected to the Drain and Source pins (12 V, the level required to power the components).

It was decided that the new circuit would be split in two, having one circuit to power the pumps, and one to power the fans. Jacob Bush also added that a solution would be to implement a smaller MOSFET called a 2N7002, which would not generate as much heat as it would switch the IRLB8721PBF with a 12 V signal instead of a 3.3 V one. This introduction of the 2N7002 MOSFET caused inverse logic to occur. The 2N7002 was added into both the pumps and fan circuit.

The explanation of the inverse logic was found to be hard to convey, so an explanation that best describes why will be included along with an image of the two states that the circuit can be in, shown in Figure 8.

Written Explanation of how the 2N7002 driving an IRLB8721PBF circuit operates:

- ON STATE: When the Gate pin of the 2N7002 is driven low, the Source and Drain pins will remain in their “normally open” state. This caused the Gate pin of the IRLB8721PBF to be driven high, connecting it’s Drain and Source pins, and letting current flow between them through to ground, which will turn the LED on.
- OFF STATE: When the Gate pin of the 2N7002 is driven high, the Source and Drain pins will change from their “normally open” state to a closed state, causing the Gate pin of the IRLB8721PBF to be driven low, leaving it’s Drain and Source pins in their “normally open” state, and the LED will remain off.

Visual Explanation of how the 2N7002 driving an IRLB8721PBF circuit operates:

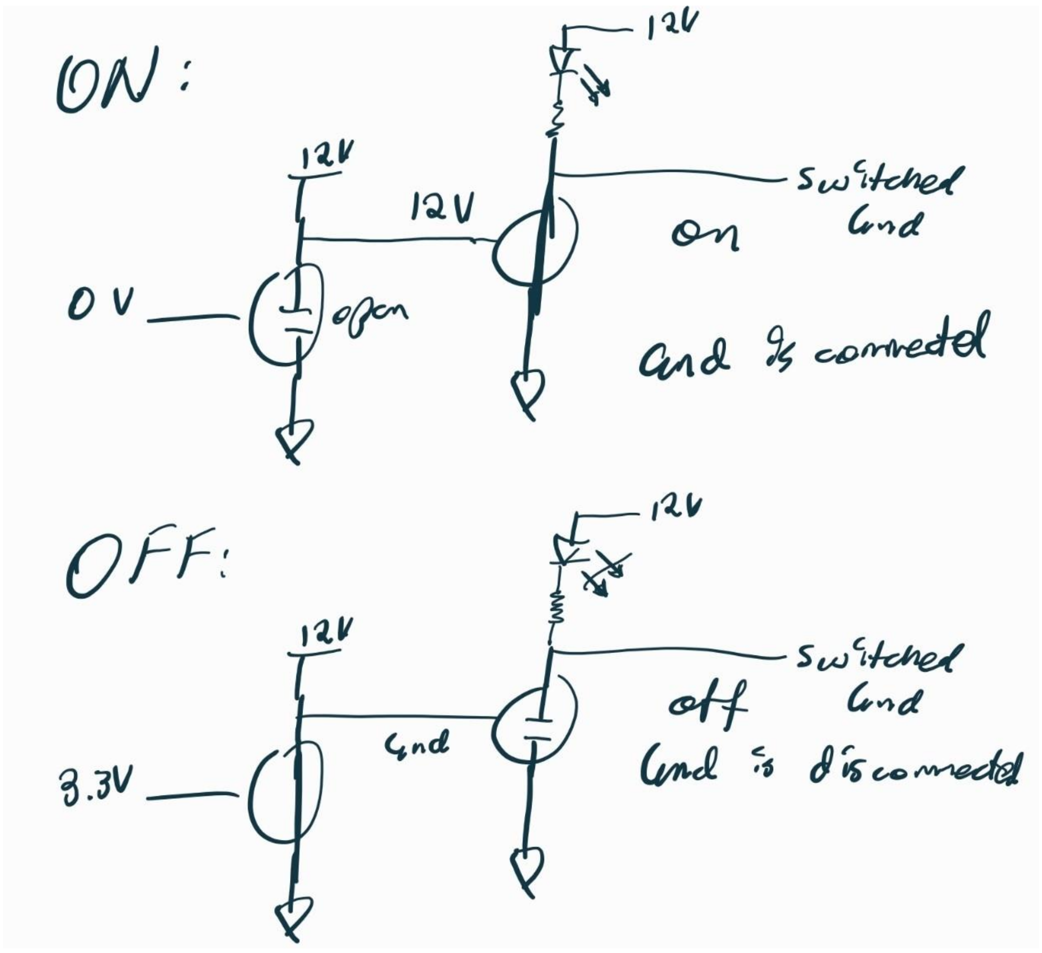


Figure 8 Visual Explanation of Cooling Circuit Inverse Logic

Cooling Switching Circuit Voltage Clamp LTspice validation

Another issue that arose due to the introduction of the 2N7002 MOSFET was in the case where the teensy was disconnected from the circuit when the PCB was running, the connected device would turn on. This occurred due to the 2N7002 MOSFET not being powered, causing the 12 V line to power the IRLB8721PBF MOSFET through its Gate pin, resulting in its Drain and Source pins connecting and grounding the load. To combat this, it was found that a voltage divider of 5 V to 3.3 V could be used, along with a 3.3 V Zener to ensure that it stays at 3.3 V. The three states of teensy signal, high, low, and disconnected, are shown through an LTspice simulation in Figure 9, Figure 10, and Figure 11. An arbitrary Zener diode was chosen, which did not have the correct breakdown voltage, but it was close enough.

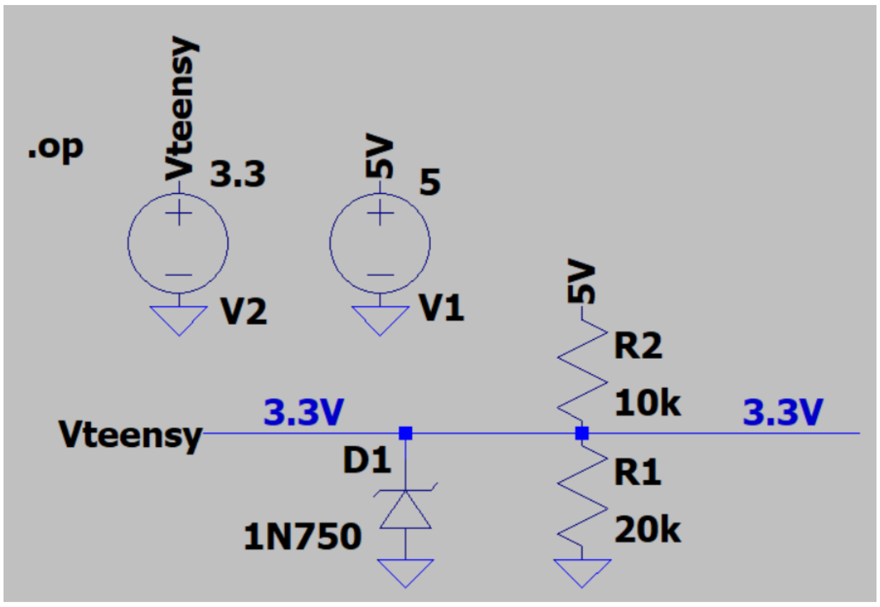


Figure 9 Double MOSFET Circuit Voltage Clamp Section, Teensy Signal High

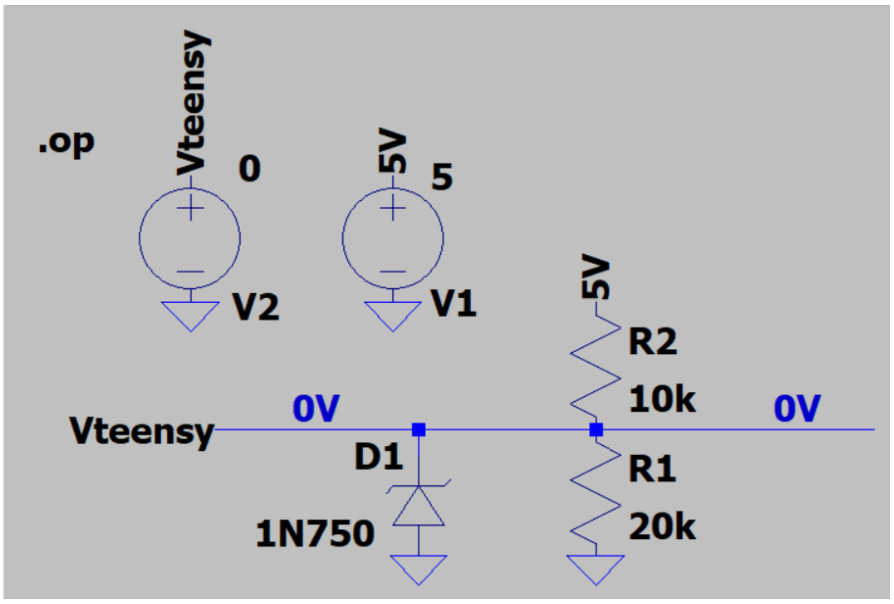


Figure 10 Double MOSFET Circuit Voltage Clamp Section, Teensy Signal Low

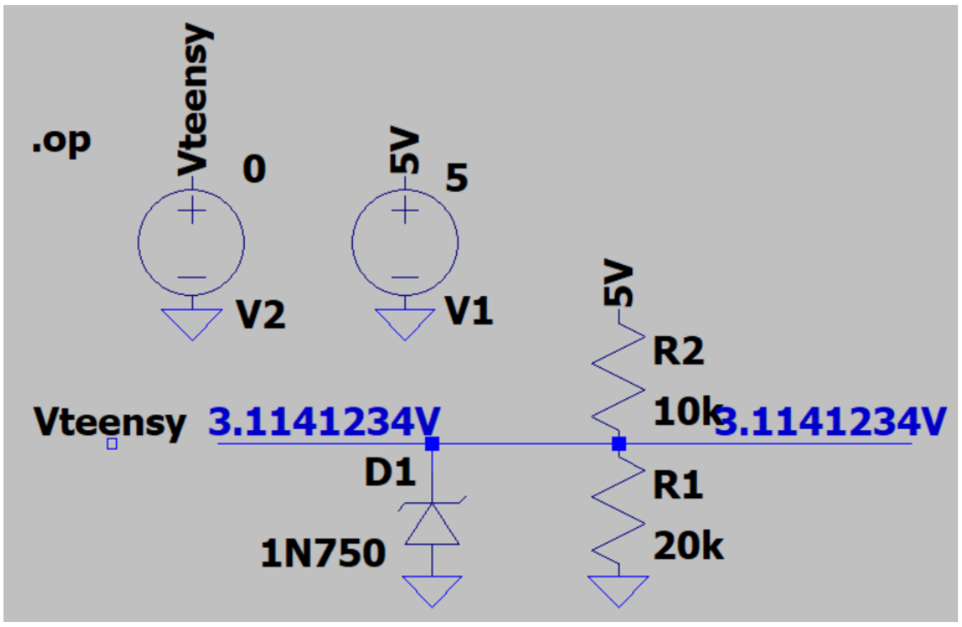


Figure 11 Double MOSFET Circuit Voltage Clamp Section, Teensy Signal Disconnected

Cooling Switching Circuit LTspice Validation

LTspice was used to validate the logic and functionality of the circuit, shown in Figure 12, Figure 13, and Figure 14. When Vo is at 12 V, the attached device is turned off since ground being switched. When Vo is close to 0 V, the connected device is on since it’s connected to ground. When the teensy signal is disconnected, the circuit is pulled close to 3.3 V, keeping the connected device off. Arbitrary N-Channel MOSFET, Zener diode and LED models were chosen, each of which were close to what would be used on the actual PCB, this caused inaccurate values to be output from the circuit.

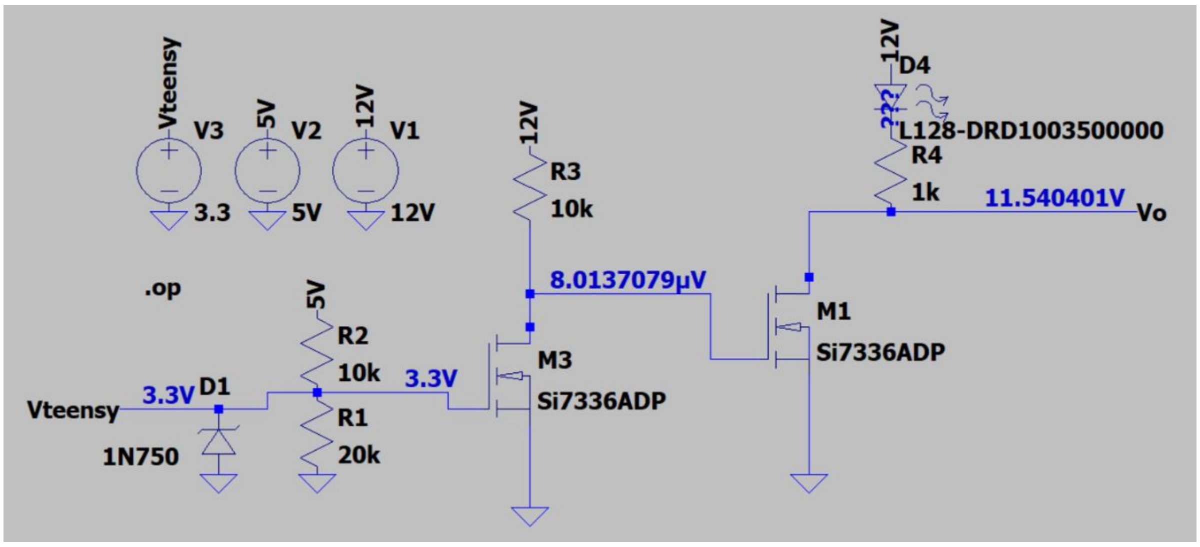


Figure 12 Double MOSFET Circuit Output When Teensy Signal is High

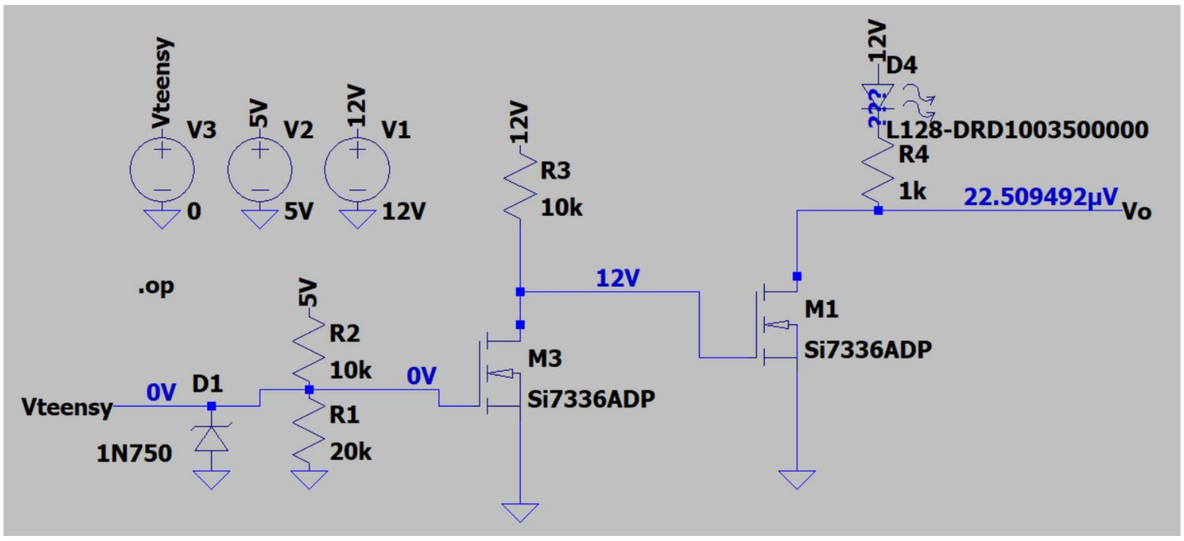


Figure 13 Double MOSFET Circuit Output When Teensy Signal is Low

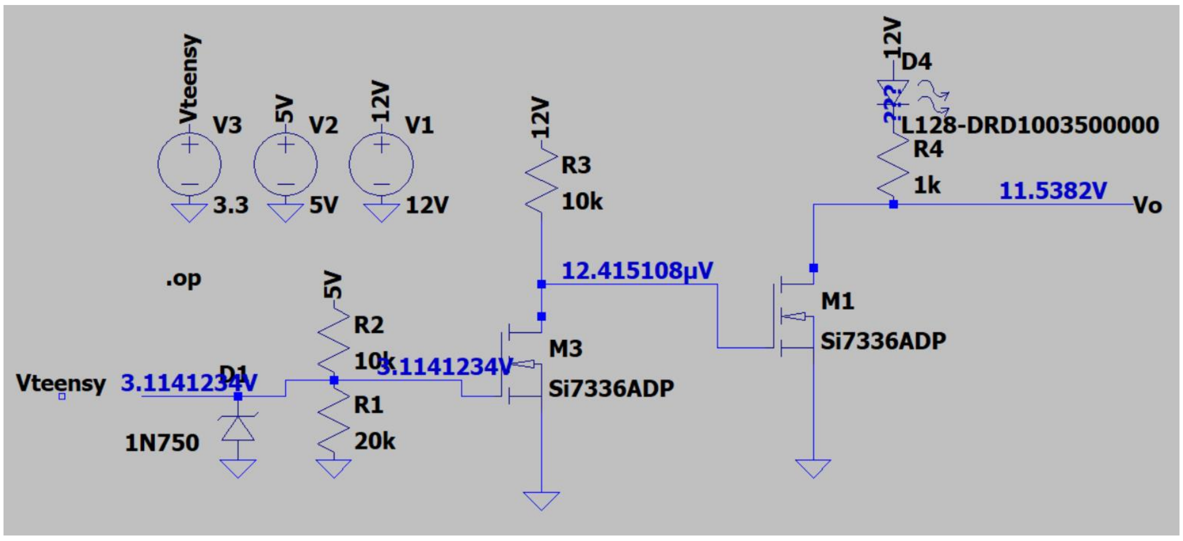


Figure 14 Double MOSFET Circuit Output When Teensy Signal is Disconnected

Both circuit topologies are shown in Figure 15.

