EV Battery Pack Challenge

Electrical and Computer Engineering

Team: Geoffrey Chavez, Aden Lee, Marlen Rojas-Reyes, Genesis Torres-Romero and Gustavo Magana

Advisor: Dr. Curtis Wang

Client: Battery Workforce Challenge

Charging Eagles

CAL STATE LA

Agenda

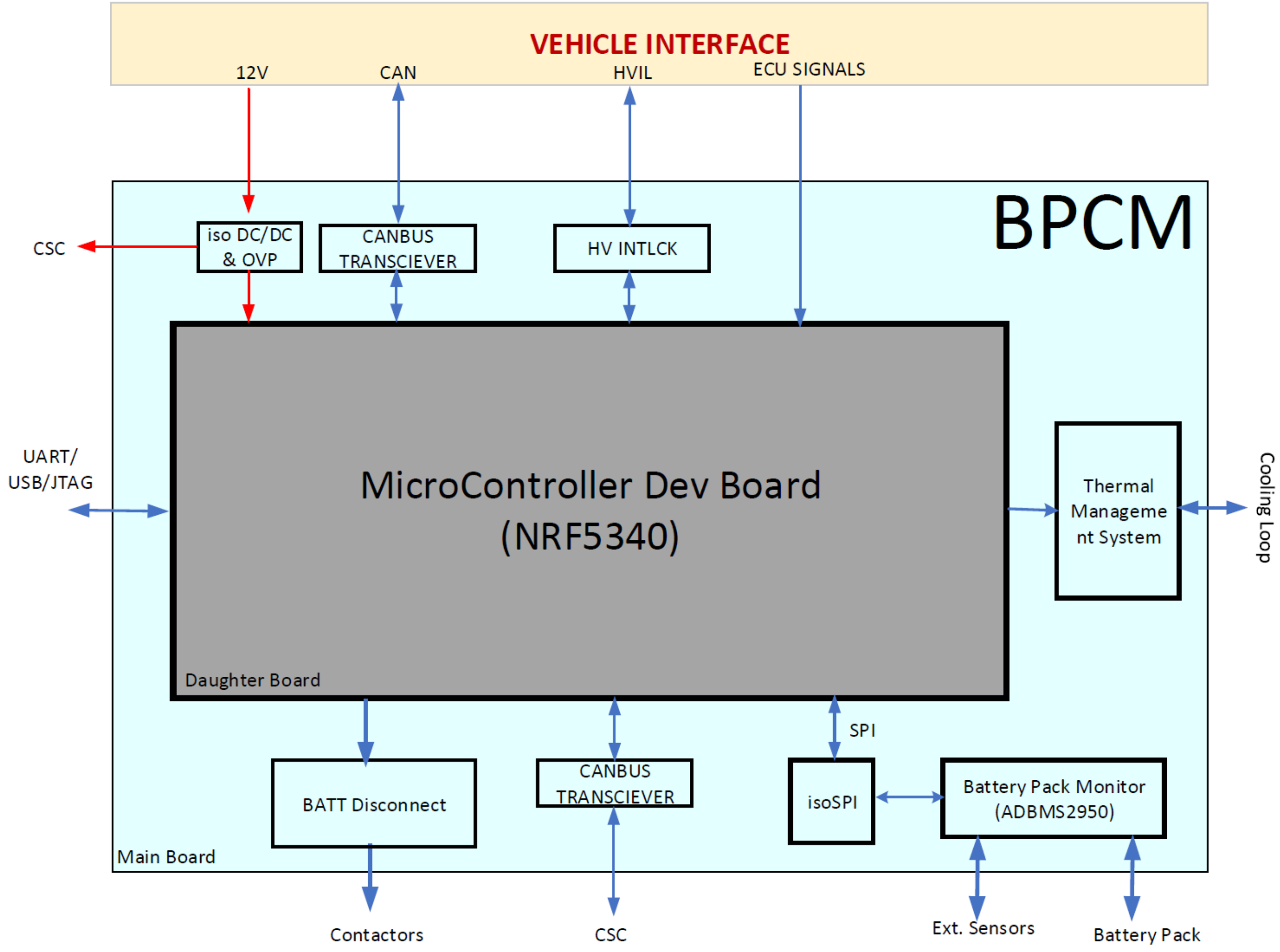
- **Project Background**
- Project Organization
- Designs Overview
  - Design Failure Mode and Effects Analysis (DFMEA)
  - Cell Sensing Circuit
  - Battery Disconnect Unit
  - 7-Segment Display and Thermistor
  - HVIL/LOI Protypes
  - PCB Design
- Conclusion

Project Background

The Battery Workforce Challenge collegiate competition is a three-year engineering competition that challenges North American universities and their community college partners to design, build, test and integrate an advanced EV battery pack into a Stellantis vehicle.

BATTERY WORKFORCE CHALLENGE

Project Background



Project Objective

The objective of the EV Battery Pack Challenge aims to prompt the Battery Pack Module (BPCM) to facilitate the battery's state of charge, adequate cell balancing, and the development of an all-inclusive monitoring system for the overall health of the Charging Eagle's battery pack. Secondly, it focuses on designing and developing the HVIL (High Voltage Interlock Loop) and Loss of Isolation Detection systems, ensuring compliance with the specific requirements outlined by the competition organizers.

Agenda

- Project Background
- **Project Organization**
- Designs Overview
    - Design Failure Mode and Effects Analysis (DFMEA)
    - Cell Sensing Circuit
    - Battery Disconnect Unit
    - 7-Segment Display and Thermistor
    - HVIL/LOI Protypes
    - PCB Design
- Conclusion

Team Picture

Team Breakdown

Genesis & Geoffrey (State of Charge & Cell Balancing)

Role: Battery Technology Team

Marlen & Gustavo (High Voltage Interlock & Loss of Isolation)

Role: High Voltage Systems Team

Aden (PCB Design & Build)

Role: PCB Design and Integration Specialist

No.	Requirement	Objective
1	DFMEA	• Identify required functions of the concept designs and potential failure modes within their concept design
2	CSC	• Using ADI chipset to develop a battery function (balancing or voltage monitoring • Demonstrated with the use of 12V assembly or 12V power supply/resistor board
3	Design BDU	• Demonstrate Power-up and Power-down sequence of vehicle • Create hardware and be able to turn items on/off through codes and sending messages from laptop
4	7- Segment Display /Thermistor	• Using a STM board and an nRF52 successfully control a 7-segment display and read data from a thermistor over SPI/CAN communication.
5	HVIL/LOI	• Develop a Schematic and functional prototype of the two subsystems
6	PCBs	• Design a CAN Bus and a PCB that can connect to M450/ECU microcontrollers

Agenda

- Project Background
- Project Organization
- **Designs Overview**
  - Design Failure Mode and Effects Analysis (DFMEA)
  - Cell Sensing Circuit
  - Battery Disconnect Unit
  - 7-Segment Display and Thermistor
  - HVIL/LOI Protypes
  - PCB Design
- Conclusion

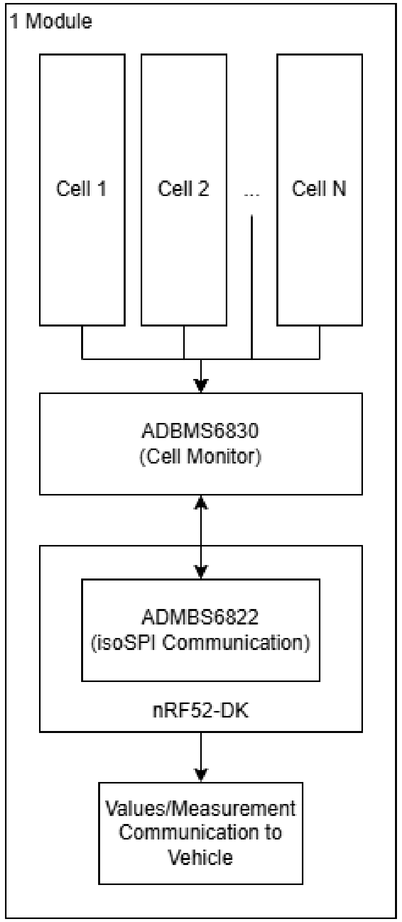
Design Failure Mode and Effects Analysis (DFMEA)

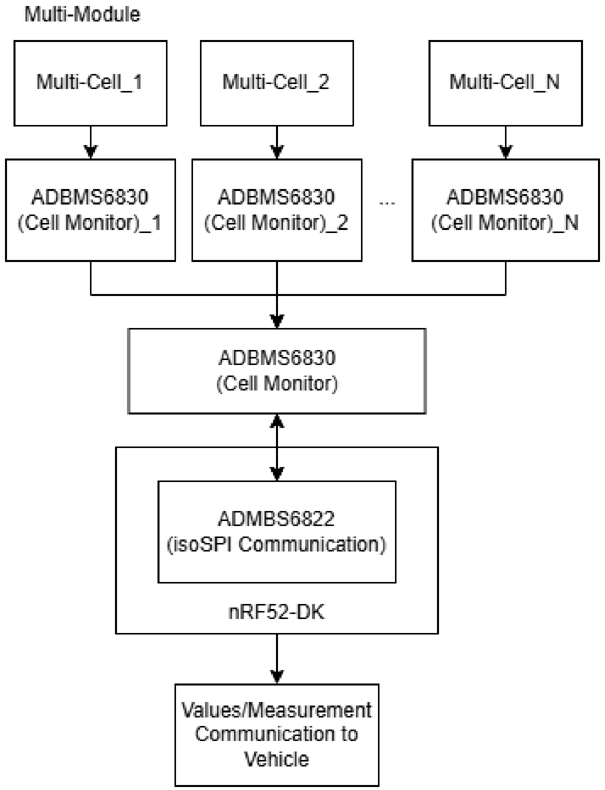
individual part/interface	item elementary function	requirement of the function	generic failure	potential failure modes
Cells	send voltage	Carries Voltage capacity within designed limits	Cell leakage, expiration, or degradation	Overvoltage, capacity fade, internal short circuit
Power Supply	Gives power	Provides stable power and ensures isolation for cell monitoring.	Power failure.	Low or no power, power spikes, faulty isolation.
CSC	allows passive/active cell balancing to occur, cell sensing circuit; checks status of cells	Balances cells and checks their status (voltage, temperature, etc.).	Incorrect cell balancing.	Unbalanced cells, failed sensing.
NTCLE thermistor	ensure the battery pack remains within safe temperature limits when CaNBus/BMU is collecting data from CSC's	Ensure temperature within range during charging/discharging.	Inaccurate temperature readings.	Overheating, undetected temperature rise.
CANbus	high-speed communication of voltage, current, temperature, and state-of-charge (SoC) data between the cells	Real-time, reliable data communication between cells and BMS.	Communication failure.	Bus communication error, signal loss, data corruption.
SOC	Monitors and reports state of charge (SOC) of cells	Accurately check the SOC of cells.	Faulty SOC readings.	SOC algorithm failure, drift in readings.