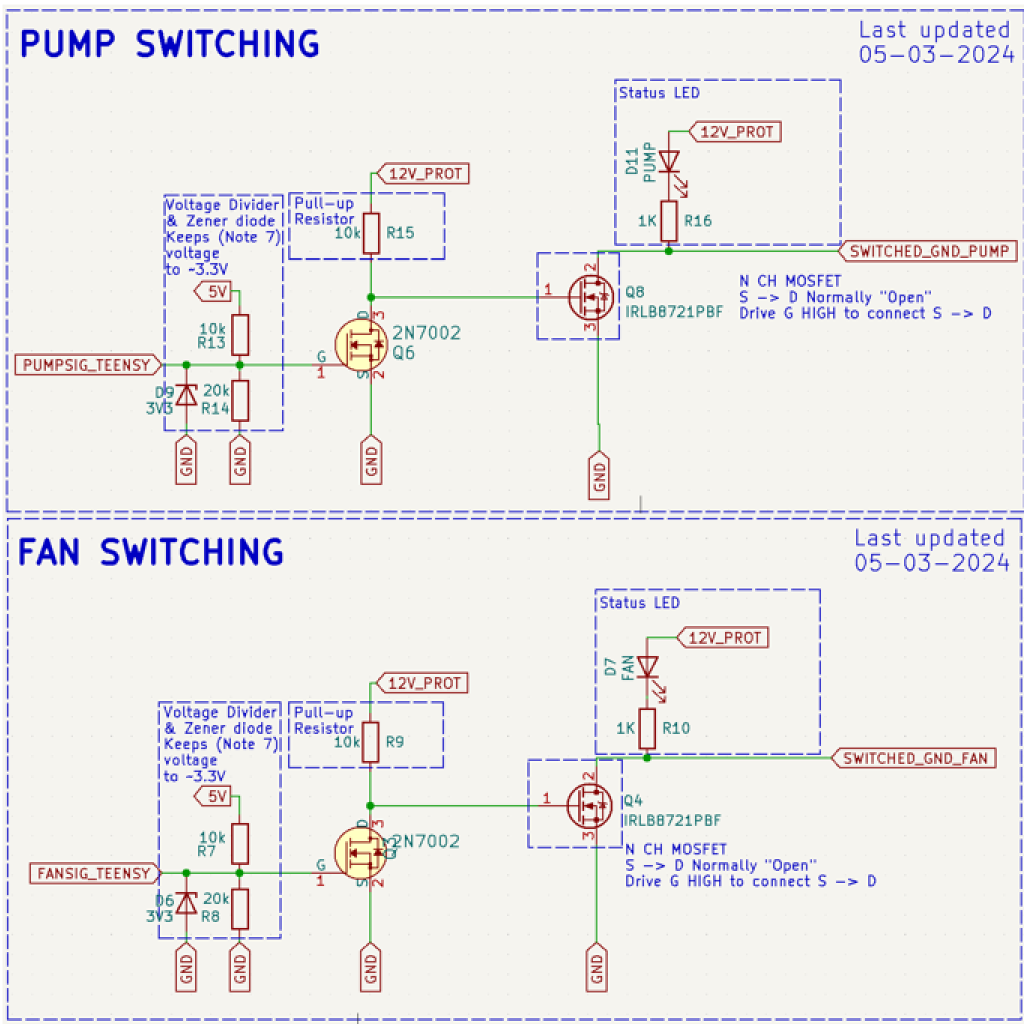


Figure 15 Cooling Switching Circuit Topology

Shutdown Circuit Voltage Protection

The shutdown circuit is a 12 V signal that runs throughout the car. If a fault occurs anywhere along the circuit, it will trip the car and cause it to turn off high voltage. There are certain points around the car where it will monitor for a trip, such as e-stop buttons.

Since the MOTHER board is the low voltage side of NU23's CEN, thus containing a teensy microcontroller, it had to manage and record each stage of the shutdown circuit that ran through it. It did this with an input protection circuit, shown in Figure 16, which consisted of a voltage divider to bring the 12 V signal down to under 3.3 V, along with a 3.3 V Zener diode in case anything went wrong with assembly or during operation. This precaution was implemented in the case where a voltage above 3.3 V was input into the teensy, it would cause the teensy to break, making it inconsistent during operation and ultimately making it unusable. The circuit also had a test point and status LED for ease of life when making sure that the circuit functioned correctly. There was also a reverse polarity protection diode implemented, shown as D29 in Figure 16, this diode was not completely necessary, but it was implemented so that if the teensy breaks, it has a reduced chance of causing false readings at other points in the shutdown circuit.

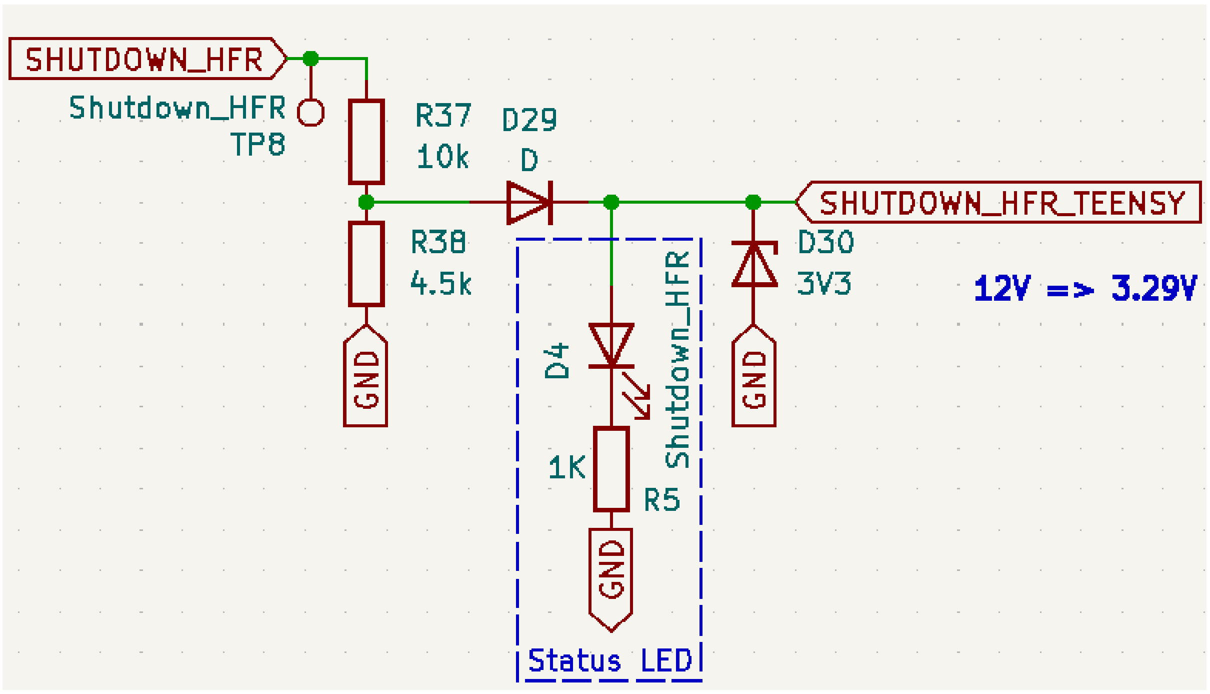


Figure 16 Shutdown Circuit Input Protection Circuit Example

An LTspice simulation was conducted to test the validity of this circuit, shown in Figure 17 which proved to work correctly. The component values were chosen to best match the components used on the PCB, which resulted in a value being output that is far lower than what will be output from the actual circuit in the test. This was due to the two diodes that were used had greater forward voltage drops than the ones that would be used on the PCB.

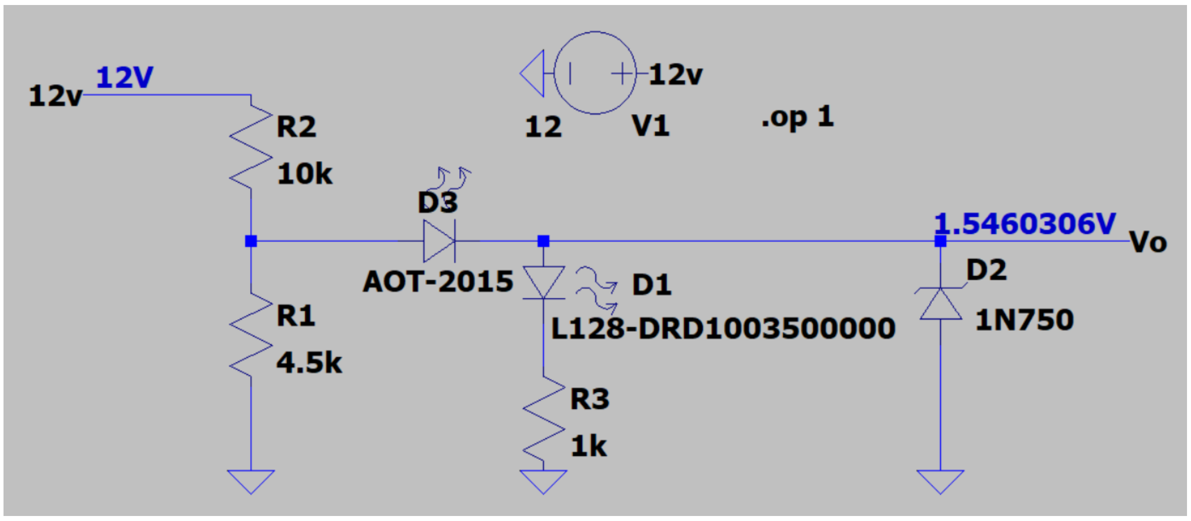


Figure 17 Shutdown Circuit Input Protection LTspice Validation

The shutdown circuit starts by coming in from the DEN, going up to the TSAL, coming back down through the HFR, and then down into the HIP and comes back as TSMS (Tractive System Master Switch). The return signal from the HIP was decided to be called TSMS since the HIP has two shutdown circuit points inside it, the HVD interlock and TSMS. The TSMS is the last check that it does before coming back into the CEN. The topology is shown in Figure 18.

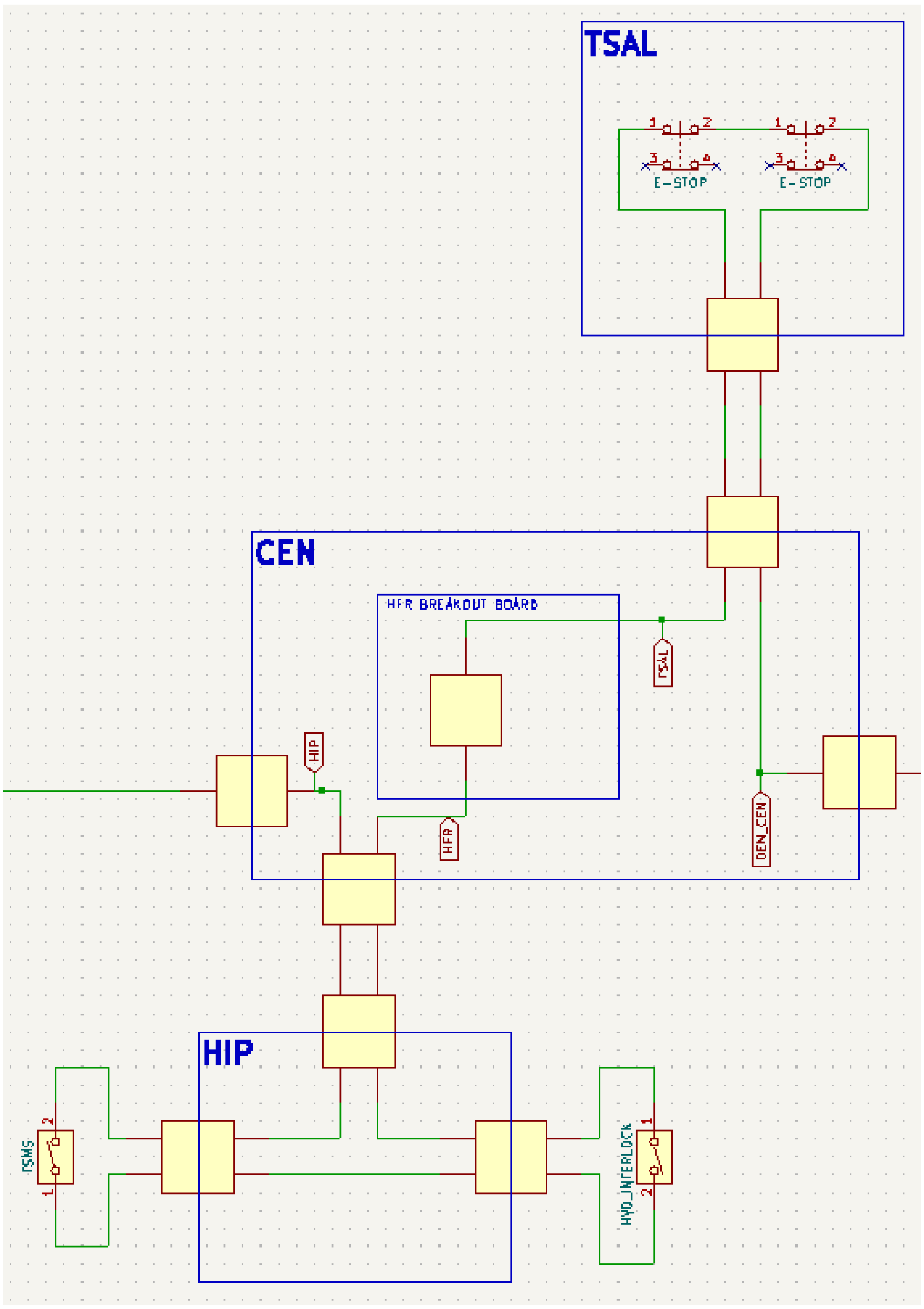


Figure 18 Shutdown Circuit in Proximity to the CEN

Fuses to Protect Other Circuits

There was a fuse holder block implemented due to the number of fuses that were necessary around the MOTHER board. These fuses were primarily used on circuits that leave the PCB, since if these circuits were to fault outside of the board, they could cause the board itself to fault and possibly break. Extra fuse holders were also added so that if a fuse broke during testing on track, a new one could be swapped in quickly. The layout of the fuse block can be seen in Figure 19.

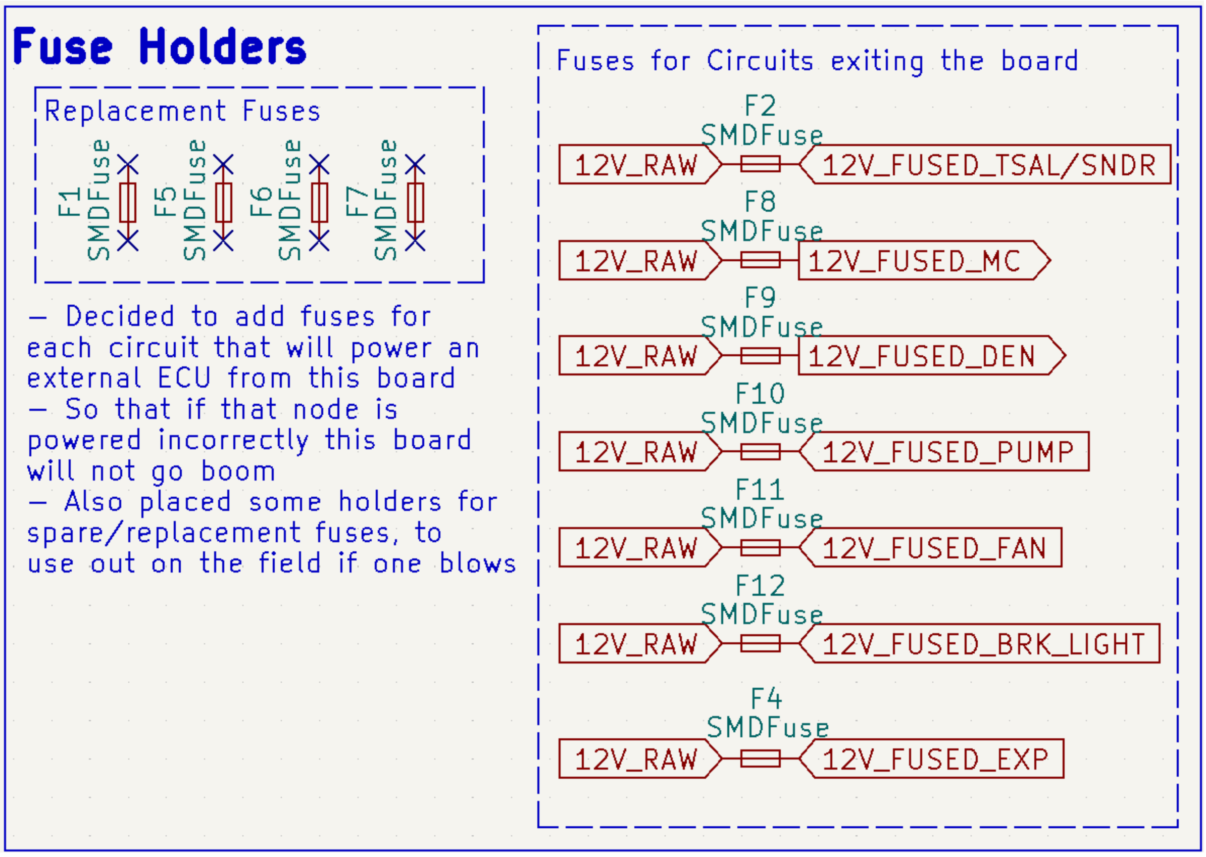

Fuse Holders
Replacement Fuses
F1 SMDFuse
F5 SMDFuse
F6 SMDFuse
F7 SMDFuse
– Decided to add fuses for each circuit that will power an external ECU from this board
– So that if that node is powered incorrectly this board will not go boom
– Also placed some holders for spare/replacement fuses, to use out on the field if one blows
Fuses for Circuits exiting the board
F2 SMDFuse 12V_RAW 12V_FUSED_TSAL/SNDR
F8 SMDFuse 12V_RAW 12V_FUSED_MC
F9 SMDFuse 12V_RAW 12V_FUSED_DEN
F10 SMDFuse 12V_RAW 12V_FUSED_PUMP
F11 SMDFuse 12V_RAW 12V_FUSED_FAN
F12 SMDFuse 12V_RAW 12V_FUSED_BRK_LIGHT
F4 SMDFuse 12V_RAW 12V_FUSED_EXP

Figure 19 Fuses Block

Breakout Boards

With the new iteration of the CEN, the Brake Sensor Plausibility Device (BSPD) and Hard Fault Reset (HFR) circuits have been relocated onto smaller PCBs. It was done since these two circuits are more complex and intricate than the rest of the circuits on the MOTHER board. These two smaller PCBs where designed, manufactured and mostly commissioned by Jacob Bush, a former Final Year Project (FYP) student. Jacob Bush has written an in-depth report on these two boards, as well as another breakout board that is located on the HIP, so little information will be contained within this report about these two PCBs. Figure 20 shows the schematic symbols for the BSPD and HFR input and output signals.

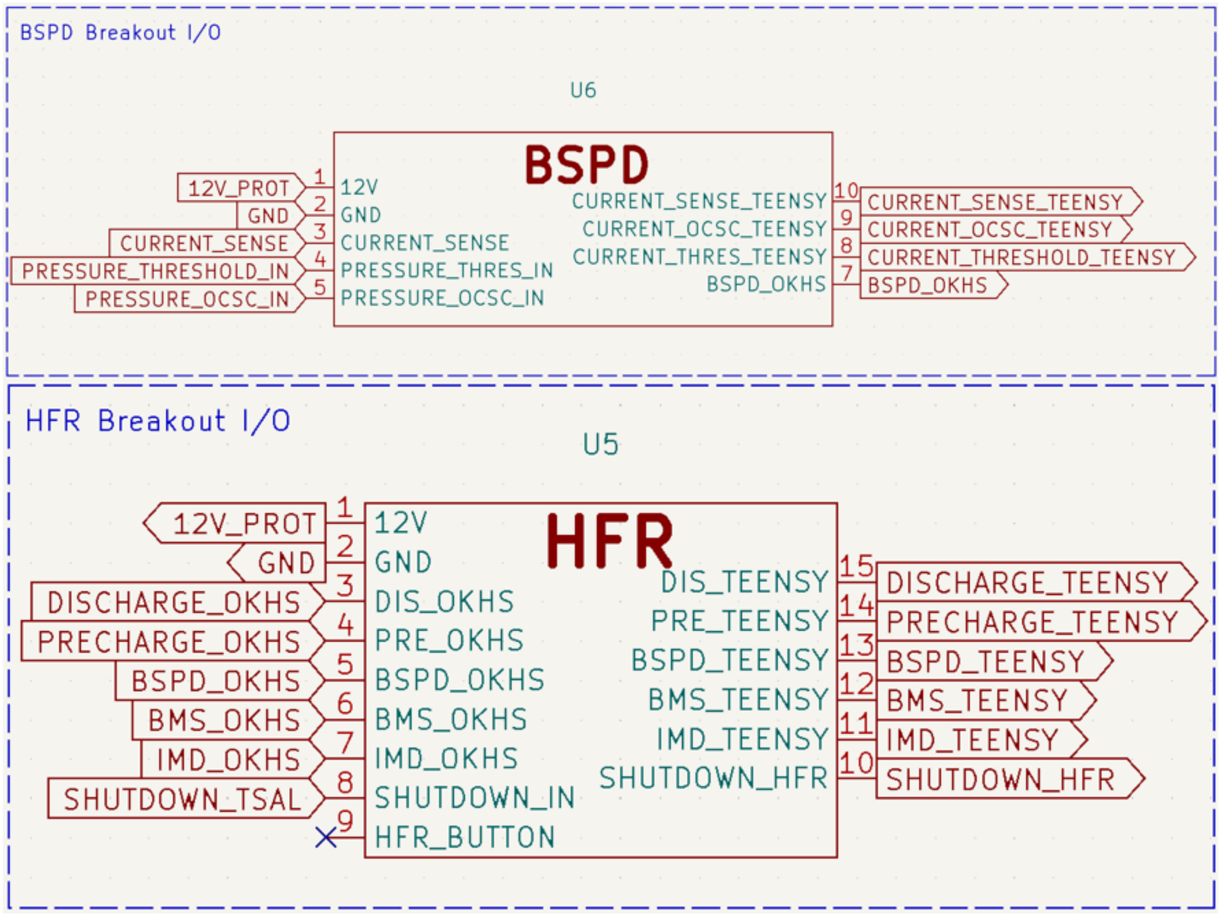


Figure 20 BSPD and HFR Interface to MOTHER board

PCB Layout Iterations

After the schematic was finished, the board layouts could be started. This went through many iterations as multiple meetings were had with Alexander Gregg and the Chief Engineer, Joshua Wenham, on where the PCB would be seated on NU24 and what shape would be most beneficial. The main two lessons learnt during this process was that the best design was one that had the littlest unused space and to not lay traces until the design is 100% decided on by all individuals that will interact with the board. If traces were laid before a final design was chosen, it introduced far more work, since the original traces laid would have needed to be removed and re-laid. This was the best choice since most often it takes more time to try and adapt the old trace layout to a new design.

The iterations of the MOTHER board PCB are depicted in the following figures: Figure 21, Figure 22, Figure 23, and Figure 24. Through the first two iterations, the traces were laid manually, but this ended up taking too much time to get a new layout each iteration. So instead, a KiCad plugin called Freerouting which allows different parameters to be adjusted to autoroute traces was used. While it did not give as satisfying of a result as manually laying the traces, it did speed up the process significantly.

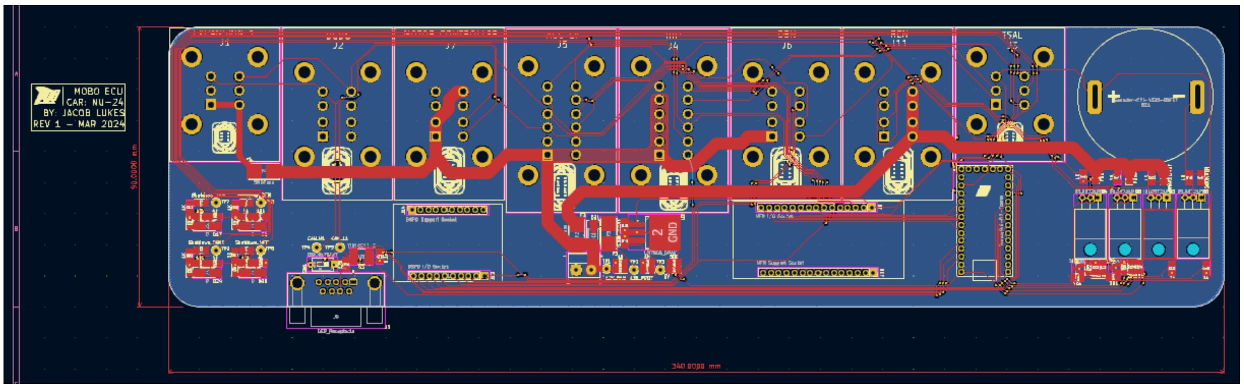


Figure 21 MOTHER board First Iteration

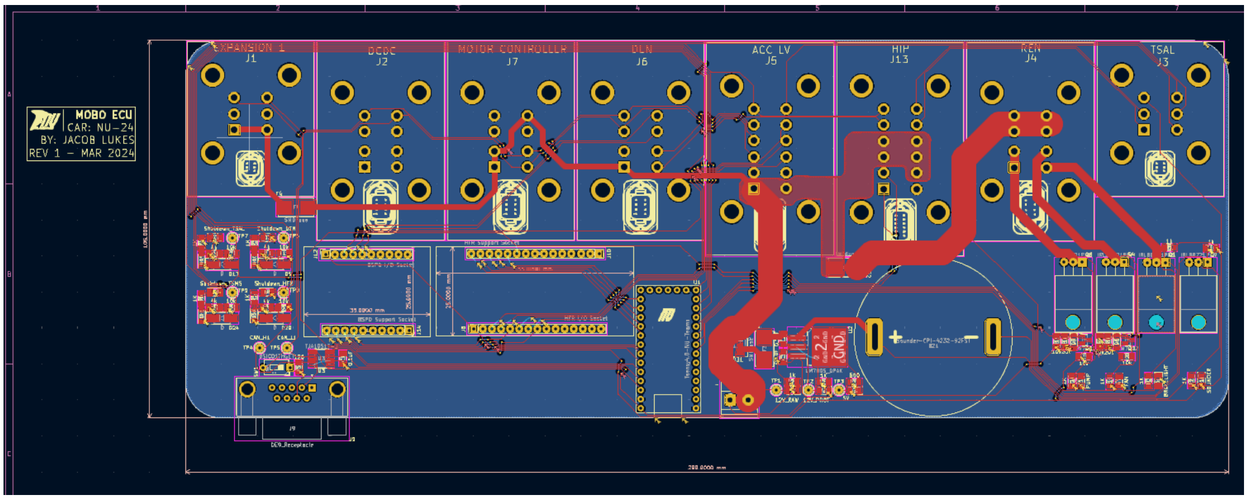


Figure 22 MOTHER board Second Iteration

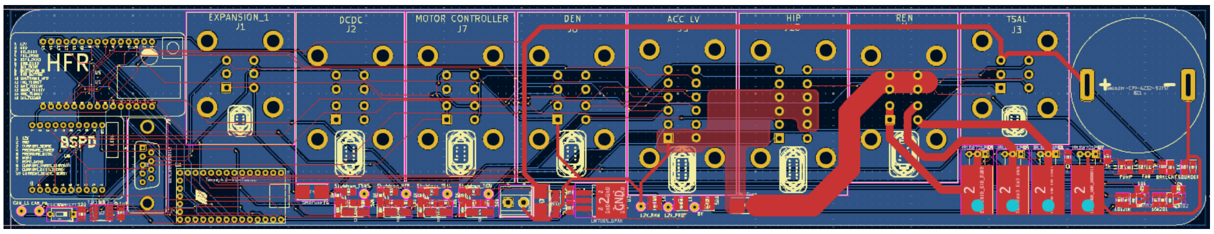


Figure 23 MOTHER board Third Iteration

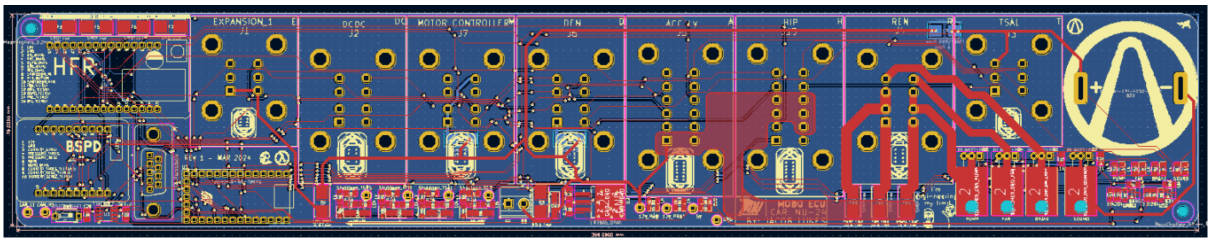


Figure 24 MOTHER board Fourth and Final Iteration

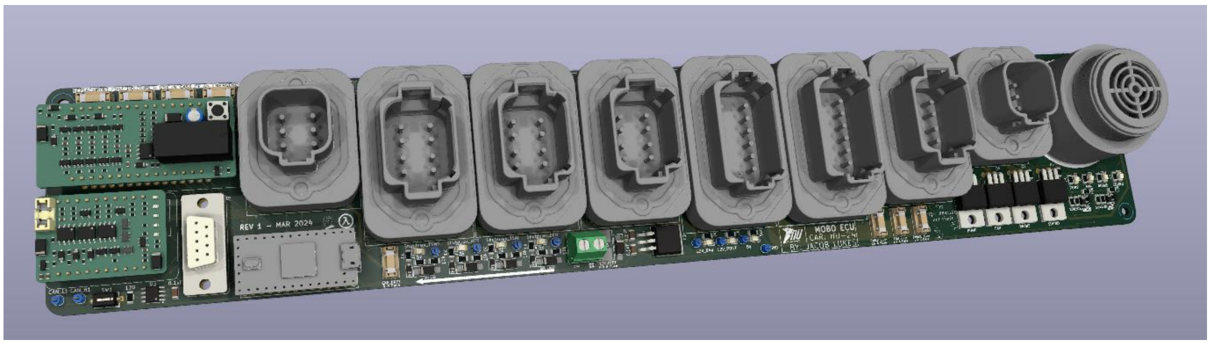


Figure 25 3D View of Final MOTHER board Iteration

Manufacturing and Commissioning

Once the final design for the PCB was decided upon, it was sent off to PCBGOGO to get manufactured. During this time the code was put together and a unit test plan was made.

Once the board arrived, the surface and through hole mount components were placed in accordance with the unit test plan shown in Figure 27. This allowed for each circuit to be tested individually to avoid putting it all together and turning it on, only to find out major issues and thus avoiding the process of manually error testing to find out what is wrong.

Lessons Learnt During Manufacturing

The main lesson that was learnt during soldering components was that the 2N7002 MOSFETs are very delicate and easy to break; extra caution should be taken when soldering and testing these MOSFETs.

Use of Stencil During Manufacturing

A stencil was used to lay solder paste onto the pads of all the surface mount components. This allowed for less time to be spent applying the paste to all the individual pads.

Mistake Made During Commissioning

During an integration test of the HIP and CEN, a misunderstanding occurred, resulting in the author suppling high voltage into the low voltage section of the HIP. This caused a 555 timer on the TSAL breakout board to blow, along with the teensy, HFR breakout board, BSPD breakout board and the main fuse on the MOTHER board. New versions of each of these components along with a new fuse was replaced. High voltage was input into the correct spot on the HIP board, and both boards functioned correctly.

Creation of Looms

Two looms were manufactured, one for the connection of the CEN and HIP and the other to connect the CEN to the brake light, radiator fan and pumps at the rear of the car. The second loom was greatly aided with the help of Aaron Gray. These looms were a bit daunting at first but turned out to be very straightforward to make. Learning experiences include:

- Twisting the two initial wires together and then concentrically twisting the remaining wires around the centre pair. Concentrically means to bring two wires together and wrap them around the central two, leaving a small gap in between each turn of the two wires brought together.
- Making sure to keep high tension on the wires during the entire process, to make sure they do not unravel themselves.
- Use a lubricate of sorts to make the process of adding heat shrink around the wires quick and easy.

Completing the BSPD Breakout Board Commissioning

Per the 2024 FSAE competition rules, the BSPD must trigger the shutdown circuit if 5 kW or more of power is drawn from the motor at any given time. To do this, the BSPD receives an analogue voltage signal from a current sensor. This signal represents the current that is supplied from the accumulator through the HVD. On NU24, this value is read from a bus bar that runs through the sensor. When the analogue voltage is input into the BSPD breakout board, it is compared to a reference voltage through a comparator, if it is below the reference voltage, the output from the comparator is high, if it is above the reference voltage, the output is low. On NU23, the reference voltage was calculated through a potentiometer, which acted as a voltage divider, with 5 V inputted and the reference voltage outputted, but this was considered to be susceptible to vibrations and had a large of a footprint. On the BSPD breakout board, the potentiometer was removed and replaced with two resistors to act as the voltage divider. When Jacob Bush commissioned the breakout boards, he did not have access to the current sensor, so the resistor values were not able to be determined.

To obtain the correct resistor values, a bench test would be conducted. This test would consist of the HIP and CEN connected through their joining loom. The current that would flow through the current sensor to trip the shutdown circuit at 5 kW can be found through the power equation given below. The value of the voltage is given by the nominal voltage level of the accumulator, on NU24 the maximum voltage is equal to 454 V, meaning that the nominal voltage is 400 V. Through using the equation, the current value at 5 kW and 400 V, is 12.5 A.

$$P = VI$$

The above-mentioned value of current can be simulated by winding a wire 10 times around the current sensor, this means that the value of current that is inputted will get multiplied by 10. Meaning that only 1.25 A of current will need to be inputted into the wire. To input this current into the wire, a device called an electronic load was used in series with a power supply. In this configuration, the electronic load can act as a variable resistor, setting the current draw of this 'resistor' to whatever is required (1.25 A in this instance). When this is done, the value of the current sensor signal voltage can be read. Caution should be taken to ensure that the voltage level found is the voltage that is input to the comparator, since this value is slightly different to the reading from the sensor output pin due to the gain from of the filter. On NU24, the gain was set to 100K/101K. The output voltage from the gain stage must be higher than the reference voltage (given through the voltage divider) that it is compared to. This will force the output from the comparator to go low, tripping the BSPD and shutting off HV.

After the commissioning of the MOTHER board, the resistor values that make the voltage divider were changed many times. The reason for so many changes was due to the continual discovery of different factors that affect what the reference voltage value should be. The main factor was a 16 mV leeway that the chosen comparators have between their two input voltages, meaning that the reference voltage has to be set to 16 mV lower than the desired output from gain stage of the current sensor signal. Other factors include the inaccuracies of the small components used and that the input to the voltage divider was not 5 V exactly, instead ~5.02 V. The correct resistors are not included as they have not been found yet.