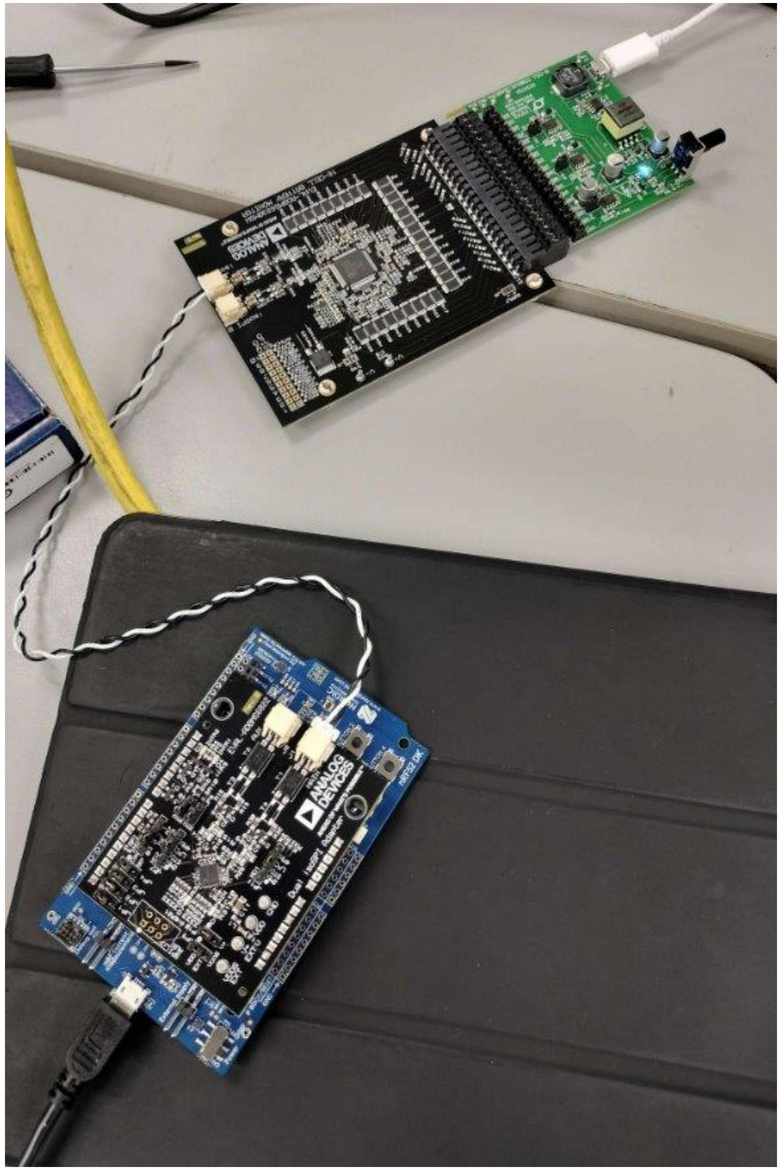
Cell Sensing Circuit (CSC)

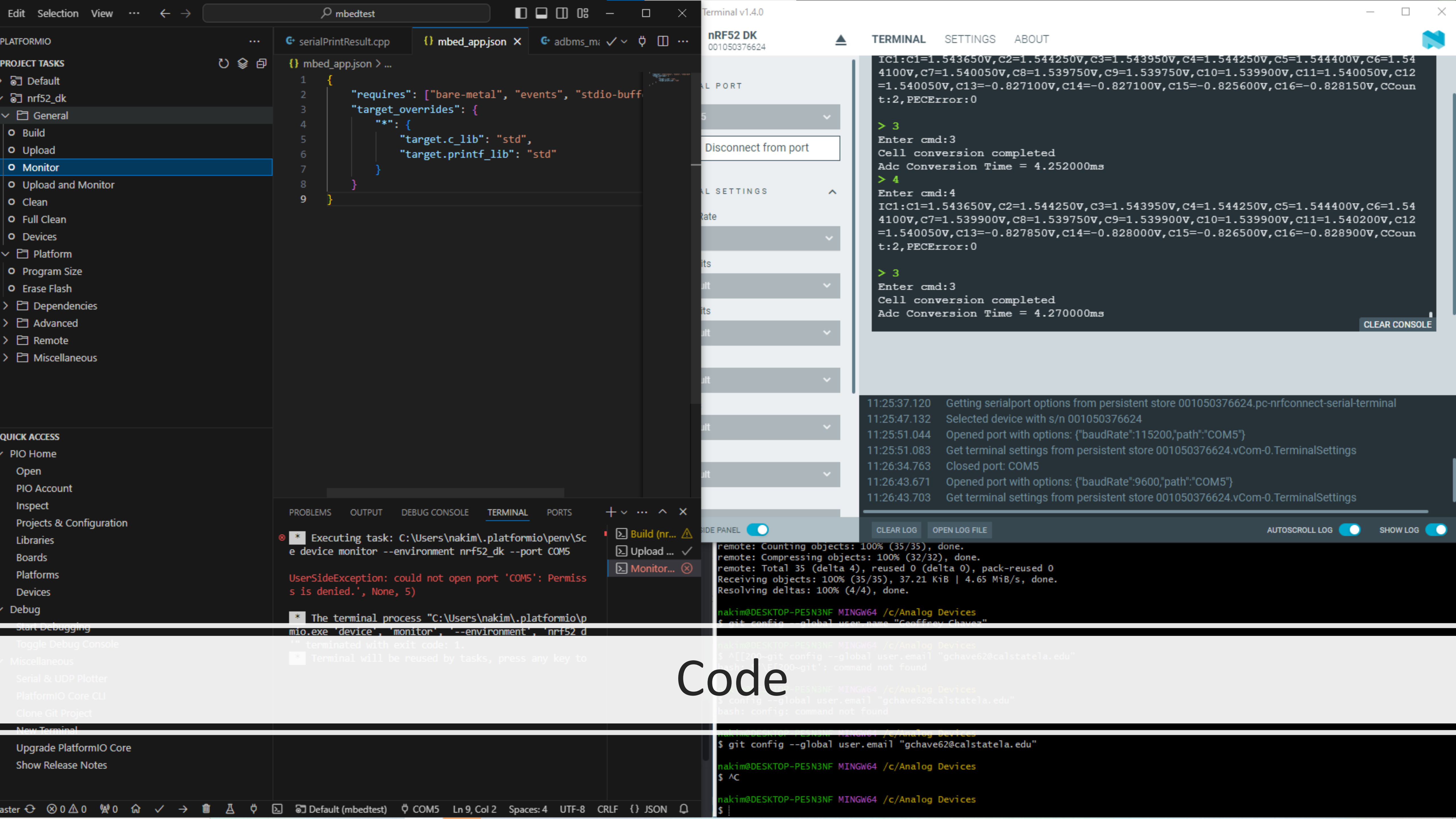
- Monitors voltage from individual cells of the battery pack
- Records temperatures
- Fault management
- Balancing Control
- Communicates with Battery Management System (BMS)

CSC Block Diagram and Circuit



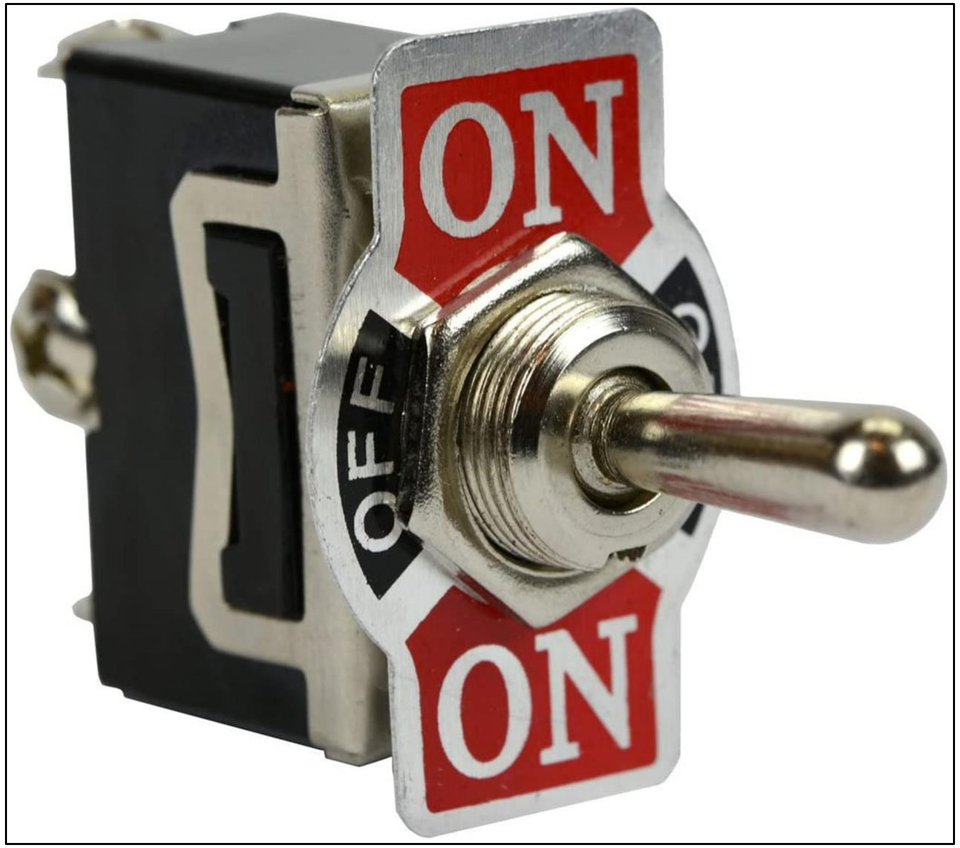





mbedtest
PLATFORMIO
PROJECT TASKS
Default
nrf52_dk
General
Build
Upload
Monitor
Upload and Monitor
Clean
Full Clean
Devices
Platform
Program Size
Erase Flash
Dependencies
Advanced
Remote
Miscellaneous
QUICK ACCESS
PIO Home
Open
PIO Account
Inspect
Projects & Configuration
Libraries
Boards
Platforms
Devices
Debug
Upgrade PlatformIO Core
Show Release Notes
serialPrintResult.cpp
mbed_app.json
{
"requires": ["bare-metal", "events", "stdio-buff
"target_overrides": {
"*": {
"target.c_lib": "std",
"target.printf_lib": "std"
}
}
}
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Executing task: C:\Users\nakim\.platformio\penv\Sc e device monitor --environment nrf52_dk --port COM5
UserSideException: could not open port 'COM5': Permission is denied.', None, 5)
The terminal process "C:\Users\nakim\.platformio\p
Build (nr...
Upload ...
Monitor...
Default (mbedtest)
COM5
Ln 9, Col 2
Spaces: 4
UTF-8
CRLF
JSON
Terminal v1.4.0
nRF52 DK
001050376624
TERMINAL
SETTINGS
ABOUT
Disconnect from port
IC1:C1=1.543650V,C2=1.544250V,C3=1.543950V,C4=1.544250V,C5=1.544400V,C6=1.544100V,C7=1.540050V,C8=1.539750V,C9=1.539750V,C10=1.539900V,C11=1.540050V,C12=1.540050V,C13=-0.827100V,C14=-0.827100V,C15=-0.825600V,C16=-0.828150V,CCount:2,PECError:0
> 3
Enter cmd:3
Cell conversion completed
Adc Conversion Time = 4.252000ms
> 4
Enter cmd:4
IC1:C1=1.543650V,C2=1.544250V,C3=1.543950V,C4=1.544250V,C5=1.544400V,C6=1.544100V,C7=1.539900V,C8=1.539750V,C9=1.539900V,C10=1.539900V,C11=1.540200V,C12=1.540050V,C13=-0.827850V,C14=-0.828000V,C15=-0.826500V,C16=-0.828900V,CCount:2,PECError:0
> 3
Enter cmd:3
Cell conversion completed
Adc Conversion Time = 4.270000ms
CLEAR CONSOLE
11:25:37.120 Getting serialport options from persistent store 001050376624.pc-nrfconnect-serial-terminal
11:25:47.132 Selected device with s/n 001050376624
11:25:51.044 Opened port with options: {"baudRate":115200,"path":"COM5"}
11:25:51.083 Get terminal settings from persistent store 001050376624.vCom-0.TerminalSettings
11:26:34.763 Closed port: COM5
11:26:43.671 Opened port with options: {"baudRate":9600,"path":"COM5"}
11:26:43.703 Get terminal settings from persistent store 001050376624.vCom-0.TerminalSettings
CLEAR LOG
OPEN LOG FILE
AUTOSCROLL LOG
SHOW LOG
remote: Counting objects: 100% (35/35), done.
remote: Compressing objects: 100% (32/32), done.
remote: Total 35 (delta 4), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (35/35), 37.21 KiB | 4.65 MiB/s, done.
Resolving deltas: 100% (4/4), done.
nakim@DESKTOP-PE5N3NF MINGW64 /c/Analog Devices
$ git config --global user.email "gchave62@calstatela.edu"
nakim@DESKTOP-PE5N3NF MINGW64 /c/Analog Devices
$ ^C
nakim@DESKTOP-PE5N3NF MINGW64 /c/Analog Devices
$

Code

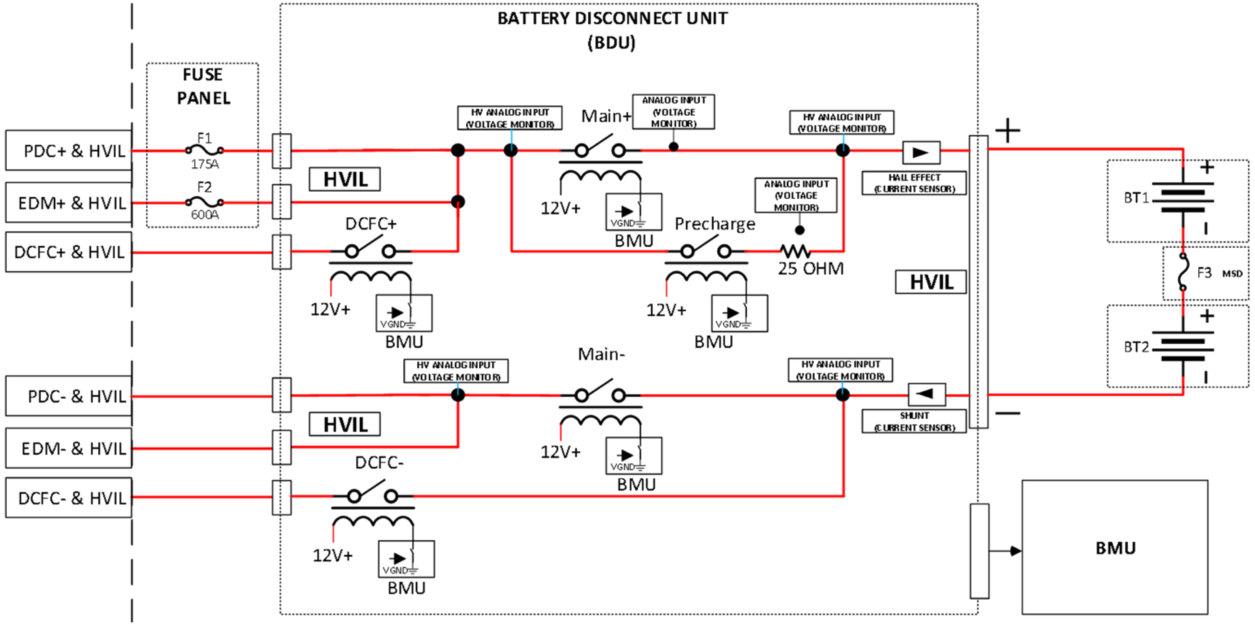
BDU - What is it?



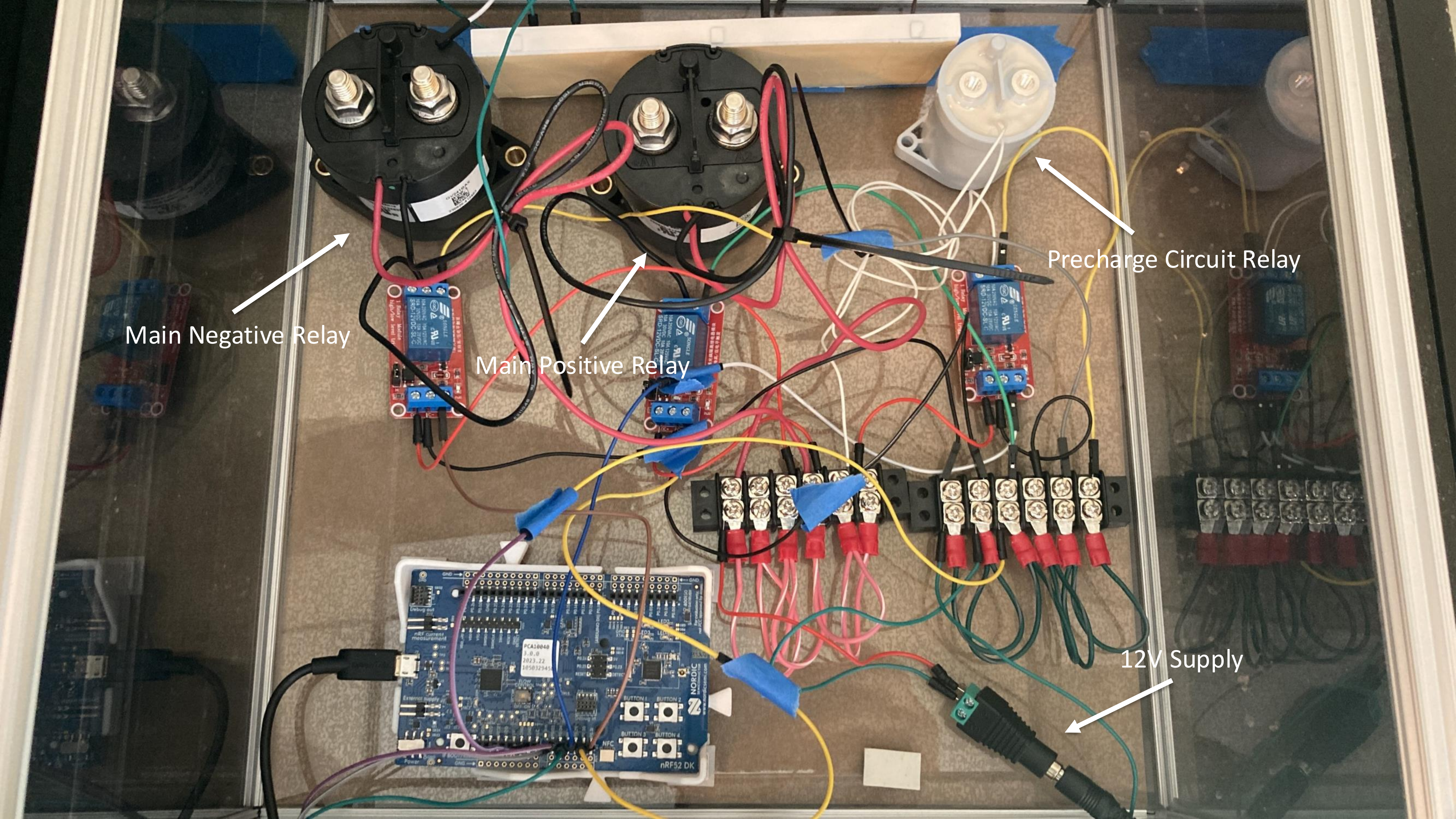
Battery Disconnect Unit

- Safety Management
- Power Distribution
- System Longevity

BDU – Functionality/ Schematic



- Primary Disconnection
  - Acts as the main link between the battery and the rest of the vehicle's electrical system.
- Pre-Charge Circuit
  - Prevents high inrush currents when connecting the battery to the powertrain.
- Sequential Activation
  - Based on State Machine implementation.


Main Negative Relay
Main Positive Relay
Precharge Circuit Relay
12V Supply
nRF52 DK
NORDIC
PCA10040

7-Segment Display & Thermistor

PIO Home | main.cpp

7_seg_and_thermistor > src > main.cpp > ...