Instead, an example has been included to show how the resistor values were found. The values given were found to be incorrect. They were also found with the assumption that the cut off current had to be 11 A instead of 12.5 A. Initially they were chosen to be the same, to get a reference voltage of 2.5 V. At the time, this value was questioned resulting in it being adjusted. This adjustment turned out to be incorrect, meaning the cut off power was 11.5 kW instead of the competition requirement of 5 kW. The value of the reference voltage was then later found to be 2.534 V, the resistors for the voltage divider for the current sensor reference voltage were set as r1 = 107 kΩ and r2 = 110 kΩ depicted in Figure 26. Giving a value of 2.53456 V and resulting in the cut off power being 4.78 kW.

The main lesson learnt during this process of choosing the resistor values was that the output voltage from the current sensor was very sensitive in this application. With an inspection of the datasheet, the graph supplied shows the output voltage changes with respect to the input current, the gradient is very steep at currents around 10 A, resulting in the requirement for the reference voltage to be very precise.

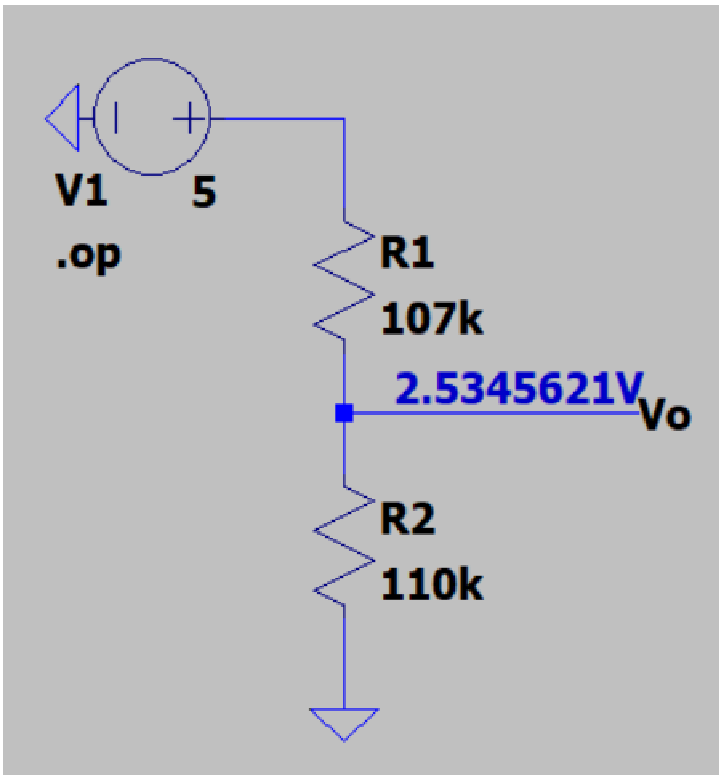


Figure 26 Current Sensor Reference Voltage

Code

The code that was written for the teensy on the MOTHER board was a combination of the code from NU23’s REN and NU23’s CEN. Some unimportant parts were taken out, such as large amounts of commented out code that was not being used. The file is called CEN_V9.

An integration of the new LV system on NU23 was organised, causing a new code file to be created called CEN_V9_INTEGRATION_TEST. This was done since most of the variables had to be changed to fit the EV3 DBC file, the DBC file that NU23 used.

Screenshots of the two code files can be found in the appendix section.

Track Day Testing

Integration of NU24’s LV Systems to NU23

The team got all of NU24’s LV systems working correctly together after a very long night. NU24’s LV systems were then strapped to the chassis of NU23 for an on-campus and Sydney Motor Sport Park (SMSP) track day. During the long night of integrating the new LV systems, it was found that the shutdown circuit lost too much voltage by the time it got to the accumulator to switch on HV. This was fixed with an addition of a component that was thought could be removed from NU23. The component was an interpose (a relay that boosts the shutdown circuit signal before it goes into the accumulator). This fixed the issue for the two testing days but would ultimately be fixed with a pre-charge PCB that would be integrated into the accumulator.

On Campus Testing

During the on-campus track day a couple of issues with the LV systems and a cooling pump arose, namely being that there were Zener diodes attached to the analogue signals coming from the pedal positions. This caused the signal to be inconsistent, which was fixed by removing these Zener diodes. Another issue that occurred was with the BSPD, it would sometimes cut out when the driver would come to a stop. In the data, it appeared as the current sense signal changing from its usual state, to peaking up to 200 A for a period of time before the car would turn off. At the time it was believed that the current sensor was malfunctioning. Another issue that occurred every time the car was turned on was that the HFR button on the HFR breakout board had to be pressed. At the time it was believed to be occurring due to a 2N7002 MOSFET being specified incorrectly for the expected current draw of the circuit. The circuit being the soft starter of the HFR. The cooling pump issue occurred due to one of the pumps not turning on, causing the motor to overheat during operation. This was fixed, and the temperatures fixed themselves.

Sydney Motor Sport Park Testing

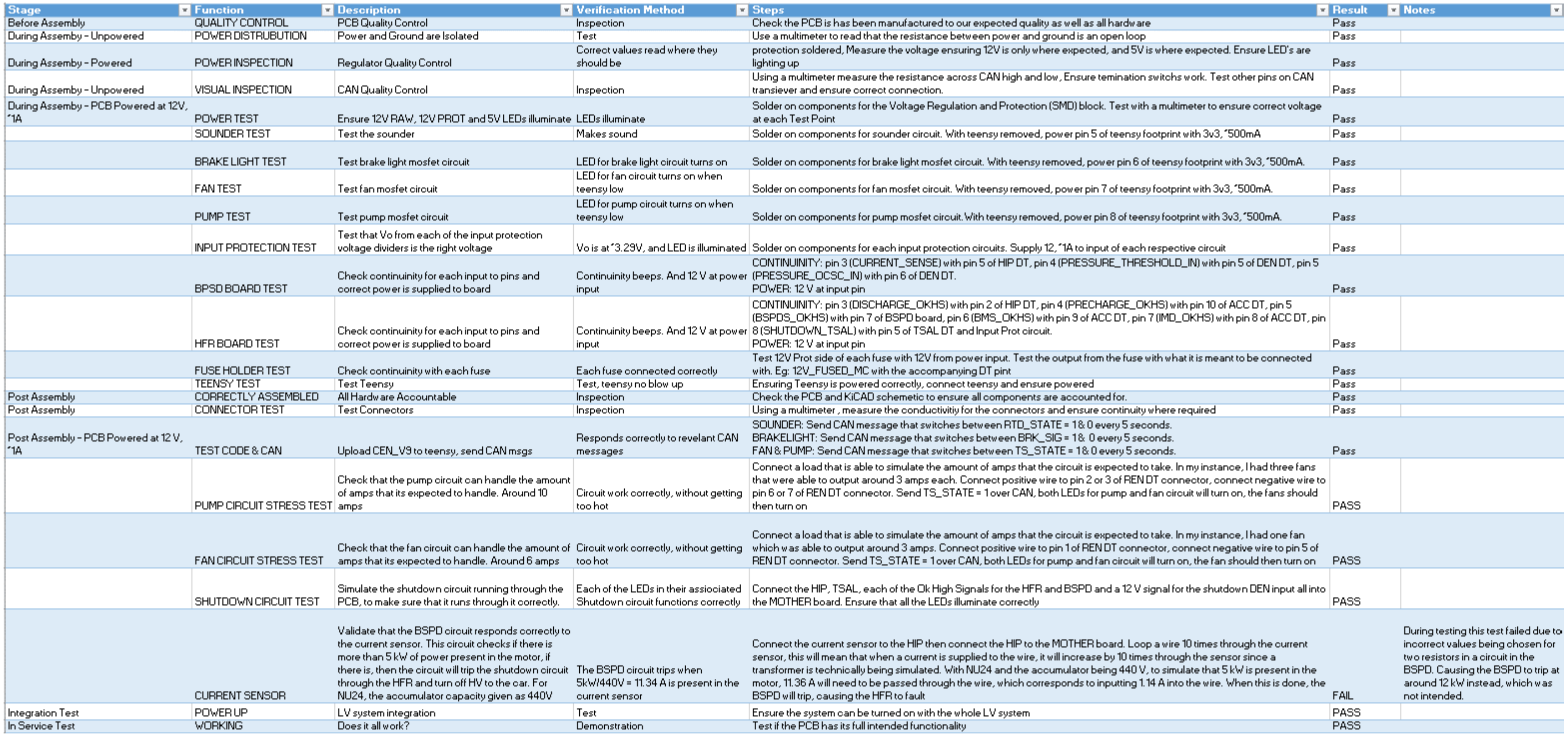
Next, NU23 fitted with NU24’s LV systems was taken to SMSP for some more testing. The BSPD was still having problems. The old current sensor from NU23’s LV system was swapped in, but it was having the same issue. The DBC file was changed to account for signed numbers, as it was thought that maybe the current sense signal value was going negative, this did not seem to have any effect. Next it was believed that the current sensor was being powered incorrectly, this would cause the signal to periodically spike. Testing was conducted by simulating the car running on the bench and wiggling around the current sensor wires to see if the issue could be repeated. Next it was believed that due to the current sense signal being an analogue signal, and the long distance that it covered, it would induce lots of noise and cause its voltage level to be inconsistent. The BSPD does contain a first order passive filter to minimise the noise, due to this, the author was unsure if the induced noise was a prominent issue.

The BSPD was mainly having issues in the first half of the day. Although, after lunch it seemed to hold up fine. Another test was taken place during the day, the test of Kieran Burgess’ new brake disks, which meant that the car would have to perform many large brake tests. These tests included the car reaching speeds over 70 km/h, and under these conditions the CEN functioned perfectly without any issues. This caused a bit of confusion since the BSPD did not trip once during these brake tests.

The temperature issue that occurred during the on-campus testing formulated again late in the afternoon. The same cooling pump was the root of the issue, though it was fixed and worked for the rest of the day. It was ultimately removed from the car and replaced with a new pump.

Unit Test Checklist

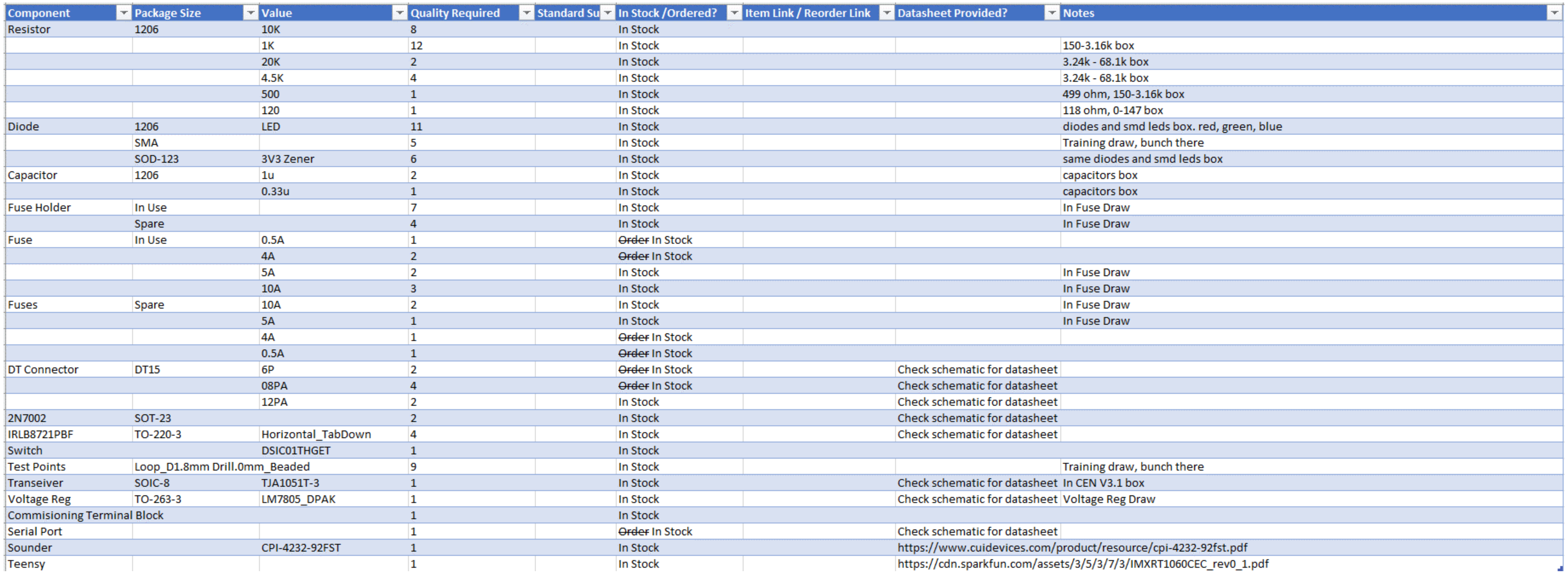
The checklist in Figure 27 can be found in the Github Repo under Racing-NU24 > CEN > PCB > MOTHER board > PCB_DDR_CEN_MOTHERBOARD_V2.3 > Unit Test Checklist



Stage	Function	Description	Verification Method	Steps	Result	Notes
Before Assembly	QUALITY CONTROL	PCB Quality Control	Inspection	Check the PCB is has been manufactured to our expected quality as well as all hardware	Pass	
During Assemby - Unpowered	POWER DISTRUBUTION	Power and Ground are Isolated	Test	Use a multimeter to read that the resistance between power and ground is an open loop	Pass	
During Assemby - Powered	POWER INSPECTION	Regulator Quality Control	Correct values read where they should be	protection soldered, Measure the voltage ensuring 12V is only where expected, and 5V is where expected. Ensure LED's are lighting up	Pass	
During Assemby - Unpowered	VISUAL INSPECTION	CAN Quality Control	Inspection	Using a multimeter measure the resistance across CAN high and low, Ensure temination switchs work. Test other pins on CAN transiever and ensure correct connection.	Pass	
During Assemby - PCB Powered at 12V, ~1A	POWER TEST	Ensure 12V RAW, 12V PROT and 5V LEDs illuminate	LEDs illuminate	Solder on components for the Voltage Regulation and Protection (SMD) block. Test with a multimeter to ensure correct voltage at each Test Point	Pass	
	SOUNDER TEST	Test the sounder	Makes sound	Solder on components for sounder circuit. With teensy removed, power pin 5 of teensy footprint with 3v3, ~500mA	Pass	
	BRAKE LIGHT TEST	Test brake light mosfet circuit	LED for brake light circuit turns on	Solder on components for brake light mosfet circuit. With teensy removed, power pin 6 of teensy footprint with 3v3, ~500mA.	Pass	
	FAN TEST	Test fan mosfet circuit	LED for fan circuit turns on when teensy low	Solder on components for fan mosfet circuit. With teensy removed, power pin 7 of teensy footprint with 3v3, ~500mA.	Pass	
	PUMP TEST	Test pump mosfet circuit	LED for pump circuit turns on when teensy low	Solder on components for pump mosfet circuit. With teensy removed, power pin 8 of teensy footprint with 3v3, ~500mA.	Pass	
	INPUT PROTECTION TEST	Test that Vo from each of the input protection voltage dividers is the right voltage	Vo is at ~3.29V, and LED is illuminated	Solder on components for each input protection circuits. Supply 12, ~1A to input of each respective circuit	Pass	
	BPSD BOARD TEST	Check continuinity for each input to pins and correct power is supplied to board	Continuinity beeps. And 12 V at power input	CONTINUINITY: pin 3 (CURRENT_SENSE) with pin 5 of HIP DT, pin 4 (PRESSURE_THRESHOLD_IN) with pin 5 of DEN DT, pin 5 (PRESSURE_OCSC_IN) with pin 6 of DEN DT. POWER: 12 V at input pin	Pass	
	HFR BOARD TEST	Check continuinity for each input to pins and correct power is supplied to board	Continuinity beeps. And 12 V at power input	CONTINUINITY: pin 3 (DISCHARGE_OKHS) with pin 2 of HIP DT, pin 4 (PRECHARGE_OKHS) with pin 10 of ACC DT, pin 5 (BSPDS_OKHS) with pin 7 of BSPD board, pin 6 (BMS_OKHS) with pin 9 of ACC DT, pin 7 (IMD_OKHS) with pin 8 of ACC DT, pin 8 (SHUTDOWN_TSAL) with pin 5 of TSAL DT and Input Prot circuit. POWER: 12 V at input pin	Pass	
	FUSE HOLDER TEST	Check continuinity with each fuse	Each fuse connected correctly	Test 12V Prot side of each fuse with 12V from power input. Test the output from the fuse with what it is meant to be connected with. Eg: 12V_FUSED_MC with the accompanying DT pint	Pass	
	TEENSY TEST	Test Teensy	Test, teensy no blow up	Ensuring Teensy is powered correctly, connect teensy and ensure powered	Pass	
Post Assembly	CORRECTLY ASSEMBLED	All Hardware Accountable	Inspection	Check the PCB and KiCAD schemetic to ensure all components are accounted for.	Pass	
Post Assembly	CONNECTOR TEST	Test Connectors	Inspection	Using a multimeter , measure the conductivitiy for the connectors and ensure continuity where required	Pass	
Post Assembly - PCB Powered at 12 V, ~1A	TEST CODE & CAN	Upload CEN_V9 to teensy, send CAN msgs	Responds correctly to revelant CAN messages	SOUNDER: Send CAN message that switches between RTD_STATE = 1& 0 every 5 seconds. BRAKELIGHT: Send CAN message that switches between BRK_SIG = 1& 0 every 5 seconds. FAN & PUMP: Send CAN message that switches between TS_STATE = 1& 0 every 5 seconds.	Pass	
	PUMP CIRCUIT STRESS TEST	Check that the pump circuit can handle the amount of amps that its expected to handle. Around 10 amps	Circuit work correctly, without getting too hot	Connect a load that is able to simulate the amount of amps that the circuit is expected to take. In my instance, I had three fans that were able to output around 3 amps each. Connect positive wire to pin 2 or 3 of REN DT connector, connect negative wire to pin 6 or 7 of REN DT connector. Send TS_STATE = 1 over CAN, both LEDs for pump and fan circuit will turn on, the fans should then turn on	PASS	
	FAN CIRCUIT STRESS TEST	Check that the fan circuit can handle the amount of amps that its expected to handle. Around 6 amps	Circuit work correctly, without getting too hot	Connect a load that is able to simulate the amount of amps that the circuit is expected to take. In my instance, I had one fan which was able to output around 3 amps. Connect positive wire to pin 1 of REN DT connector, connect negative wire to pin 5 of REN DT connector. Send TS_STATE = 1 over CAN, both LEDs for pump and fan circuit will turn on, the fan should then turn on	PASS	
	SHUTDOWN CIRCUIT TEST	Simulate the shutdown circuit running through the PCB, to make sure that it runs through it correctly.	Each of the LEDs in their assiociated Shutdown circuit functions correctly	Connect the HIP, TSAL, each of the Ok High Signals for the HFR and BSPD and a 12 V signal for the shutdown DEN input all into the MOTHER board. Ensure that all the LEDs illuminate correctly	PASS	
	CURRENT SENSOR	Validate that the BSPD circuit responds correctly to the current sensor. This circuit checks if there is more than 5 kW of power present in the motor, if there is, then the circuit will trip the shutdown circuit through the HFR and turn off HV to the car. For NU24, the accumulator capacity given as 440V	The BSPD circuit trips when 5kW/440V = 11.34 A is present in the current sensor	Connect the current sensor to the HIP then connect the HIP to the MOTHER board. Loop a wire 10 times through the current sensor, this will mean that when a current is supplied to the wire, it will increase by 10 times through the sensor since a transformer is technically being simulated. With NU24 and the accumulator being 440 V, to simulate that 5 kW is present in the motor, 11.36 A will need to be passed through the wire, which corresponds to inputting 1.14 A into the wire. When this is done, the BSPD will trip, causing the HFR to fault	FAIL	During testing this test failed due to incorrect values being chosen for two resistors in a circuit in the BSPD. Causing the BSPD to trip at around 12 kW instead, which was not intended.
Integration Test	POWER UP	LV system integration	Test	Ensure the system can be turned on with the whole LV system	PASS	
In Service Test	WORKING	Does it all work?	Demonstration	Test if the PCB has its full intended functionality	PASS	