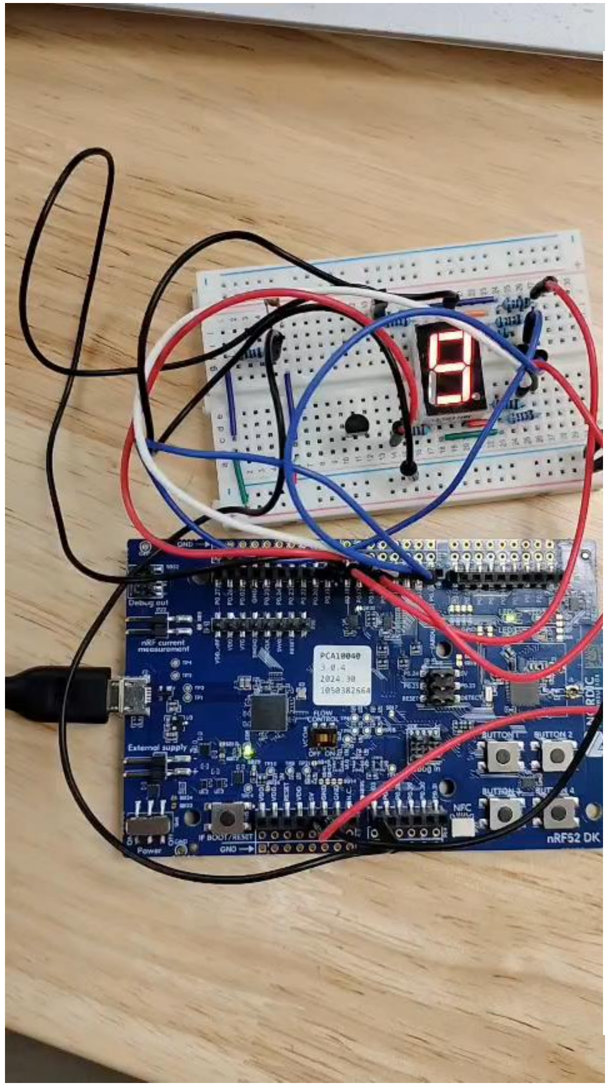
```cpp
#include "SevSeg.h"
SevSeg sevseg;

// Thermistor parameters from the datasheet
#define RT0 10000
#define B 3977

// Our series resistor value = 10000
#define R 10000

// Variables for calculations
float RT, VR, ln, TX, T0, VRT;


void setup()
{
  //***7-Seg Setup Start***
  //Set to 1 for single-digit display
  byte numDigits = 1;
  //defines common pins while using multi-digit display. Left for single digit display
  byte digitPins[] = {};

  //Defines Arduino pin connections in order: A, B, C, D, E, F, G, DP
  byte segmentPins[] = {2, 3, 4, 5, 6, 7, 8};
  byte displayType = COMMON_CATHODE; //Use COMMON_ANODE for Common Anode display
  bool resistorsOnSegments = true; //‘false’ if resistors are connected to common pin

  //Initialize sevseg object. Use COMMON_ANODE instead of COMMON_CATHODE for CA display
  sevseg.begin(displayType, numDigits, digitPins, segmentPins, resistorsOnSegments);
  sevseg.setBrightness(90);

  //***7-Seg Definition End***

  //***Thermistor Setup Start***
  // Setup serial communication
  Serial.begin(9600);
  // Convert T0 from Celsius to Kelvin
  T0 = 25 + 273.15;
  //***Thermister Setup End***
}
void loop()
{
  //Display numbers 0-9 with 1 seconds delay
  for(int i = 0; i <= 10; i++)
  {
    if (i == 10)

```

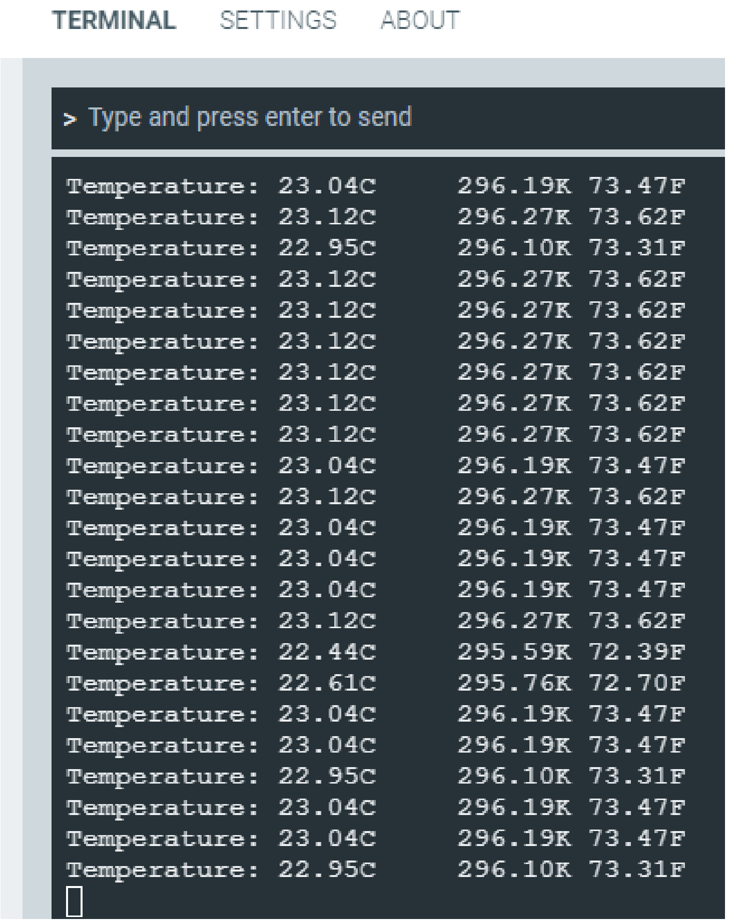
PIO Home | main.cpp

7_seg_and_thermistor > src > main.cpp > ...

```cpp
void setup()
  //***Thermister Setup End***
}
void loop()
{
  //Display numbers 0-9 with 1 seconds delay
  for(int i = 0; i <= 10; i++)
  {
    if (i == 10)

    {
      i = 0;

    }

    // Read the voltage across the thermistor
    VRT = (5.00 / 1023.00) * analogRead(PIN_A0);

    // Calculate the voltage across the resistor
    VR = 5.00 - VRT;

    // Calculate resistance of the thermistor
    RT = VRT / (VR / R);

    // Calculate temperature from thermistor resistance
    ln = log(RT / RT0);
    TX = (1 / ((ln / B) + (1 / T0))) + 3;

    // Convert to Celsius
    TX = TX - 273.15;
    Serial.print("Temperature: ");

    // Display in Celsius
    Serial.print(TX);
    Serial.print("C\t");

    // Convert and display in Kelvin
    Serial.print(TX + 273.15);
    Serial.print("K\t");

    // Convert and display in Fahrenheit
    Serial.print((TX * 1.8) + 32);
    Serial.println("F");


    sevseg.setNumber(i);
    sevseg.refreshDisplay();
    delay(1000);
  }
```

7-Segment & Thermistor Prototype

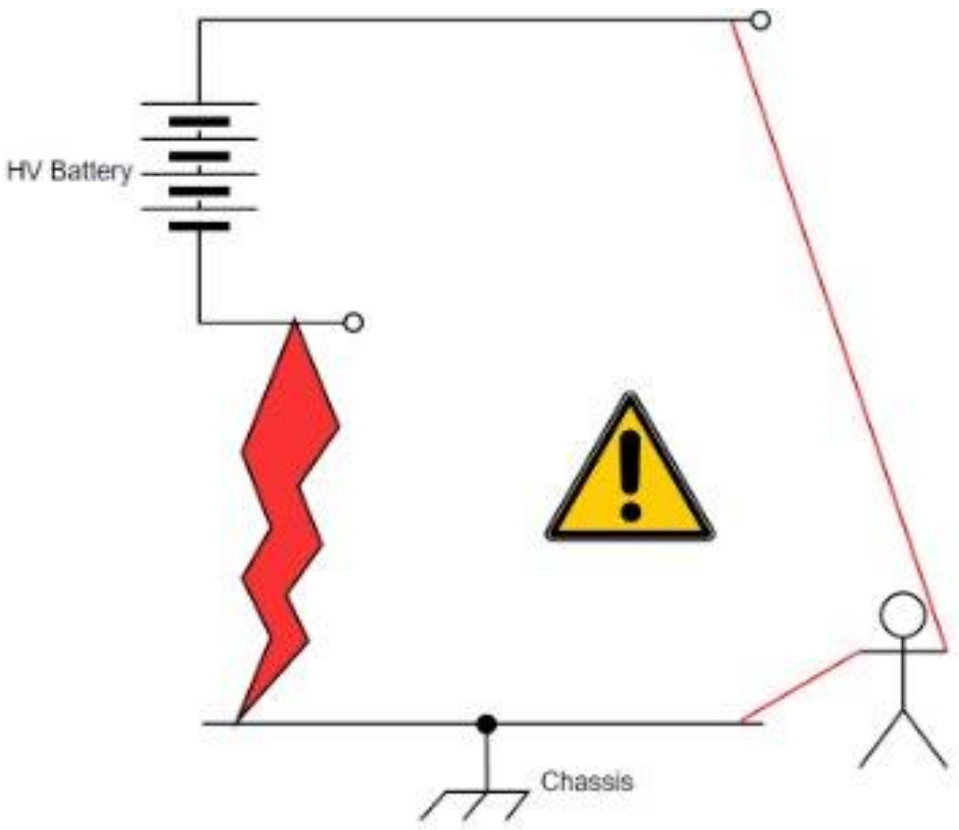




Loss of Isolation Detection

- Loss of Isolation Detection is used to monitor the isolation status between the chassis of the vehicle and the HV Bus.
    - Doing so maintains the safety of all personnel who are maintaining the vehicle, low voltage electronics, and the end user.

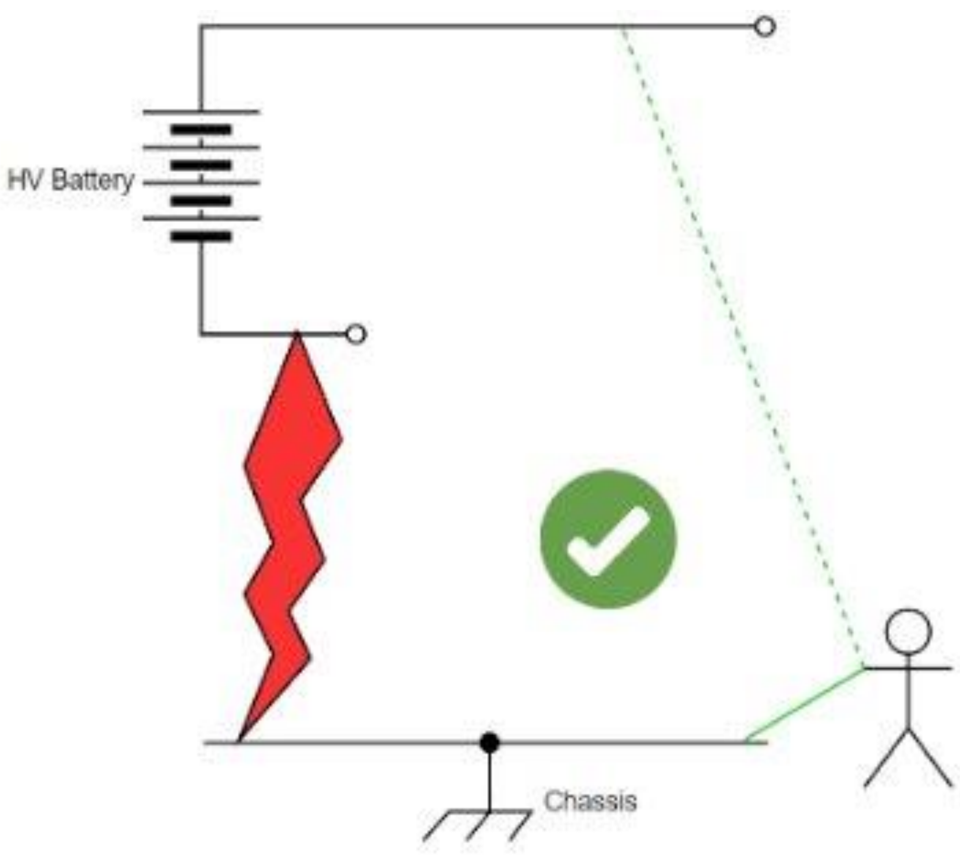
LOI Prototype – What causes LOI?