Figure 27 Unit Test Check List

Bill of Materials



Component	Package Size	Value	Quality Required	Standard Su	In Stock /Ordered?	Item Link / Reorder Link	Datasheet Provided?	Notes
Resistor	1206	10K	8		In Stock			
		1K	12		In Stock			150-3.16k box
		20K	2		In Stock			3.24k - 68.1k box
		4.5K	4		In Stock			3.24k - 68.1k box
		500	1		In Stock			499 ohm, 150-3.16k box
		120	1		In Stock			118 ohm, 0-147 box
Diode	1206	LED	11		In Stock			diodes and smd leds box. red, green, blue
	SMA		5		In Stock			Training draw, bunch there
	SOD-123	3V3 Zener	6		In Stock			same diodes and smd leds box
Capacitor	1206	1u	2		In Stock			capacitors box
		0.33u	1		In Stock			capacitors box
Fuse Holder	In Use		7		In Stock			In Fuse Draw
	Spare		4		In Stock			In Fuse Draw
Fuse	In Use	0.5A	1		~~Order~~ In Stock			
		4A	2		~~Order~~ In Stock			
		5A	2		In Stock			In Fuse Draw
		10A	3		In Stock			In Fuse Draw
Fuses	Spare	10A	2		In Stock			In Fuse Draw
		5A	1		In Stock			In Fuse Draw
		4A	1		~~Order~~ In Stock			
		0.5A	1		~~Order~~ In Stock			
DT Connector	DT15	6P	2		~~Order~~ In Stock		Check schematic for datasheet	
		08PA	4		~~Order~~ In Stock		Check schematic for datasheet	
		12PA	2		In Stock		Check schematic for datasheet	
2N7002	SOT-23		2		In Stock		Check schematic for datasheet	
IRLB8721PBF	TO-220-3	Horizontal_TabDown	4		In Stock		Check schematic for datasheet	
Switch		DSIC01THGET	1		In Stock			
Test Points	Loop_D1.8mm Drill.0mm_Beaded		9		In Stock			Training draw, bunch there
Transeiver	SOIC-8	TJA1051T-3	1		In Stock		Check schematic for datasheet	In CEN V3.1 box
Voltage Reg	TO-263-3	LM7805_DPAK	1		In Stock		Check schematic for datasheet	Voltage Reg Draw
Commisioning Terminal Block			1		In Stock			
Serial Port			1		~~Order~~ In Stock		Check schematic for datasheet	
Sounder		CPI-4232-92FST	1		In Stock		https://www.cuidevices.com/product/resource/cpi-4232-92fst.pdf	
Teensy			1		In Stock		https://cdn.sparkfun.com/assets/3/5/3/7/3/IMXRT1060CEC_rev0_1.pdf	

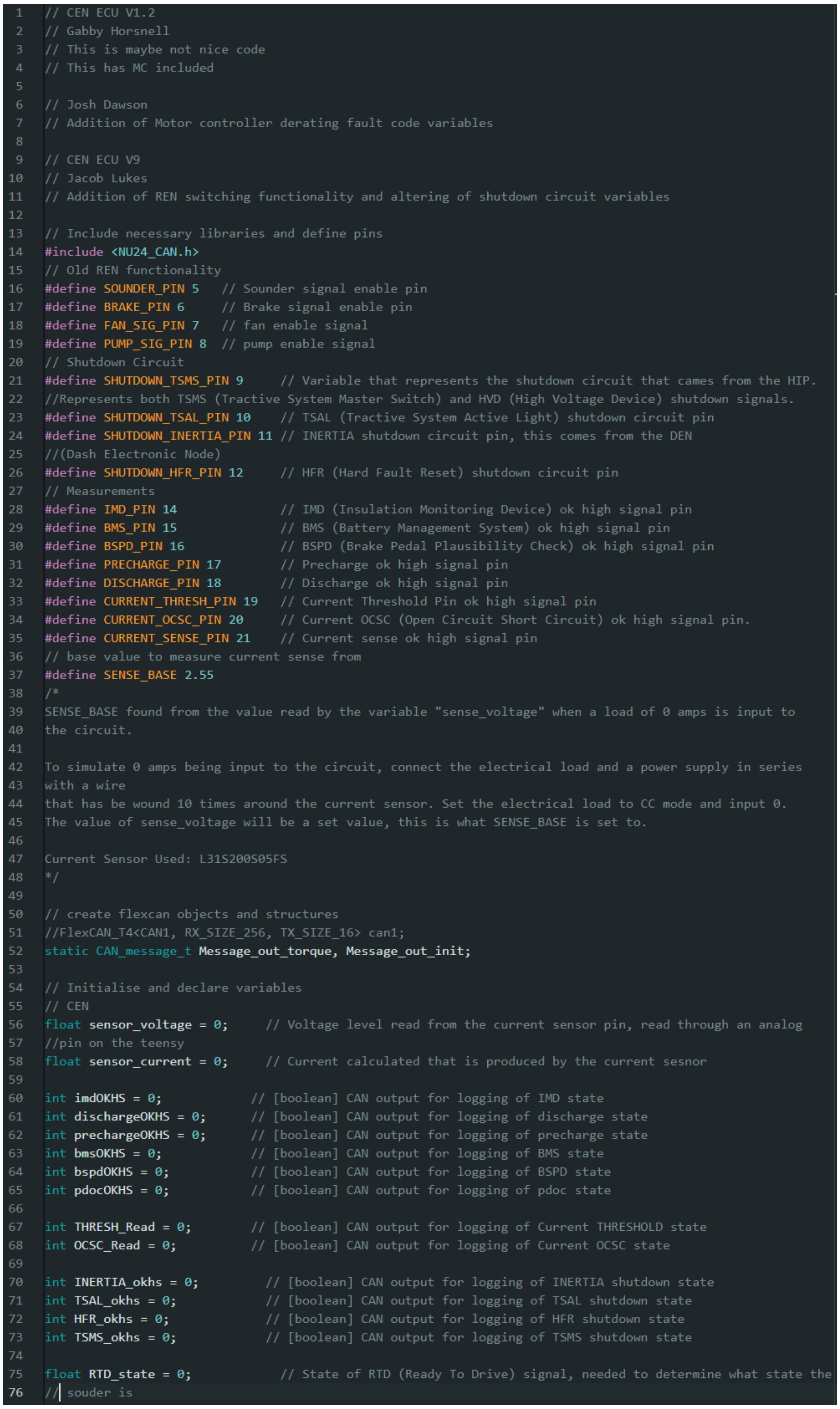
Conclusions and Recommendations

In conclusion, the new iteration of the CEN was a massive success. It elegantly integrated the systems of NU23's REN and the low voltage section NU23's CEN into one PCB. This allowed for the removal of a node from the car, but added an extra node (the HIP) for the high voltage circuitry from NU23's CEN. The weight of NU24's CEN, which includes the MOTHER board, BSPD and HFR breakout boards and the enclosure, weighed 0.75 kg. The modification of the MOSFET circuit from the REN was proved to be a massive success, as nowhere near as much heat was generated during operation. The addition of the extra fuses would help to reduce the possibility of the MOTHER board shorting from an external fault. The multiple iterations of the PCB taught a significant number of skills to the author, allowing the final PCB layout to have an efficient design and fit neatly on NU24's chassis. The field testing of the system was critical to understanding what faults occur during operation, allowing time to be spent effectively on improving the weaknesses of the PCB.

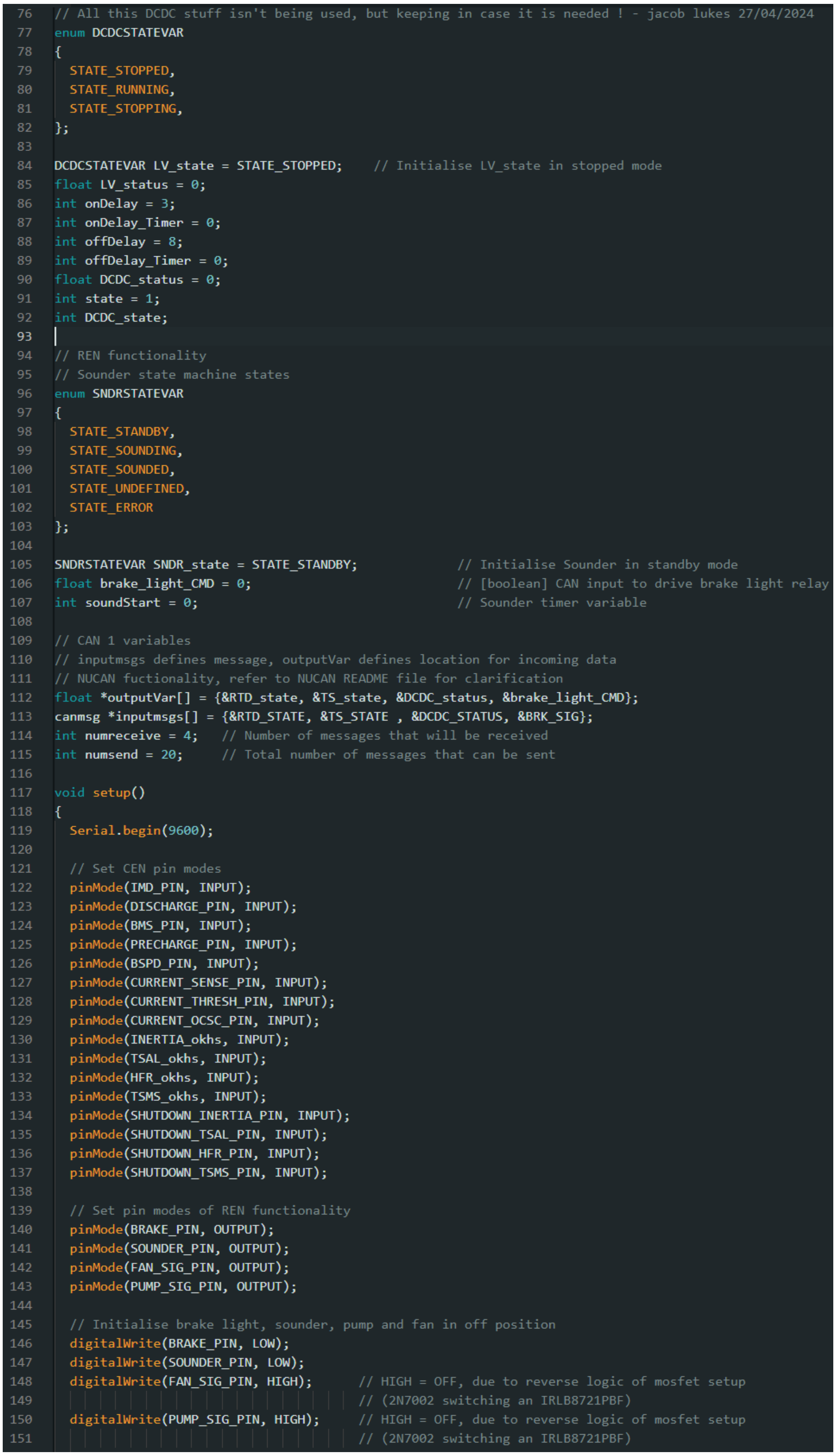
An iteration of the MOTHER board could include the interpose circuitry, allowing the pre-charge PCB to be smaller and not require the relay. In addition, more space could be left for the border around the breakout boards if their size needs to be increased in future iterations. Another improvement could be a temperature control circuit for the radiator fan, to reduce the time that the fan is on, diminishing the power used by the MOTHER board. The pump MOSFET circuit could have another IRLB8721PBF and 2N7002 circuit to allow for more current headroom for pump selection, useful if the pumps were required to be replaced by stronger ones.