Electrical shock hazard: person touching chassis and power line

- GOAL: Create an IMD (Isolation Monitoring Device)
- Exposure to harsh environmental conditions. Including: extreme temperatures, high humidity, or corrosive chemicals.
- Vibrations or mechanical stresses stemming from vehicle operation or accidents.
- Aging and wear of insulation materials over time.
- Accumulation of dirt leading to the creation of conductive pathways.
- Incidents like coolant leakage compromising isolation.
- Tools left in the battery pack during service.

LOI Prototype – Research Findings



No hazard: person touching either chassis or power line

In general Loss of Isolation systems require at a minimum an isolation resistance of 500 (ohms/volt).

Our system operates at 436 V.

A minimum isolation resistance of 218 kohms is required.

To guarantee proper isolation the industry standard is to double the resistance.

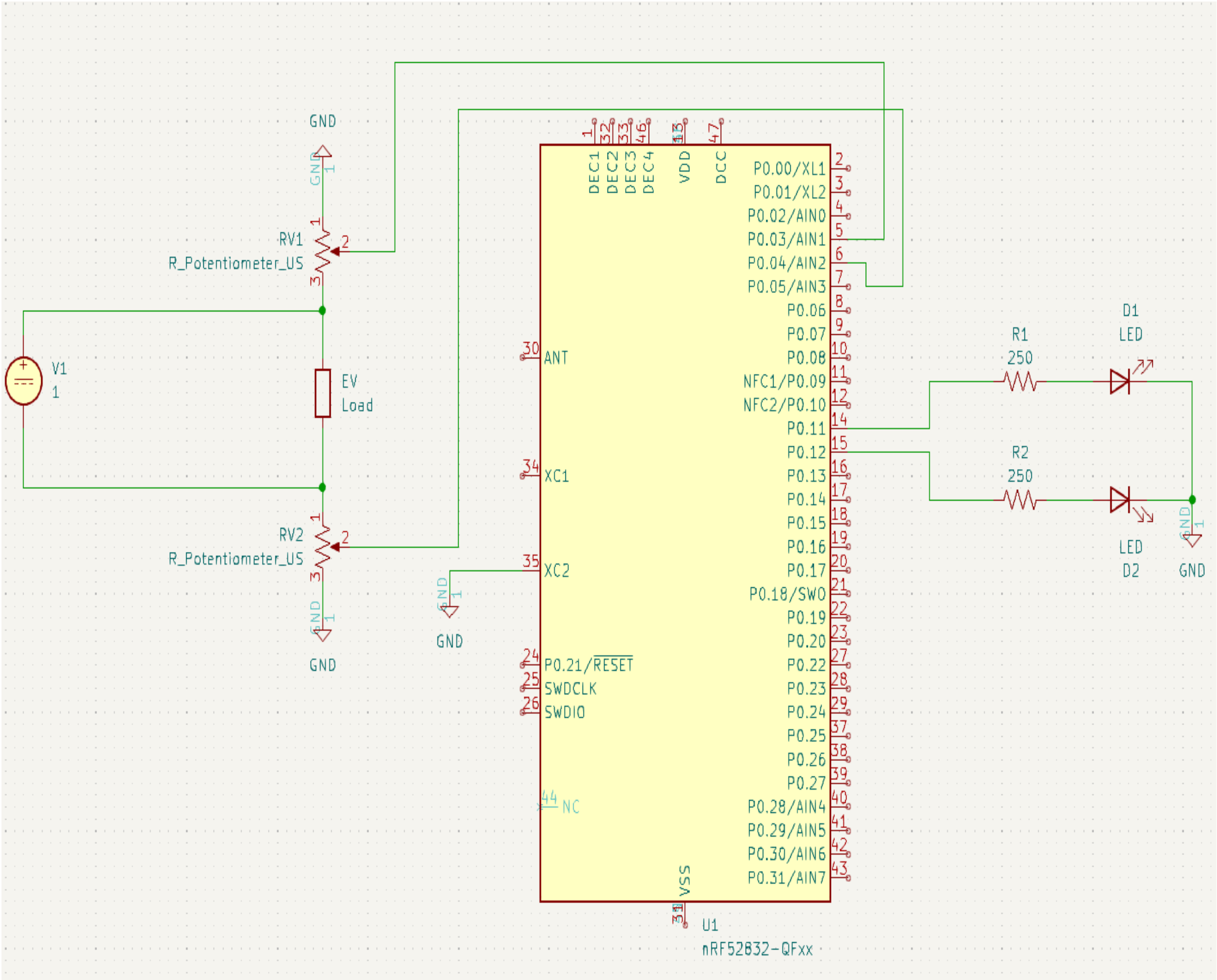
R_isolation = 500kohms

LOI Prototype – Firmware

LOI_Prototype > src > main.cpp > loop()

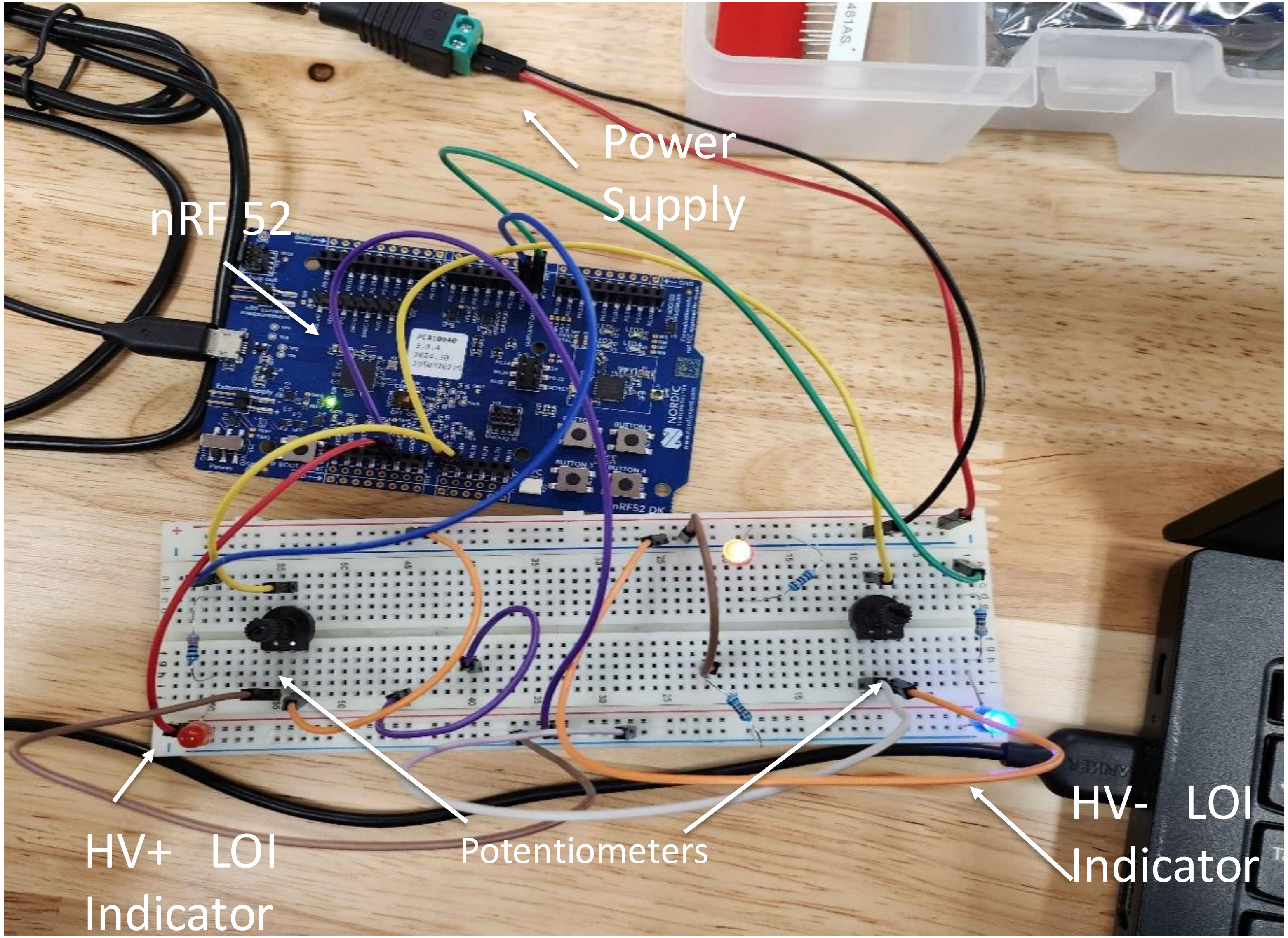
```cpp
#include <Arduino.h>


void setup() {
  Serial.begin(9600);
  // Setup 2 Pins (1 Analog for read in, 1 Digital for LED)
  pinMode(0, OUTPUT);
  pinMode(1, OUTPUT);
}

void loop() {
// int sensorValue_LOW = analogRead(A0);
int sensorValue_HIGH = analogRead(A1);

float prev_value = 2.44;
// float voltage_LOW = sensorValue_LOW * (5.0 / 1023.0);
float voltage_HIGH = sensorValue_HIGH * (5.0 /1023.0);

float curr_value = voltage_HIGH;

float diff = .15;  // Trigger Value for Monitoring, deaming LOI unsafe.


// Serial.print("LOW SIDE: ");
// Serial.println(voltage_LOW);
Serial.print("HIGH SIDE: ");
Serial.println(voltage_HIGH);


delay(500);


// When Voltage drop/rise moves outside of threshold levels an LOI event is triggered
if (abs(prev_value - curr_value) > diff ){
  Serial.println("LOSS OF ISOLATION EVENT: Error Code 42");

  // If a voltage drop occurs, we associate that with LOI on HV+ and turn on appropriate warning signals.
  if (prev_value - curr_value > 0){
    Serial.println(" - HV+");
    digitalWrite(1, HIGH);

  }

   // If a voltage rise occurs, we associate that with LOI on HV- and turn on appropriate warning signals.
  if(prev_value - curr_value < 0){
    Serial.println(" - HV-");
    digitalWrite(0, HIGH);
  }

  delay(1000);
}

// If no LOI detected warning signals remain off.
else{
digitalWrite(1, LOW);
digitalWrite(0, LOW);
  }

}

```

Loss of Isolation (LOI) Schematic



- Utilize two 10 kΩ potentiometers on the high-voltage rails to detect voltage changes indicating isolation loss; decreases at P0.04 (HV+ loss) and increases at P0.03 (HV- loss).
- Implements two LEDs (red for HV+ and blue for HV-) and a Nordic nRF52 board to signal detected isolation losses.

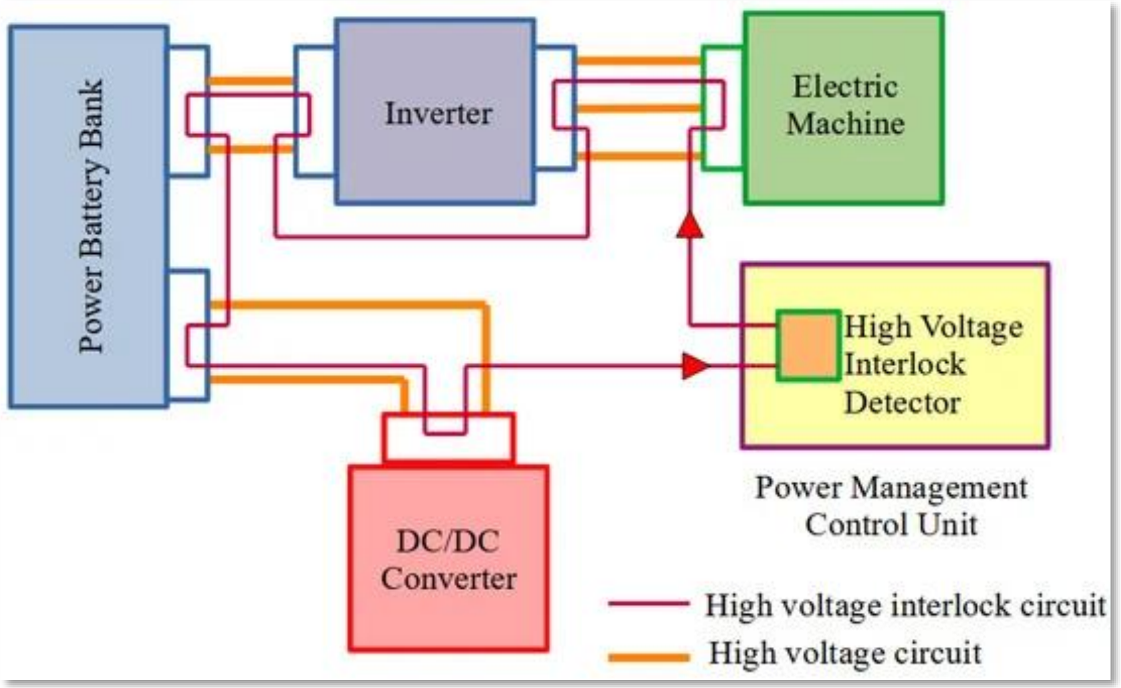
LOI Prototype – Event Detection/Signaling



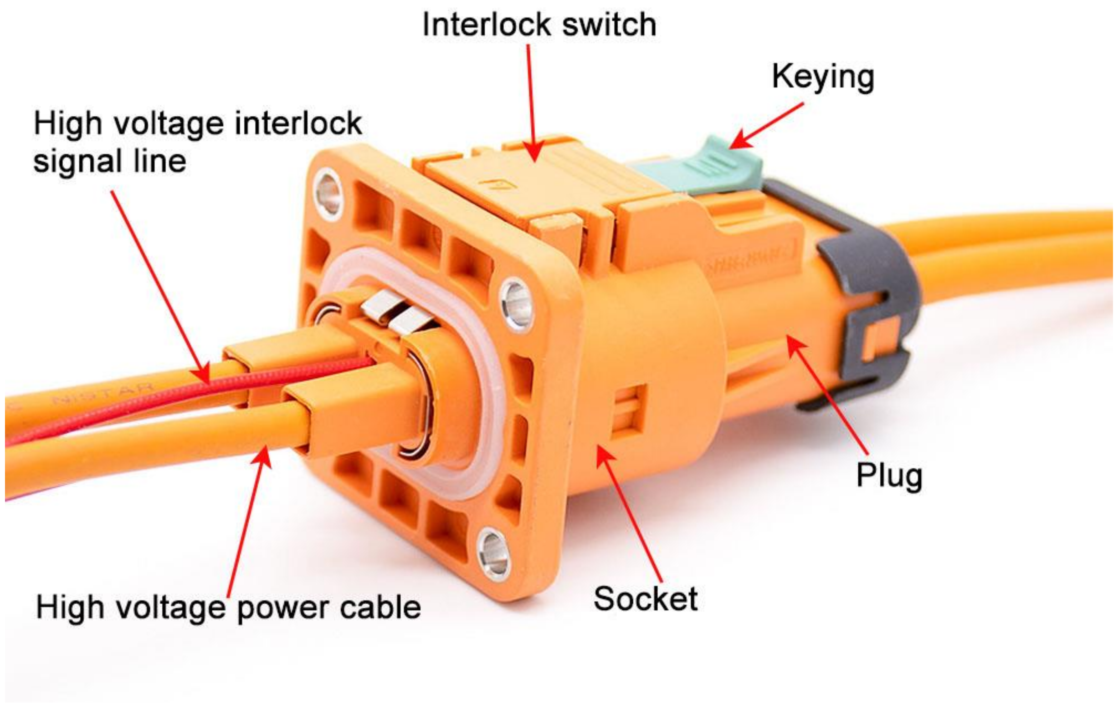
High Voltage Interlock Loop (HVIL)

- An HVIL is a safety feature commonly used in electrical systems with high-voltage applications. The purpose of an HVIL is to ensure the safe operation of the system by monitoring the presence and integrity of high-voltage connections

Example schematic of an HVIL:

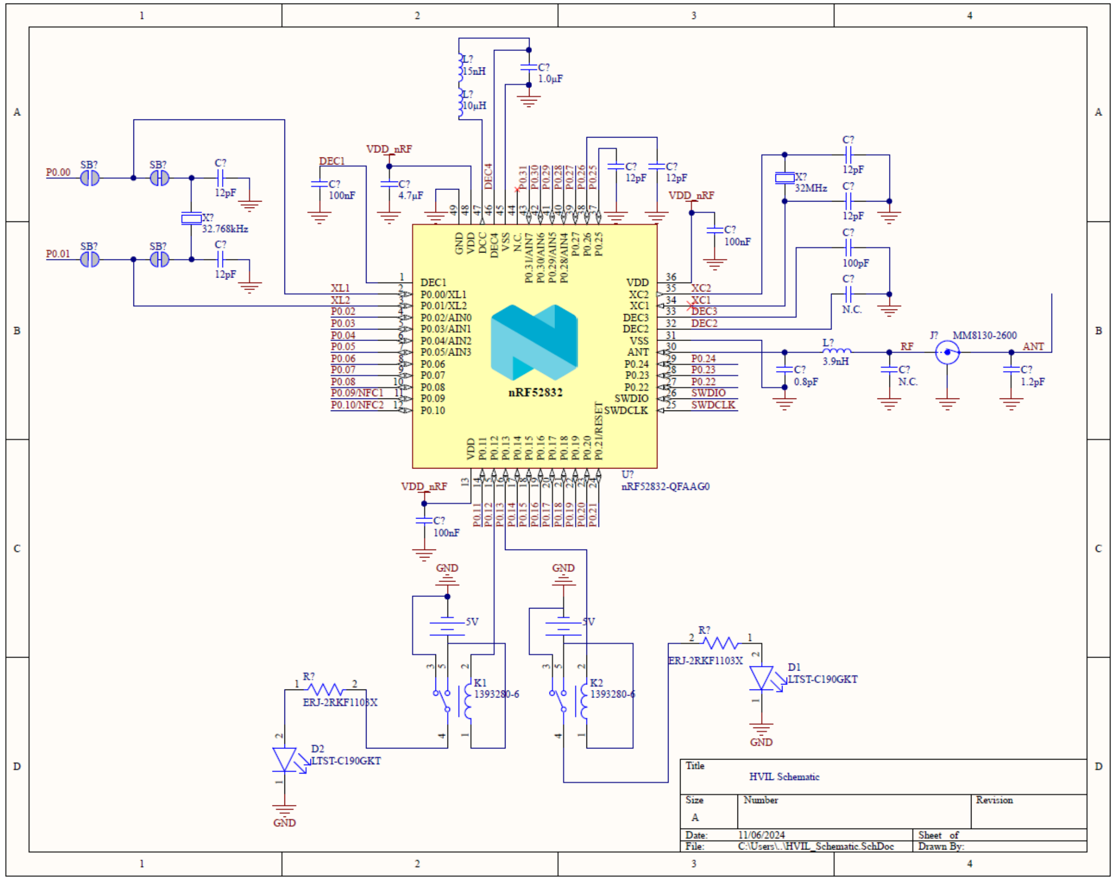


HVIL Prototype



- Goal: Create a high voltage interlock loop management system that detects faults
- Much like LOI, HVIL protects from electrical hazards, short circuits, component failures, human error, fire risks, environmental contaminants, and mechanical damage