Appendix

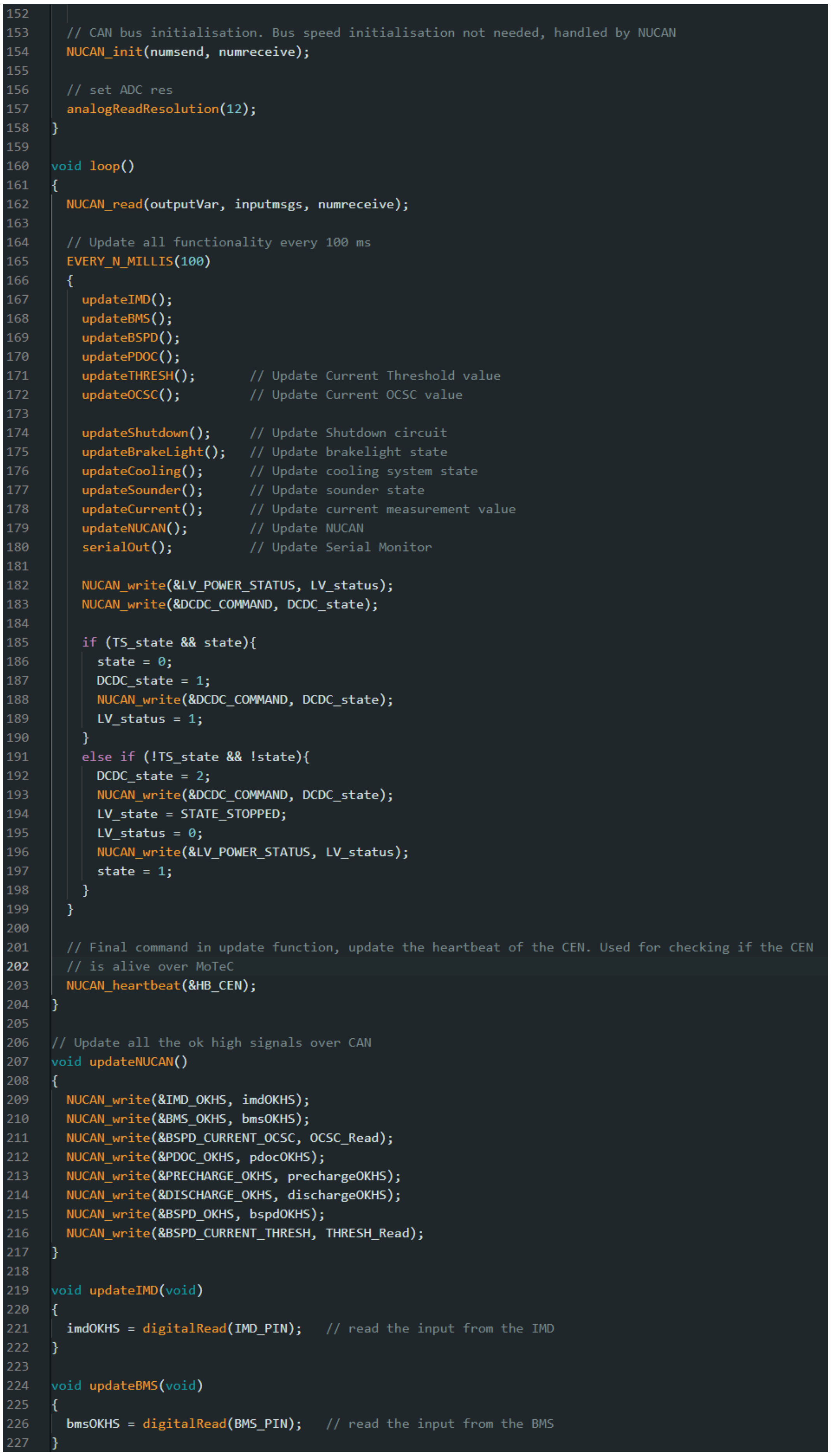
CEN_V9 code



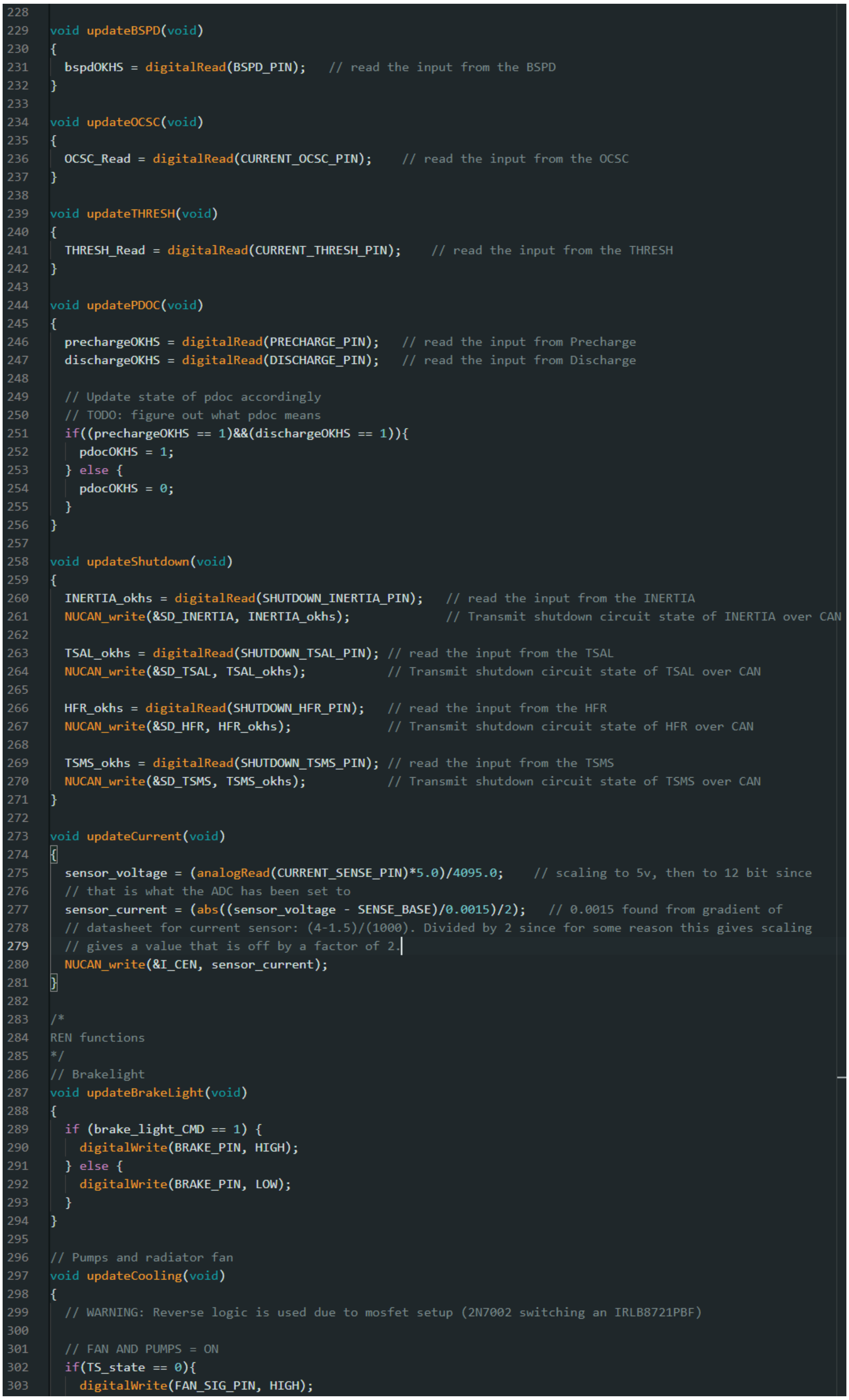
```
// CEN ECU V1.2
// Gabby Horsnell
// This is maybe not nice code
// This has MC included

// Josh Dawson
// Addition of Motor controller derating fault code variables

// CEN ECU V9
// Jacob Lukes
// Addition of REN switching functionality and altering of shutdown circuit variables

// Include necessary libraries and define pins
#include <NU24_CAN.h>
// Old REN functionality
#define SOUNDER_PIN 5   // Sounder signal enable pin
#define BRAKE_PIN 6     // Brake signal enable pin
#define FAN_SIG_PIN 7   // fan enable signal
#define PUMP_SIG_PIN 8  // pump enable signal
// Shutdown Circuit
#define SHUTDOWN_TSMS_PIN 9     // Variable that represents the shutdown circuit that cames from the HIP.
//Represents both TSMS (Tractive System Master Switch) and HVD (High Voltage Device) shutdown signals.
#define SHUTDOWN_TSAL_PIN 10    // TSAL (Tractive System Active Light) shutdown circuit pin
#define SHUTDOWN_INERTIA_PIN 11 // INERTIA shutdown circuit pin, this comes from the DEN
//(Dash Electronic Node)
#define SHUTDOWN_HFR_PIN 12     // HFR (Hard Fault Reset) shutdown circuit pin
// Measurements
#define IMD_PIN 14              // IMD (Insulation Monitoring Device) ok high signal pin
#define BMS_PIN 15              // BMS (Battery Management System) ok high signal pin
#define BSPD_PIN 16             // BSPD (Brake Pedal Plausibility Check) ok high signal pin
#define PRECHARGE_PIN 17        // Precharge ok high signal pin
#define DISCHARGE_PIN 18        // Discharge ok high signal pin
#define CURRENT_THRESH_PIN 19   // Current Threshold Pin ok high signal pin
#define CURRENT_OCSC_PIN 20     // Current OCSC (Open Circuit Short Circuit) ok high signal pin.
#define CURRENT_SENSE_PIN 21    // Current sense ok high signal pin
// base value to measure current sense from
#define SENSE_BASE 2.55
/*
SENSE_BASE found from the value read by the variable "sense_voltage" when a load of 0 amps is input to
the circuit.

To simulate 0 amps being input to the circuit, connect the electrical load and a power supply in series
with a wire
that has be wound 10 times around the current sensor. Set the electrical load to CC mode and input 0.
The value of sense_voltage will be a set value, this is what SENSE_BASE is set to.

Current Sensor Used: L31S200S05FS
*/

// create flexcan objects and structures
//FlexCAN_T4<CAN1, RX_SIZE_256, TX_SIZE_16> can1;
static CAN_message_t Message_out_torque, Message_out_init;

// Initialise and declare variables
// CEN
float sensor_voltage = 0;     // Voltage level read from the current sensor pin, read through an analog
//pin on the teensy
float sensor_current = 0;     // Current calculated that is produced by the current sesnor

int imdOKHS = 0;            // [boolean] CAN output for logging of IMD state
int dischargeOKHS = 0;      // [boolean] CAN output for logging of discharge state
int prechargeOKHS = 0;      // [boolean] CAN output for logging of precharge state
int bmsOKHS = 0;            // [boolean] CAN output for logging of BMS state
int bspdOKHS = 0;           // [boolean] CAN output for logging of BSPD state
int pdocOKHS = 0;           // [boolean] CAN output for logging of pdoc state

int THRESH_Read = 0;        // [boolean] CAN output for logging of Current THRESHOLD state
int OCSC_Read = 0;          // [boolean] CAN output for logging of Current OCSC state

int INERTIA_okhs = 0;         // [boolean] CAN output for logging of INERTIA shutdown state
int TSAL_okhs = 0;            // [boolean] CAN output for logging of TSAL shutdown state
int HFR_okhs = 0;             // [boolean] CAN output for logging of HFR shutdown state
int TSMS_okhs = 0;            // [boolean] CAN output for logging of TSMS shutdown state

float RTD_state = 0;            // State of RTD (Ready To Drive) signal, needed to determine what state the
// souder is
```



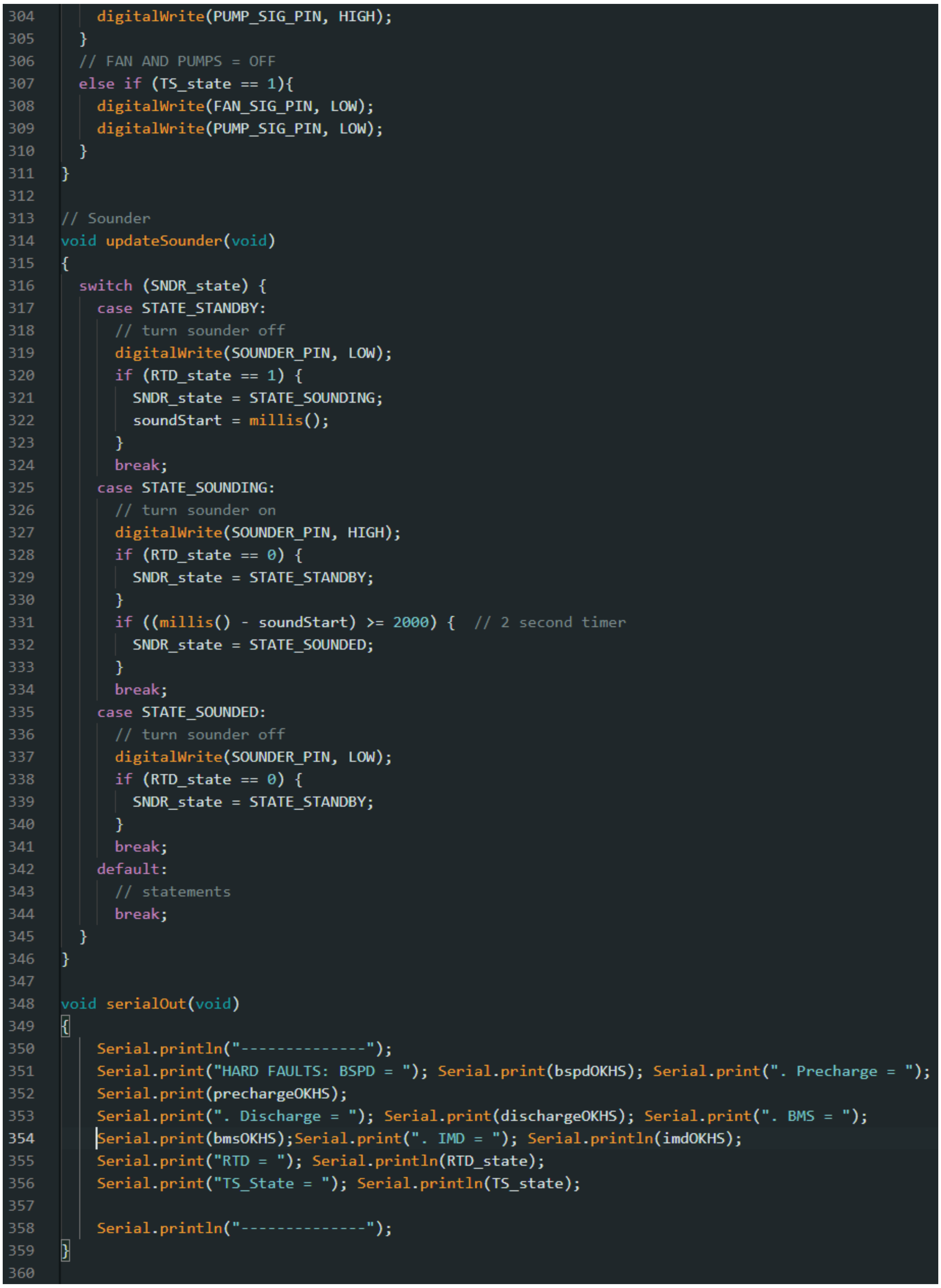
```
// All this DCDC stuff isn't being used, but keeping in case it is needed ! - jacob lukes 27/04/2024
enum DCDCSTATEVAR
{
  STATE_STOPPED,
  STATE_RUNNING,
  STATE_STOPPING,
};

DCDCSTATEVAR LV_state = STATE_STOPPED;    // Initialise LV_state in stopped mode
float LV_status = 0;
int onDelay = 3;
int onDelay_Timer = 0;
int offDelay = 8;
int offDelay_Timer = 0;
float DCDC_status = 0;
int state = 1;
int DCDC_state;

// REN functionality
// Sounder state machine states
enum SNDRSTATEVAR
{
  STATE_STANDBY,
  STATE_SOUNDING,
  STATE_SOUNDED,
  STATE_UNDEFINED,
  STATE_ERROR
};

SNDRSTATEVAR SNDR_state = STATE_STANDBY;             // Initialise Sounder in standby mode
float brake_light_CMD = 0;                           // [boolean] CAN input to drive brake light relay
int soundStart = 0;                                  // Sounder timer variable

// CAN 1 variables
// inputmsgs defines message, outputVar defines location for incoming data
// NUCAN fuctionality, refer to NUCAN README file for clarification
float *outputVar[] = {&RTD_state, &TS_state, &DCDC_status, &brake_light_CMD};
canmsg *inputmsgs[] = {&RTD_STATE, &TS_STATE , &DCDC_STATUS, &BRK_SIG};
int numreceive = 4;   // Number of messages that will be received
int numsend = 20;     // Total number of messages that can be sent

void setup()
{
  Serial.begin(9600);

  // Set CEN pin modes
  pinMode(IMD_PIN, INPUT);
  pinMode(DISCHARGE_PIN, INPUT);
  pinMode(BMS_PIN, INPUT);
  pinMode(PRECHARGE_PIN, INPUT);
  pinMode(BSPD_PIN, INPUT);
  pinMode(CURRENT_SENSE_PIN, INPUT);
  pinMode(CURRENT_THRESH_PIN, INPUT);
  pinMode(CURRENT_OCSC_PIN, INPUT);
  pinMode(INERTIA_okhs, INPUT);
  pinMode(TSAL_okhs, INPUT);
  pinMode(HFR_okhs, INPUT);
  pinMode(TSMS_okhs, INPUT);
  pinMode(SHUTDOWN_INERTIA_PIN, INPUT);
  pinMode(SHUTDOWN_TSAL_PIN, INPUT);
  pinMode(SHUTDOWN_HFR_PIN, INPUT);
  pinMode(SHUTDOWN_TSMS_PIN, INPUT);

  // Set pin modes of REN functionality
  pinMode(BRAKE_PIN, OUTPUT);
  pinMode(SOUNDER_PIN, OUTPUT);
  pinMode(FAN_SIG_PIN, OUTPUT);
  pinMode(PUMP_SIG_PIN, OUTPUT);

  // Initialise brake light, sounder, pump and fan in off position
  digitalWrite(BRAKE_PIN, LOW);
  digitalWrite(SOUNDER_PIN, LOW);
  digitalWrite(FAN_SIG_PIN, HIGH);      // HIGH = OFF, due to reverse logic of mosfet setup
                                        // (2N7002 switching an IRLB8721PBF)
  digitalWrite(PUMP_SIG_PIN, HIGH);     // HIGH = OFF, due to reverse logic of mosfet setup
                                        // (2N7002 switching an IRLB8721PBF)
```



```
  
  // CAN bus initialisation. Bus speed initialisation not needed, handled by NUCAN
  NUCAN_init(numsend, numreceive);

  // set ADC res
  analogReadResolution(12);
}

void loop()
{
  NUCAN_read(outputVar, inputmsgs, numreceive);

  // Update all functionality every 100 ms
  EVERY_N_MILLIS(100)
  {
    updateIMD();
    updateBMS();
    updateBSPD();
    updatePDOC();
    updateTHRESH();       // Update Current Threshold value
    updateOCSC();         // Update Current OCSC value

    updateShutdown();     // Update Shutdown circuit
    updateBrakeLight();   // Update brakelight state
    updateCooling();      // Update cooling system state
    updateSounder();      // Update sounder state
    updateCurrent();      // Update current measurement value
    updateNUCAN();        // Update NUCAN
    serialOut();          // Update Serial Monitor

    NUCAN_write(&LV_POWER_STATUS, LV_status);
    NUCAN_write(&DCDC_COMMAND, DCDC_state);

    if (TS_state && state){
      state = 0;
      DCDC_state = 1;
      NUCAN_write(&DCDC_COMMAND, DCDC_state);
      LV_status = 1;
    }
    else if (!TS_state && !state){
      DCDC_state = 2;
      NUCAN_write(&DCDC_COMMAND, DCDC_state);
      LV_state = STATE_STOPPED;
      LV_status = 0;
      NUCAN_write(&LV_POWER_STATUS, LV_status);
      state = 1;
    }
  }

  // Final command in update function, update the heartbeat of the CEN. Used for checking if the CEN
  // is alive over MoTeC
  NUCAN_heartbeat(&HB_CEN);
}

// Update all the ok high signals over CAN
void updateNUCAN()
{
  NUCAN_write(&IMD_OKHS, imdOKHS);
  NUCAN_write(&BMS_OKHS, bmsOKHS);
  NUCAN_write(&BSPD_CURRENT_OCSC, OCSC_Read);
  NUCAN_write(&PDOC_OKHS, pdocOKHS);
  NUCAN_write(&PRECHARGE_OKHS, prechargeOKHS);
  NUCAN_write(&DISCHARGE_OKHS, dischargeOKHS);
  NUCAN_write(&BSPD_OKHS, bspdOKHS);
  NUCAN_write(&BSPD_CURRENT_THRESH, THRESH_Read);
}

void updateIMD(void)
{
  imdOKHS = digitalRead(IMD_PIN);   // read the input from the IMD
}

void updateBMS(void)
{
  bmsOKHS = digitalRead(BMS_PIN);   // read the input from the BMS
}
```