HVIL Schematic



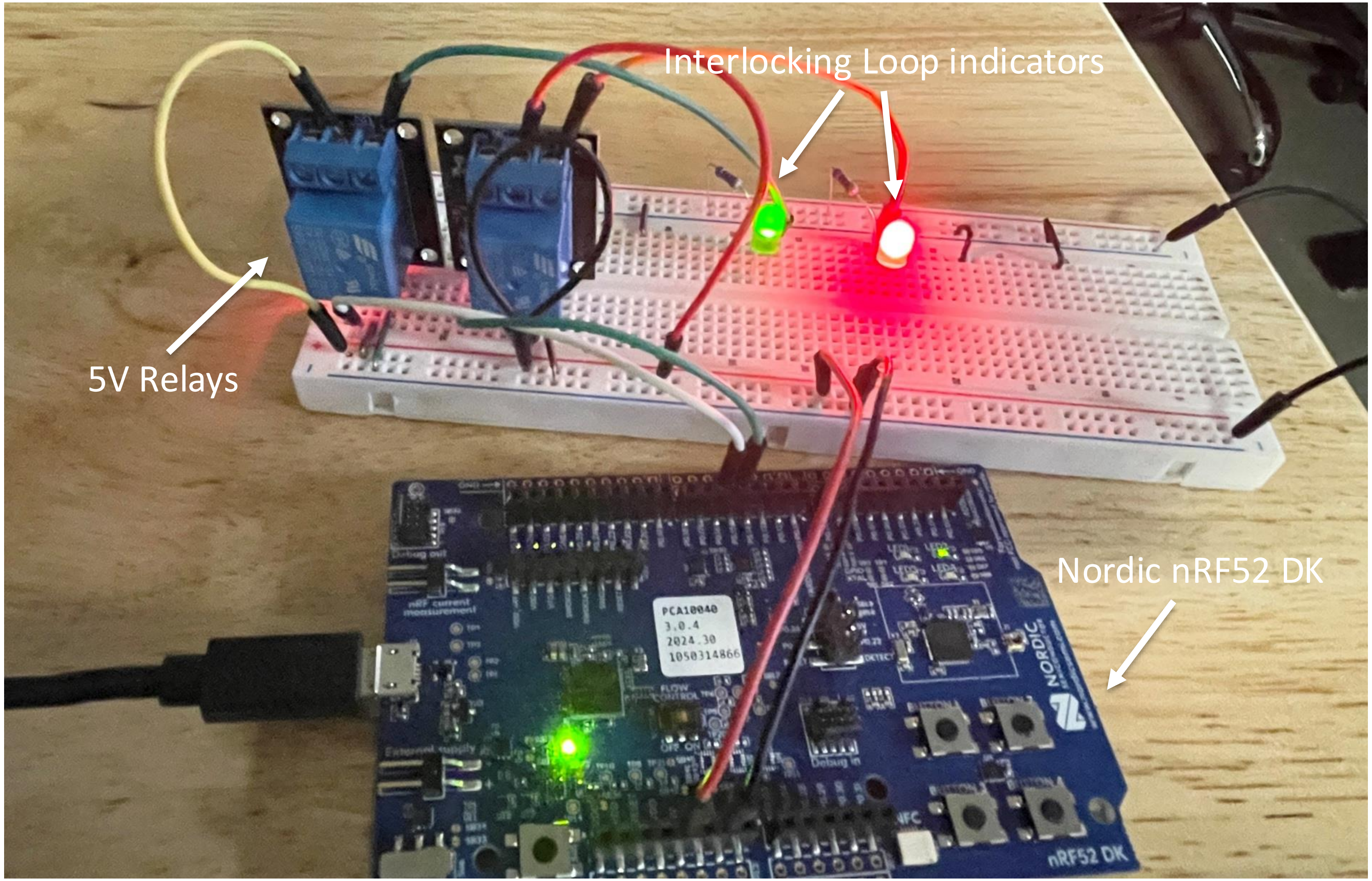
- Designed an interlock loop system using two relays and LEDs to indicate faults
- Configured relays, connected to LEDs, to mimic high-voltage systems like motors and air conditioners
- Programmed fault responses for immediate disconnection of high-voltage components

HVIL Prototype – Fault Detection

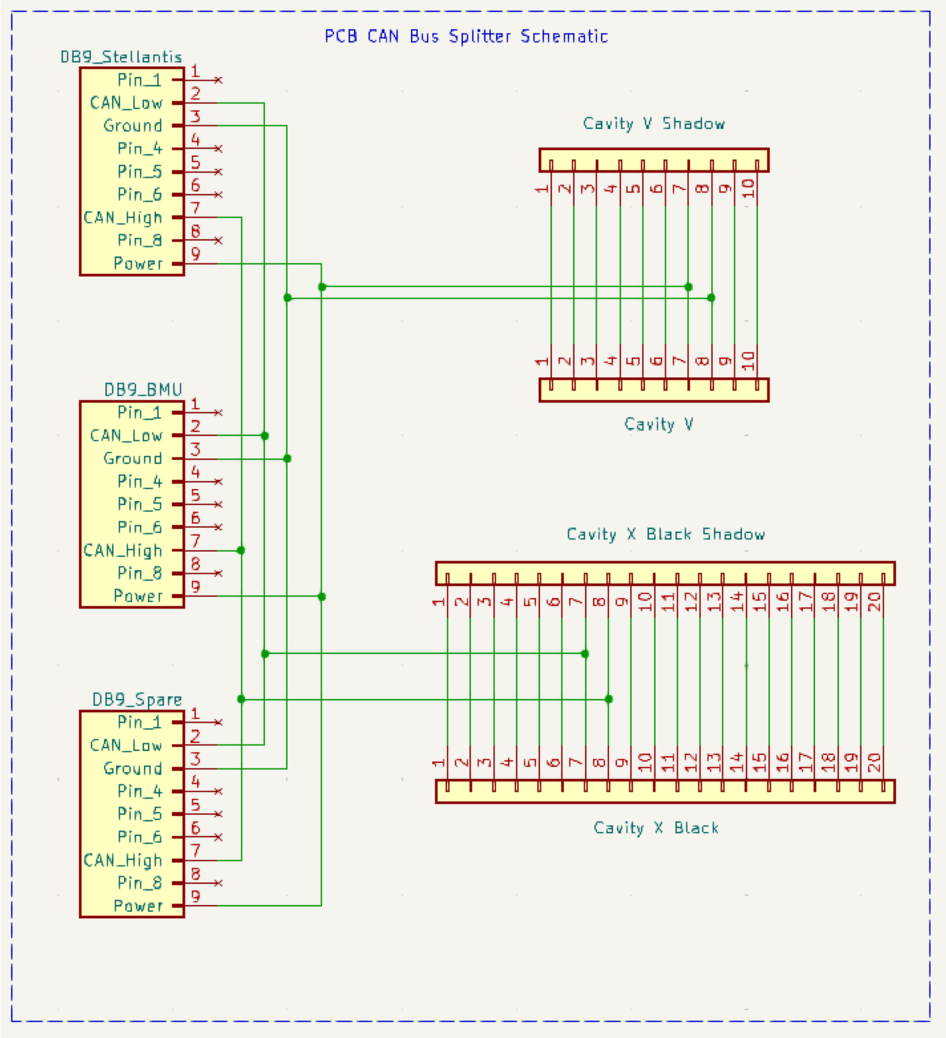
```
void loop() {
  // Check for serial input
  if (Serial.available() > 0) {
    String command = Serial.readStringUntil('\n'); // Read command
    command.trim(); // Remove any extra spaces or newlines

    if (command.equalsIgnoreCase("BREAK")) {
      Serial.println("Simulating HVIL break...");
      digitalWrite(relayACPin, inactiveLow);  // Immediately turn off AC relay
      digitalWrite(relayMotorPin, inactiveLow);  // Immediately turn off Motor relay
    } else if (command.equalsIgnoreCase("RESET")) {
      // Reset both relays to ON state
      relayACDisconnected = false;
      relayMotorDisconnected = false;
      digitalWrite(relayACPin, activeHigh);  // AC relay ON
      digitalWrite(relayMotorPin, activeHigh);  // Motor relay ON
      Serial.println("Resetting HVIL system...");
    } else if (command.equalsIgnoreCase("DISCONNECT_AC")) {
      relayACDisconnected = true; // Simulate disconnection of AC relay
      Serial.println("Simulating AC relay disconnection...");
      digitalWrite(relayACPin, inactiveLow);  // Immediately turn off AC relay
      lastDisconnectTime = millis();  // Start the 1-second timer
    } else if (command.equalsIgnoreCase("DISCONNECT_MOTOR")) {
      relayMotorDisconnected = true; // Simulate disconnection of Motor relay
      Serial.println("Simulating Motor relay disconnection...");
      digitalWrite(relayMotorPin, inactiveLow);  // Immediately turn off Motor relay
      lastMotorDisconnectTime = millis();  // Start the 1-second timer for motor
    } else if (command.equalsIgnoreCase("RECONNECT")) {
      // Reset both relays to ON state
      relayACDisconnected = false;
      relayMotorDisconnected = false;
      digitalWrite(relayACPin, activeHigh);  // AC relay ON
      digitalWrite(relayMotorPin, activeHigh);  // Motor relay ON
      Serial.println("Relays reconnected. System functioning normally.");
    } else {
      Serial.println("Unknown command. Use 'BREAK', 'RESET', 'DISCONNECT_AC', 'DISCON
```

```
// If the AC relay was disconnected, wait for 1 second, then disconnect the motor relay
if (relayACDisconnected && !relayMotorDisconnected && (millis() - lastDisconnectTime >= 1000)) {
  // After 1 second, disconnect the motor relay
  digitalWrite(relayMotorPin, inactiveLow);  // Turn off Motor relay after 1 second
  relayMotorDisconnected = true;  // Mark the Motor relay as disconnected
  Serial.println("Simulating Motor relay disconnection after 1 second.");
}

// If the motor relay was disconnected, wait for 1 second, then disconnect the AC relay
if (relayMotorDisconnected && !relayACDisconnected && (millis() - lastMotorDisconnectTime >= 1000)) {
  // After 1 second, disconnect the AC relay
  digitalWrite(relayACPin, inactiveLow);  // Turn off AC relay after 1 second
  relayACDisconnected = true;  // Mark the AC relay as disconnected
  Serial.println("Simulating AC relay disconnection after 1 second.");
}

// Apply the HVIL status message
if (relayACDisconnected || relayMotorDisconnected) {
  // A fault is detected if either relay is disconnected
  Serial.println("HVIL system: Break detected. Both relays OFF.");
} else {
  // If both relays are connected, the system is functioning normally
  Serial.println("HVIL system: Functioning normally. Relays ON.");
}

delay(100); // Refresh every 100ms
```

HVIL Prototype

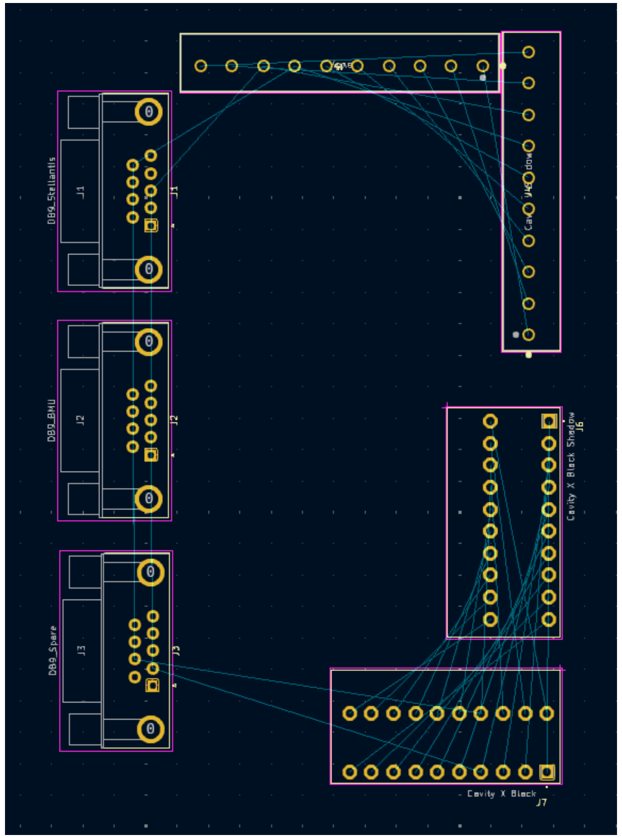


PCB (CAN Bus Splitter)