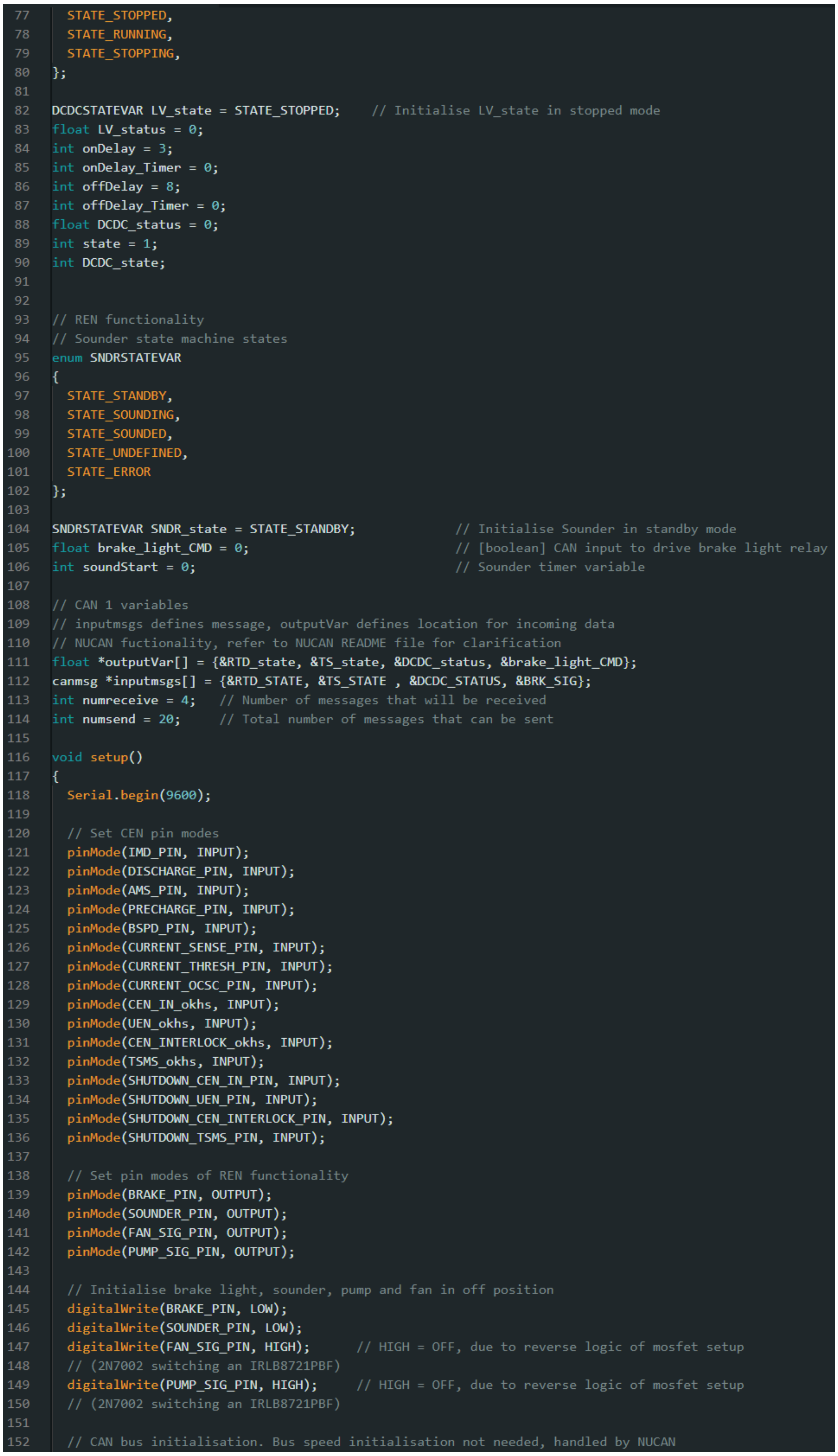
```
void updateBSPD(void)
{
  bspdOKHS = digitalRead(BSPD_PIN);  // read the input from the BSPD
}

void updateOCSC(void)
{
  OCSC_Read = digitalRead(CURRENT_OCSC_PIN);    // read the input from the OCSC
}

void updateTHRESH(void)
{
  THRESH_Read = digitalRead(CURRENT_THRESH_PIN);    // read the input from the THRESH
}

void updatePDOC(void)
{
  prechargeOKHS = digitalRead(PRECHARGE_PIN);   // read the input from Precharge
  dischargeOKHS = digitalRead(DISCHARGE_PIN);   // read the input from Discharge

  // Update state of pdoc accordingly
  // TODO: figure out what pdoc means
  if((prechargeOKHS == 1)&&(dischargeOKHS == 1)){
    pdocOKHS = 1;
  } else {
    pdocOKHS = 0;
  }
}

void updateShutdown(void)
{
  INERTIA_okhs = digitalRead(SHUTDOWN_INERTIA_PIN);   // read the input from the INERTIA
  NUCAN_write(&SD_INERTIA, INERTIA_okhs);             // Transmit shutdown circuit state of INERTIA over CAN

  TSAL_okhs = digitalRead(SHUTDOWN_TSAL_PIN); // read the input from the TSAL
  NUCAN_write(&SD_TSAL, TSAL_okhs);           // Transmit shutdown circuit state of TSAL over CAN

  HFR_okhs = digitalRead(SHUTDOWN_HFR_PIN);   // read the input from the HFR
  NUCAN_write(&SD_HFR, HFR_okhs);             // Transmit shutdown circuit state of HFR over CAN

  TSMS_okhs = digitalRead(SHUTDOWN_TSMS_PIN); // read the input from the TSMS
  NUCAN_write(&SD_TSMS, TSMS_okhs);           // Transmit shutdown circuit state of TSMS over CAN
}

void updateCurrent(void)
{
  sensor_voltage = (analogRead(CURRENT_SENSE_PIN)*5.0)/4095.0;    // scaling to 5v, then to 12 bit since
  // that is what the ADC has been set to
  sensor_current = (abs((sensor_voltage - SENSE_BASE)/0.0015)/2);   // 0.0015 found from gradient of
  // datasheet for current sensor: (4-1.5)/(1000). Divided by 2 since for some reason this gives scaling
  // gives a value that is off by a factor of 2.
  NUCAN_write(&I_CEN, sensor_current);
}

/*
REN functions
*/
// Brakelight
void updateBrakeLight(void)
{
  if (brake_light_CMD == 1) {
    digitalWrite(BRAKE_PIN, HIGH);
  } else {
    digitalWrite(BRAKE_PIN, LOW);
  }
}

// Pumps and radiator fan
void updateCooling(void)
{
  // WARNING: Reverse logic is used due to mosfet setup (2N7002 switching an IRLB8721PBF)

  // FAN AND PUMPS = ON
  if(TS_state == 0){
    digitalWrite(FAN_SIG_PIN, HIGH);
```



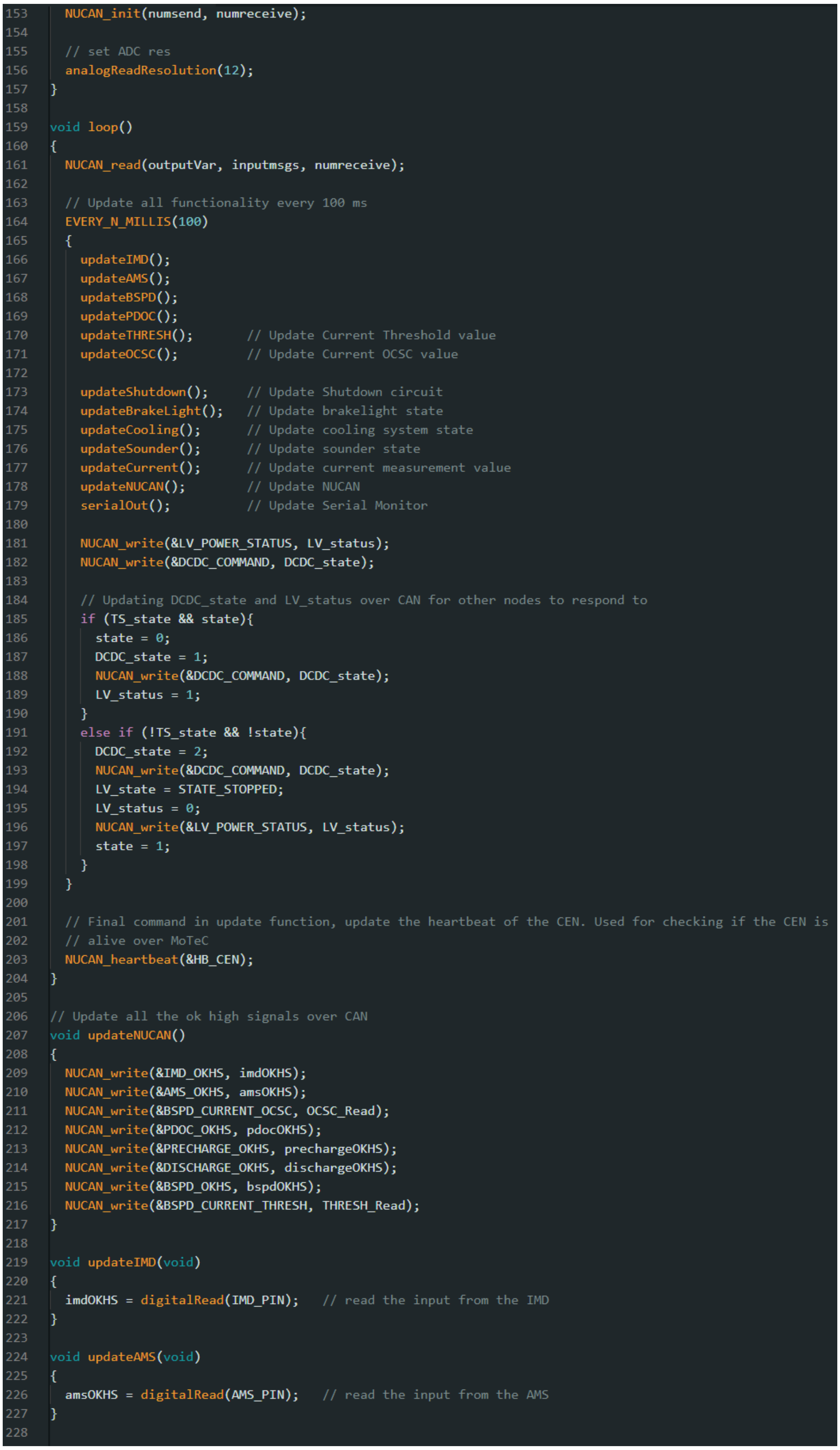
```
    digitalWrite(PUMP_SIG_PIN, HIGH);
  }
  // FAN AND PUMPS = OFF
  else if (TS_state == 1){
    digitalWrite(FAN_SIG_PIN, LOW);
    digitalWrite(PUMP_SIG_PIN, LOW);
  }
}

// Sounder
void updateSounder(void)
{
  switch (SNDR_state) {
    case STATE_STANDBY:
      // turn sounder off
      digitalWrite(SOUNDER_PIN, LOW);
      if (RTD_state == 1) {
        SNDR_state = STATE_SOUNDING;
        soundStart = millis();
      }
      break;
    case STATE_SOUNDING:
      // turn sounder on
      digitalWrite(SOUNDER_PIN, HIGH);
      if (RTD_state == 0) {
        SNDR_state = STATE_STANDBY;
      }
      if ((millis() - soundStart) >= 2000) {  // 2 second timer
        SNDR_state = STATE_SOUNDED;
      }
      break;
    case STATE_SOUNDED:
      // turn sounder off
      digitalWrite(SOUNDER_PIN, LOW);
      if (RTD_state == 0) {
        SNDR_state = STATE_STANDBY;
      }
      break;
    default:
      // statements
      break;
  }
}

void serialOut(void)
{
    Serial.println("-------------");
    Serial.print("HARD FAULTS: BSPD = "); Serial.print(bspdOKHS); Serial.print(". Precharge = ");
    Serial.print(prechargeOKHS);
    Serial.print(". Discharge = "); Serial.print(dischargeOKHS); Serial.print(". BMS = ");
    Serial.print(bmsOKHS);Serial.print(". IMD = "); Serial.println(imdOKHS);
    Serial.print("RTD = "); Serial.println(RTD_state);
    Serial.print("TS_State = "); Serial.println(TS_state);

    Serial.println("-------------");
}
```

CEN_V9_INTEGRATION_TEST Code

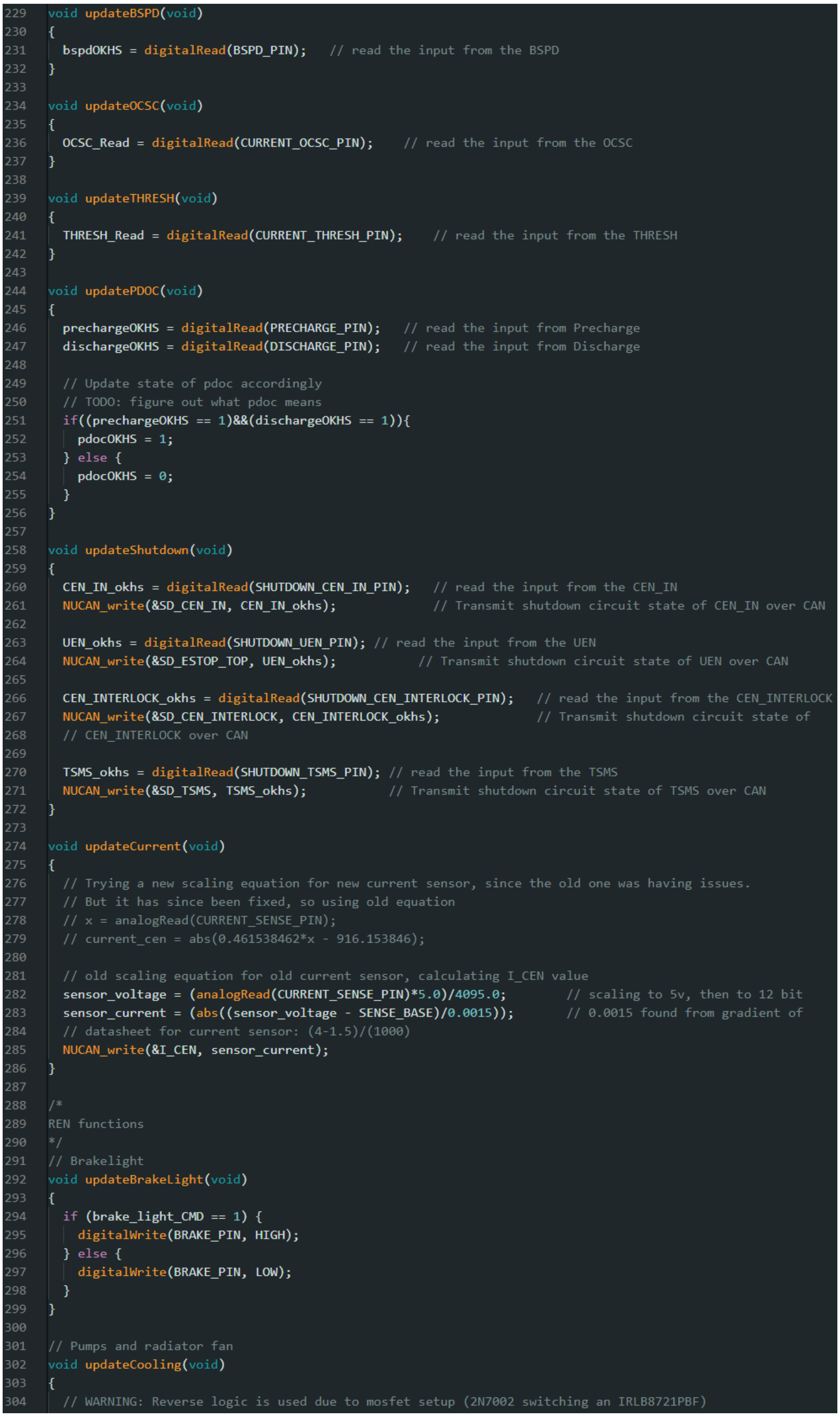
```
// CEN ECU V1.2
// Gabby Horsnell
// This is maybe not nice code
// This has MC included

// Josh Dawson
// Addition of Motor controller derating fault code variables

// CEN ECU V9
// Jacob Lukes
// Addition of REN switching functionality and altering of shutdown circuit variables

// Include necessary libraries and define pins
#include <EV3_CAN.h>
// Old REN functionality
#define SOUNDER_PIN 5   // Sounder signal enable pin
#define BRAKE_PIN 6     // Brake signal enable pin
#define FAN_SIG_PIN 7   // fan enable signal
#define PUMP_SIG_PIN 8  // pump enable signal
// Shutdown Circuit
#define SHUTDOWN_TSMS_PIN 9     // Variable that represents the shutdown circuit that cames from the HIP.
// Represents both TSMS (Tractive System Master Switch) and HVD (High Voltage Device) shutdown signals.
#define SHUTDOWN_UEN_PIN 10    // UEN (Tractive System Active Light) shutdown circuit pin
#define SHUTDOWN_CEN_IN_PIN 11 // CEN_IN shutdown circuit pin, this comes from the DEN (Dash Electronic Node)
#define SHUTDOWN_CEN_INTERLOCK_PIN 12     // CEN_INTERLOCK (Hard Fault Reset) shutdown circuit pin
// Measurements
#define IMD_PIN 14              // IMD (Insulation Monitoring Device) ok high signal pin
#define AMS_PIN 15              // AMS (Battery Management System) ok high signal pin
#define BSPD_PIN 16             // BSPD (Brake Pedal Plausibility Check) ok high signal pin
#define PRECHARGE_PIN 17        // Precharge ok high signal pin
#define DISCHARGE_PIN 18        // Discharge ok high signal pin
#define CURRENT_THRESH_PIN 19   // Current Threshold Pin ok high signal pin
#define CURRENT_OCSC_PIN 20     // Current OCSC (Open Circuit Short Circuit) ok high signal pin.
#define CURRENT_SENSE_PIN 21    // Current sense ok high signal pin
// base value to measure current sense from
#define SENSE_BASE 2.44  // bspd resistor values = 15k, 16k -> v_divider = 16k/(15k+16k).
/*
also changing from sense base from 2.515 to 2.44, to match the datasheet for the current sensor's
graph scaling value
Current Sensor Used: L31S200S05FS
*/

// create flexcan objects and structures
//FlexCAN_T4<CAN1, RX_SIZE_256, TX_SIZE_16> can1;
static CAN_message_t Message_out_torque, Message_out_init;

// Initialise and declare variables
// CEN
float sensor_voltage = 0;     // Voltage level read from the current sensor pin, read through an analog
// pin on the teensy
float sensor_current = 0;     // Current calculated that is produced by the current sesnor

int imdOKHS = 0;            // [boolean] CAN output for logging of IMD state
int dischargeOKHS = 0;      // [boolean] CAN output for logging of discharge state
int prechargeOKHS = 0;      // [boolean] CAN output for logging of precharge state
int amsOKHS = 0;            // [boolean] CAN output for logging of AMS state
int bspdOKHS = 0;           // [boolean] CAN output for logging of BSPD state
int pdocOKHS = 0;           // [boolean] CAN output for logging of pdoc state

int THRESH_Read = 0;        // [boolean] CAN output for logging of Current THRESHOLD state
int OCSC_Read = 0;          // [boolean] CAN output for logging of Current OCSC state

int CEN_IN_okhs = 0;         // [boolean] CAN output for logging of CEN_IN shutdown state
int UEN_okhs = 0;            // [boolean] CAN output for logging of UEN shutdown state
int CEN_INTERLOCK_okhs = 0;              // [boolean] CAN output for logging of CEN_INTERLOCK shutdown state
int TSMS_okhs = 0;            // [boolean] CAN output for logging of TSMS shutdown state

float RTD_state = 0;            // State of RTD (Ready To Drive) signal, needed to determine what state the
// souder is
float TS_state = 0;             // State of the LV Power System (needed for PUMP and FAN)

// DCDC
// Count for DCDC Disabled to stop fans and associated equipment
// All this DCDC stuff isn't being used, but keeping in case it is needed ! - jacob lukes 27/04/2024
enum DCDCSTATEVAR
{
```



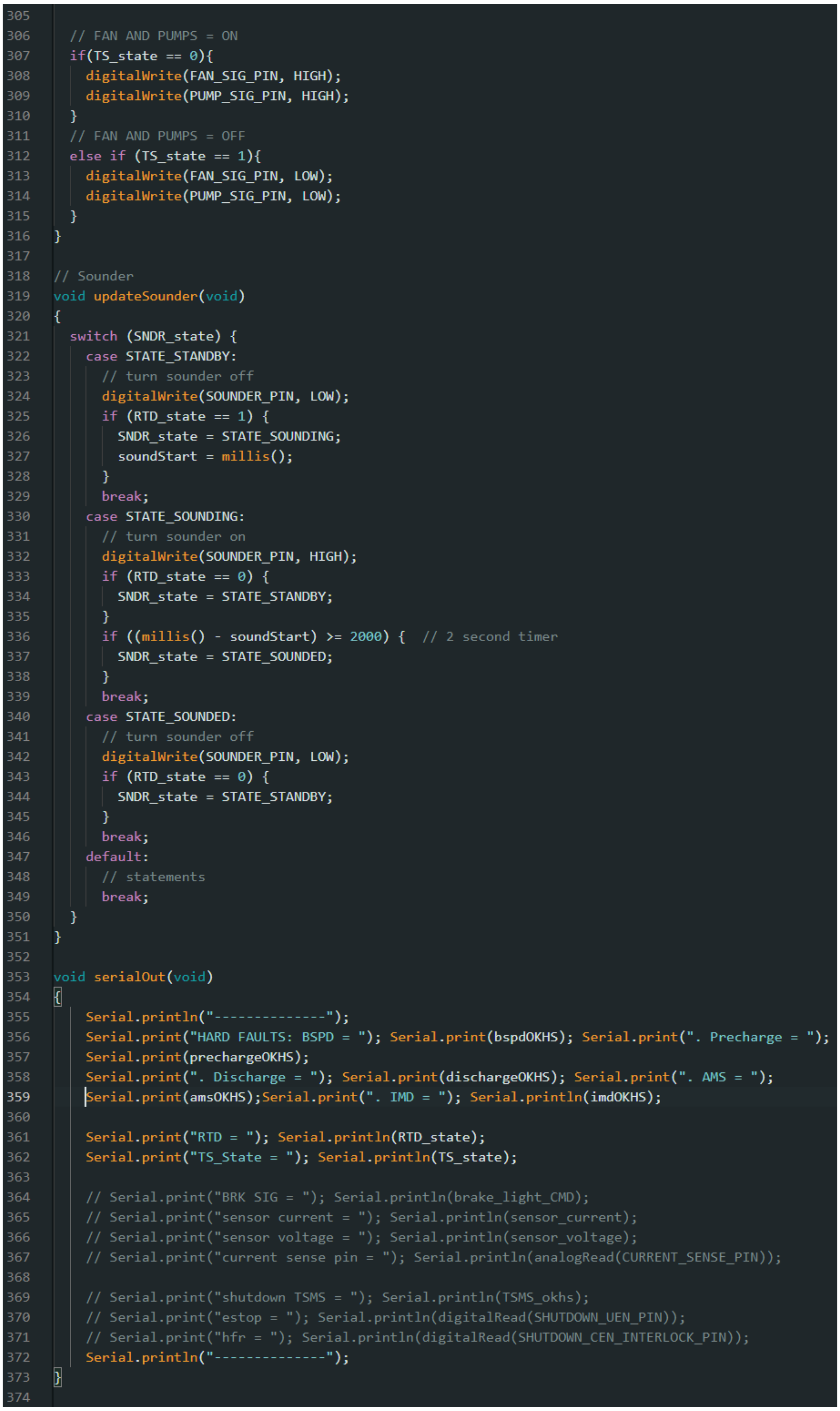
```
  STATE_STOPPED,
  STATE_RUNNING,
  STATE_STOPPING,
};

DCDCSTATEVAR LV_state = STATE_STOPPED;    // Initialise LV_state in stopped mode
float LV_status = 0;
int onDelay = 3;
int onDelay_Timer = 0;
int offDelay = 8;
int offDelay_Timer = 0;
float DCDC_status = 0;
int state = 1;
int DCDC_state;


// REN functionality
// Sounder state machine states
enum SNDRSTATEVAR
{
  STATE_STANDBY,
  STATE_SOUNDING,
  STATE_SOUNDED,
  STATE_UNDEFINED,
  STATE_ERROR
};

SNDRSTATEVAR SNDR_state = STATE_STANDBY;             // Initialise Sounder in standby mode
float brake_light_CMD = 0;                           // [boolean] CAN input to drive brake light relay
int soundStart = 0;                                  // Sounder timer variable

// CAN 1 variables
// inputmsgs defines message, outputVar defines location for incoming data
// NUCAN fuctionality, refer to NUCAN README file for clarification
float *outputVar[] = {&RTD_state, &TS_state, &DCDC_status, &brake_light_CMD};
canmsg *inputmsgs[] = {&RTD_STATE, &TS_STATE , &DCDC_STATUS, &BRK_SIG};
int numreceive = 4;   // Number of messages that will be received
int numsend = 20;     // Total number of messages that can be sent

void setup()
{
  Serial.begin(9600);

  // Set CEN pin modes
  pinMode(IMD_PIN, INPUT);
  pinMode(DISCHARGE_PIN, INPUT);
  pinMode(AMS_PIN, INPUT);
  pinMode(PRECHARGE_PIN, INPUT);
  pinMode(BSPD_PIN, INPUT);
  pinMode(CURRENT_SENSE_PIN, INPUT);
  pinMode(CURRENT_THRESH_PIN, INPUT);
  pinMode(CURRENT_OCSC_PIN, INPUT);
  pinMode(CEN_IN_okhs, INPUT);
  pinMode(UEN_okhs, INPUT);
  pinMode(CEN_INTERLOCK_okhs, INPUT);
  pinMode(TSMS_okhs, INPUT);
  pinMode(SHUTDOWN_CEN_IN_PIN, INPUT);
  pinMode(SHUTDOWN_UEN_PIN, INPUT);
  pinMode(SHUTDOWN_CEN_INTERLOCK_PIN, INPUT);
  pinMode(SHUTDOWN_TSMS_PIN, INPUT);

  // Set pin modes of REN functionality
  pinMode(BRAKE_PIN, OUTPUT);
  pinMode(SOUNDER_PIN, OUTPUT);
  pinMode(FAN_SIG_PIN, OUTPUT);
  pinMode(PUMP_SIG_PIN, OUTPUT);

  // Initialise brake light, sounder, pump and fan in off position
  digitalWrite(BRAKE_PIN, LOW);
  digitalWrite(SOUNDER_PIN, LOW);
  digitalWrite(FAN_SIG_PIN, HIGH);      // HIGH = OFF, due to reverse logic of mosfet setup
  // (2N7002 switching an IRLB8721PBF)
  digitalWrite(PUMP_SIG_PIN, HIGH);     // HIGH = OFF, due to reverse logic of mosfet setup
  // (2N7002 switching an IRLB8721PBF)

  // CAN bus initialisation. Bus speed initialisation not needed, handled by NUCAN
```



```
  NUCAN_init(numsend, numreceive);

  // set ADC res
  analogReadResolution(12);
}

void loop()
{
  NUCAN_read(outputVar, inputmsgs, numreceive);

  // Update all functionality every 100 ms
  EVERY_N_MILLIS(100)
  {
    updateIMD();
    updateAMS();
    updateBSPD();
    updatePDOC();
    updateTHRESH();       // Update Current Threshold value
    updateOCSC();         // Update Current OCSC value

    updateShutdown();     // Update Shutdown circuit
    updateBrakeLight();   // Update brakelight state
    updateCooling();      // Update cooling system state
    updateSounder();      // Update sounder state
    updateCurrent();      // Update current measurement value
    updateNUCAN();        // Update NUCAN
    serialOut();          // Update Serial Monitor

    NUCAN_write(&LV_POWER_STATUS, LV_status);
    NUCAN_write(&DCDC_COMMAND, DCDC_state);

    // Updating DCDC_state and LV_status over CAN for other nodes to respond to
    if (TS_state && state){
      state = 0;
      DCDC_state = 1;
      NUCAN_write(&DCDC_COMMAND, DCDC_state);
      LV_status = 1;
    }
    else if (!TS_state && !state){
      DCDC_state = 2;
      NUCAN_write(&DCDC_COMMAND, DCDC_state);
      LV_state = STATE_STOPPED;
      LV_status = 0;
      NUCAN_write(&LV_POWER_STATUS, LV_status);
      state = 1;
    }
  }

  // Final command in update function, update the heartbeat of the CEN. Used for checking if the CEN is
  // alive over MoTeC
  NUCAN_heartbeat(&HB_CEN);
}

// Update all the ok high signals over CAN
void updateNUCAN()
{
  NUCAN_write(&IMD_OKHS, imdOKHS);
  NUCAN_write(&AMS_OKHS, amsOKHS);
  NUCAN_write(&BSPD_CURRENT_OCSC, OCSC_Read);
  NUCAN_write(&PDOC_OKHS, pdocOKHS);
  NUCAN_write(&PRECHARGE_OKHS, prechargeOKHS);
  NUCAN_write(&DISCHARGE_OKHS, dischargeOKHS);
  NUCAN_write(&BSPD_OKHS, bspdOKHS);
  NUCAN_write(&BSPD_CURRENT_THRESH, THRESH_Read);
}

void updateIMD(void)
{
  imdOKHS = digitalRead(IMD_PIN);   // read the input from the IMD
}

void updateAMS(void)
{
  amsOKHS = digitalRead(AMS_PIN);   // read the input from the AMS
}

```



```
void updateBSPD(void)
{
  bspdOKHS = digitalRead(BSPD_PIN);   // read the input from the BSPD
}

void updateOCSC(void)
{
  OCSC_Read = digitalRead(CURRENT_OCSC_PIN);    // read the input from the OCSC
}

void updateTHRESH(void)
{
  THRESH_Read = digitalRead(CURRENT_THRESH_PIN);    // read the input from the THRESH
}

void updatePDOC(void)
{
  prechargeOKHS = digitalRead(PRECHARGE_PIN);   // read the input from Precharge
  dischargeOKHS = digitalRead(DISCHARGE_PIN);   // read the input from Discharge

  // Update state of pdoc accordingly
  // TODO: figure out what pdoc means
  if((prechargeOKHS == 1)&&(dischargeOKHS == 1)){
    pdocOKHS = 1;
  } else {
    pdocOKHS = 0;
  }
}

void updateShutdown(void)
{
  CEN_IN_okhs = digitalRead(SHUTDOWN_CEN_IN_PIN);   // read the input from the CEN_IN
  NUCAN_write(&SD_CEN_IN, CEN_IN_okhs);             // Transmit shutdown circuit state of CEN_IN over CAN

  UEN_okhs = digitalRead(SHUTDOWN_UEN_PIN); // read the input from the UEN
  NUCAN_write(&SD_ESTOP_TOP, UEN_okhs);           // Transmit shutdown circuit state of UEN over CAN

  CEN_INTERLOCK_okhs = digitalRead(SHUTDOWN_CEN_INTERLOCK_PIN);   // read the input from the CEN_INTERLOCK
  NUCAN_write(&SD_CEN_INTERLOCK, CEN_INTERLOCK_okhs);             // Transmit shutdown circuit state of
  // CEN_INTERLOCK over CAN

  TSMS_okhs = digitalRead(SHUTDOWN_TSMS_PIN); // read the input from the TSMS
  NUCAN_write(&SD_TSMS, TSMS_okhs);           // Transmit shutdown circuit state of TSMS over CAN
}

void updateCurrent(void)
{
  // Trying a new scaling equation for new current sensor, since the old one was having issues.
  // But it has since been fixed, so using old equation
  // x = analogRead(CURRENT_SENSE_PIN);
  // current_cen = abs(0.461538462*x - 916.153846);

  // old scaling equation for old current sensor, calculating I_CEN value
  sensor_voltage = (analogRead(CURRENT_SENSE_PIN)*5.0)/4095.0;        // scaling to 5v, then to 12 bit
  sensor_current = (abs((sensor_voltage - SENSE_BASE)/0.0015));       // 0.0015 found from gradient of
  // datasheet for current sensor: (4-1.5)/(1000)
  NUCAN_write(&I_CEN, sensor_current);
}

/*
REN functions
*/
// Brakelight
void updateBrakeLight(void)
{
  if (brake_light_CMD == 1) {
    digitalWrite(BRAKE_PIN, HIGH);
  } else {
    digitalWrite(BRAKE_PIN, LOW);
  }
}

// Pumps and radiator fan
void updateCooling(void)
{
  // WARNING: Reverse logic is used due to mosfet setup (2N7002 switching an IRLB8721PBF)
```



```

  // FAN AND PUMPS = ON
  if(TS_state == 0){
    digitalWrite(FAN_SIG_PIN, HIGH);
    digitalWrite(PUMP_SIG_PIN, HIGH);
  }
  // FAN AND PUMPS = OFF
  else if (TS_state == 1){
    digitalWrite(FAN_SIG_PIN, LOW);
    digitalWrite(PUMP_SIG_PIN, LOW);
  }
}

// Sounder
void updateSounder(void)
{
  switch (SNDR_state) {
    case STATE_STANDBY:
      // turn sounder off
      digitalWrite(SOUNDER_PIN, LOW);
      if (RTD_state == 1) {
        SNDR_state = STATE_SOUNDING;
        soundStart = millis();
      }
      break;
    case STATE_SOUNDING:
      // turn sounder on
      digitalWrite(SOUNDER_PIN, HIGH);
      if (RTD_state == 0) {
        SNDR_state = STATE_STANDBY;
      }
      if ((millis() - soundStart) >= 2000) {  // 2 second timer
        SNDR_state = STATE_SOUNDED;
      }
      break;
    case STATE_SOUNDED:
      // turn sounder off
      digitalWrite(SOUNDER_PIN, LOW);
      if (RTD_state == 0) {
        SNDR_state = STATE_STANDBY;
      }
      break;
    default:
      // statements
      break;
  }
}

void serialOut(void)
{
    Serial.println("-------------");
    Serial.print("HARD FAULTS: BSPD = "); Serial.print(bspdOKHS); Serial.print(". Precharge = ");
    Serial.print(prechargeOKHS);
    Serial.print(". Discharge = "); Serial.print(dischargeOKHS); Serial.print(". AMS = ");
    Serial.print(amsOKHS);Serial.print(". IMD = "); Serial.println(imdOKHS);

    Serial.print("RTD = "); Serial.println(RTD_state);
    Serial.print("TS_State = "); Serial.println(TS_state);

    // Serial.print("BRK SIG = "); Serial.println(brake_light_CMD);
    // Serial.print("sensor current = "); Serial.println(sensor_current);
    // Serial.print("sensor voltage = "); Serial.println(sensor_voltage);
    // Serial.print("current sense pin = "); Serial.println(analogRead(CURRENT_SENSE_PIN));

    // Serial.print("shutdown TSMS = "); Serial.println(TSMS_okhs);
    // Serial.print("estop = "); Serial.println(digitalRead(SHUTDOWN_UEN_PIN));
    // Serial.print("hfr = "); Serial.println(digitalRead(SHUTDOWN_CEN_INTERLOCK_PIN));
    Serial.println("-------------");
}

```