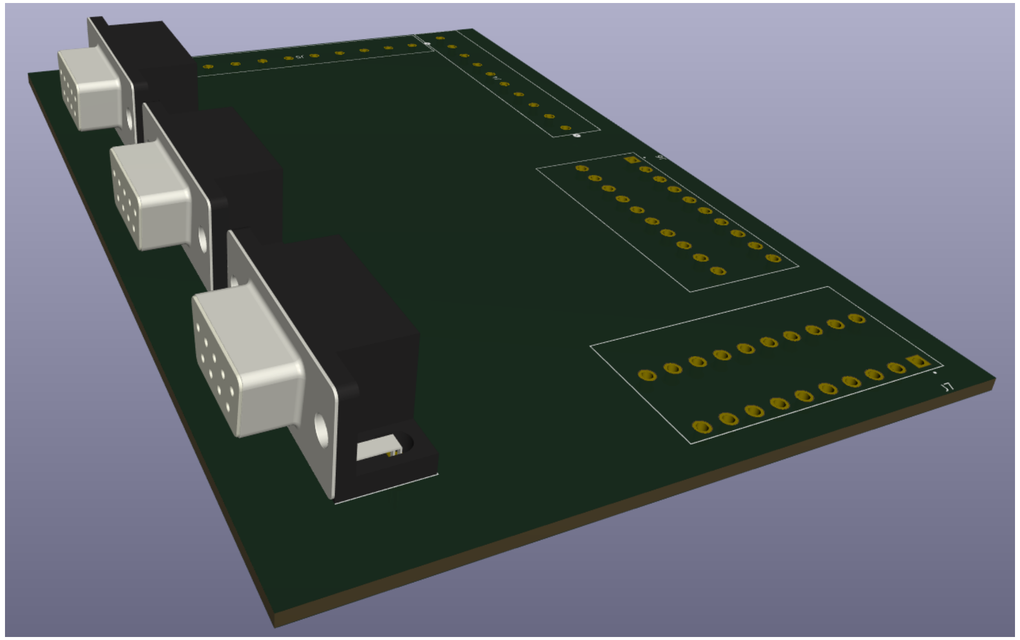


- Designed PCB to interface M450 microcontrollers with electric vehicle/ECU
- Chose DB9 connectors for the connection
- Initial draft used two DB9 connectors for microcontroller-ECU connection
- Revised design to include a third DB9 for spare and shadow cavities to reduce wear on actual connectors
- Final design simplified and reorganized

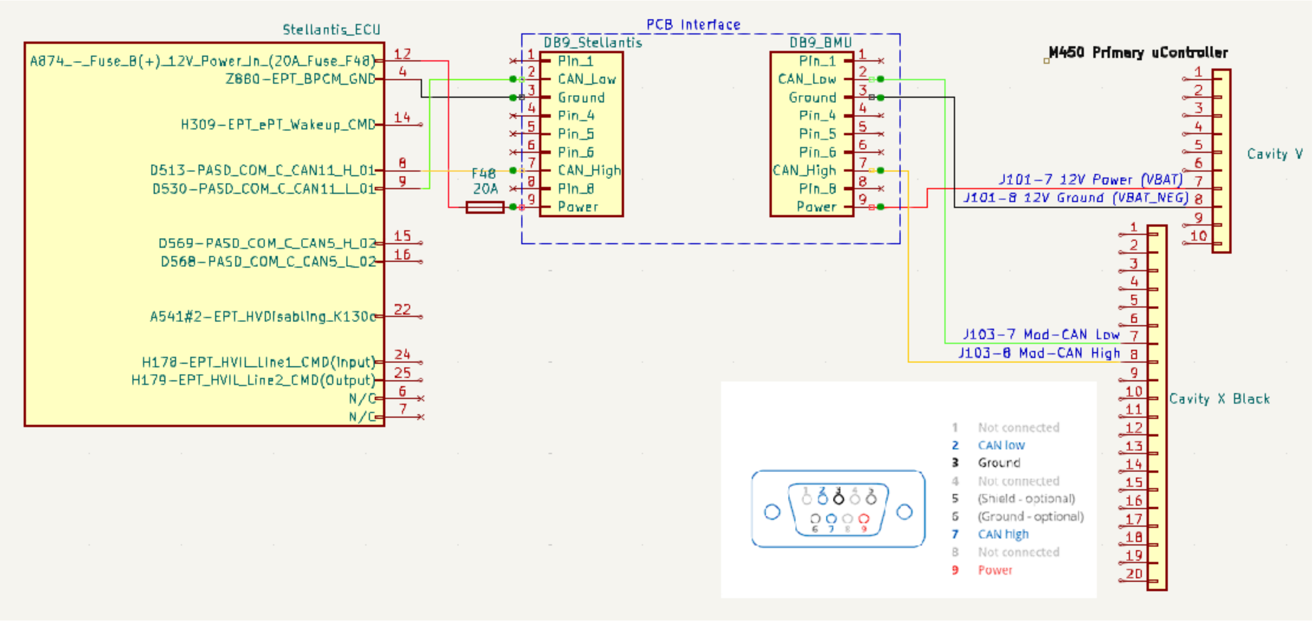
PCB (CAN Bus Splitter) cont.



PCB Editor Figure

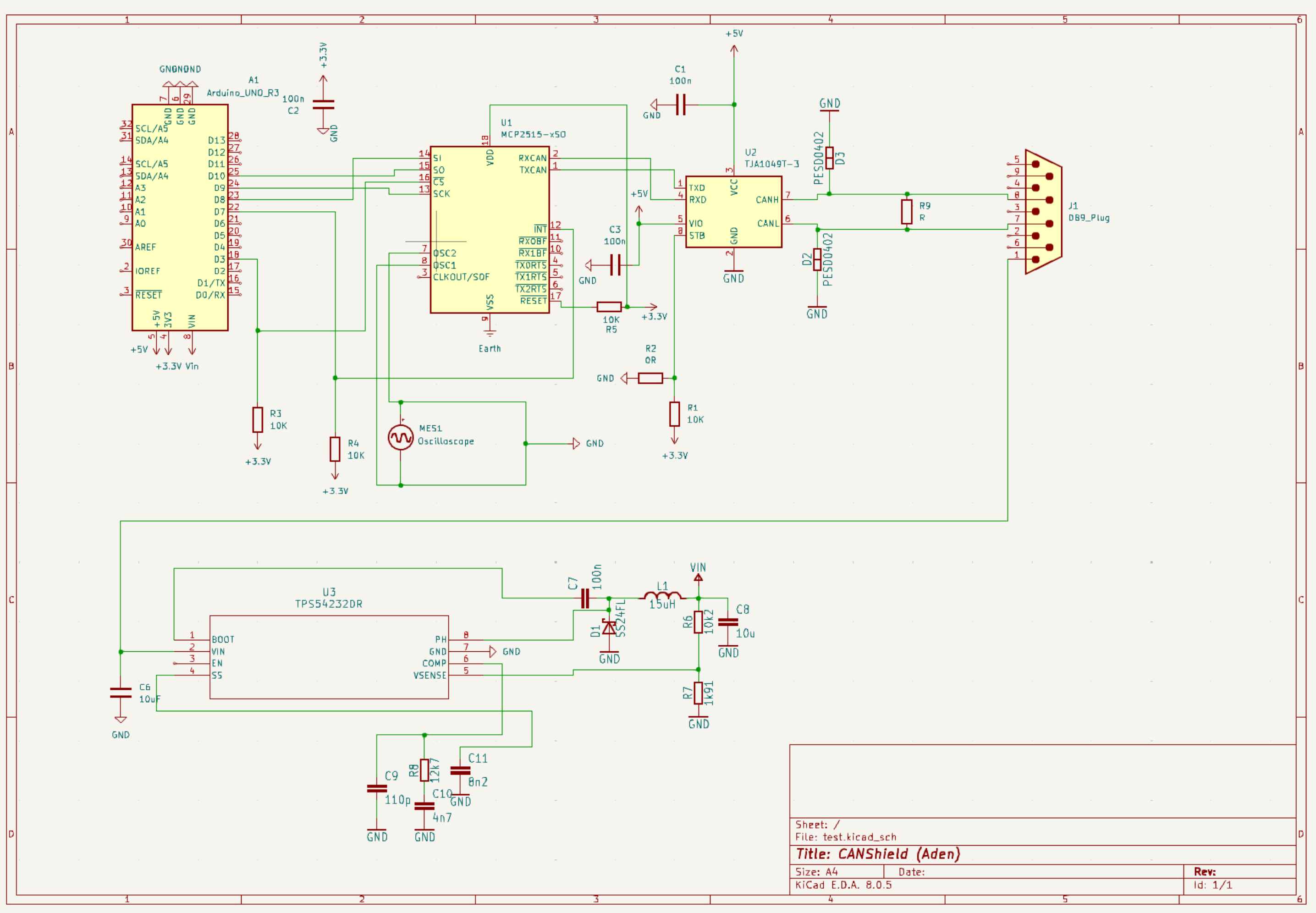


3D Model of PCB



Initial Schematic Draft

PCB (CAN Bus Shield)



- Designed PCB for CAN shield to regulate power and protect the CAN bus from voltage spikes and noise

Key components include:

- **MCP2515 (U1)**: CAN controller IC that handles communication protocol via SPI interface
- **TJA1049T/3 (U2)**: CAN transceiver that converts digital messages to differential signals (CANH and CANL)
- **10MHz Crystal Oscillator (MES1)**: Provides clock signal to the MCP2515 for accurate timing
- **TPS5432DR (U3)**: DC-DC buck converter that steps down input voltage to stable 3.3V for the MCP2515 and other components
- **Capacitors (C1, C2, C3, etc.)**: Decouple and filter circuit to stabilize voltage, smooth spikes, and filter noise
- The design ensures proper operation by sending CAN data from the microcontroller to MCP2515, which formats and transmits data via TJA1049T/3 to the CAN bus through the DB9 connector

Agenda

- Project Background
- Project Organization
- Designs Overview
  - Design Failure Mode and Effects Analysis (DFMEA)
  - Cell Sensing Circuit
  - Battery Disconnect Unit
  - 7-Segment Display and Thermistor
  - HVIL/LOI Protypes
  - PCB Design
- **Conclusion**

Conclusion

- Was it successful? Did we meet expectations?
- Our team successfully designed and developed key components of the Battery Pack Control Module (BPCM), including the Cell Sensing Circuit (CSC), Battery Disconnect Unit (BDU), 7-Segment Display, Thermistor, LOI, and HVIL prototypes. These systems work together to ensure battery safety, monitoring, and functionality, meeting the objectives outlined by the competition. Moving forward, we aim to refine these designs for full integration into the EV while complying with Stellantis requirements.

U.S. DEPARTMENT OF ENERGY

STELLANTIS

BATTERY WORKFORCE CHALLENGE PROGRAM