Temperature Sensing and Data Delivery for EV Battery Management

Team 9

Dylan McCain, Ryan Seidel, Beckett Jaeger, Alex Herzog

Contents

1. Overview

1.1 Purpose of the System

The Temperature Sensing and Data Delivery system is designed to monitor and report battery cell temperatures in the EV battery pack developed by the Rose-Hulman Battery Workforce Challenge Team. Real-time temperature monitoring is essential to prevent thermal runaway and to optimize the cooling and overall performance of the battery pack.

1.2 Integration with the EV Battery Pack

The system seamlessly integrates with the EV battery pack by connecting directly to the battery management system (BMS) via an I2C bus. Temperature sensors are strategically placed within the battery modules, ensuring accurate thermal monitoring. The data collected is processed and transmitted for real-time analysis and decision-making.

1.3 Components Overview

- **Process Board** – The central unit responsible for data collection, processing, and communication.
- **Sensor Modules** – Individual temperature sensors that connect directly to the process board.
- **Microcontroller (ATMega328PB)** – Runs the system software, manages sensor data, and controls communication.

1.4 Basic Operational Principles

The system continuously collects temperature data from up to 32 sensors connected to the process board. The microcontroller processes the data and transmits it over an I2C bus to the BMS. This allows the BMS to monitor battery temperatures in real time and take necessary actions to prevent overheating or thermal imbalances.

2. Hardware

2.1 Process Board

This section outlines the materials, connections, PCB fabrication, and assembly processes involved in the design and production of the Process Board. It also explains the design choices behind the board. The Process Board includes an ATMEGA328PB microcontroller, eight sensor connectors, two switching regulators, and a multiplexer for sensor addressing.

2.1.1 Fabrication and Assembly

Refer to the appendix for PCB fabrication and assembly instructions.

2.1.2 Connections, I/O and testing pads

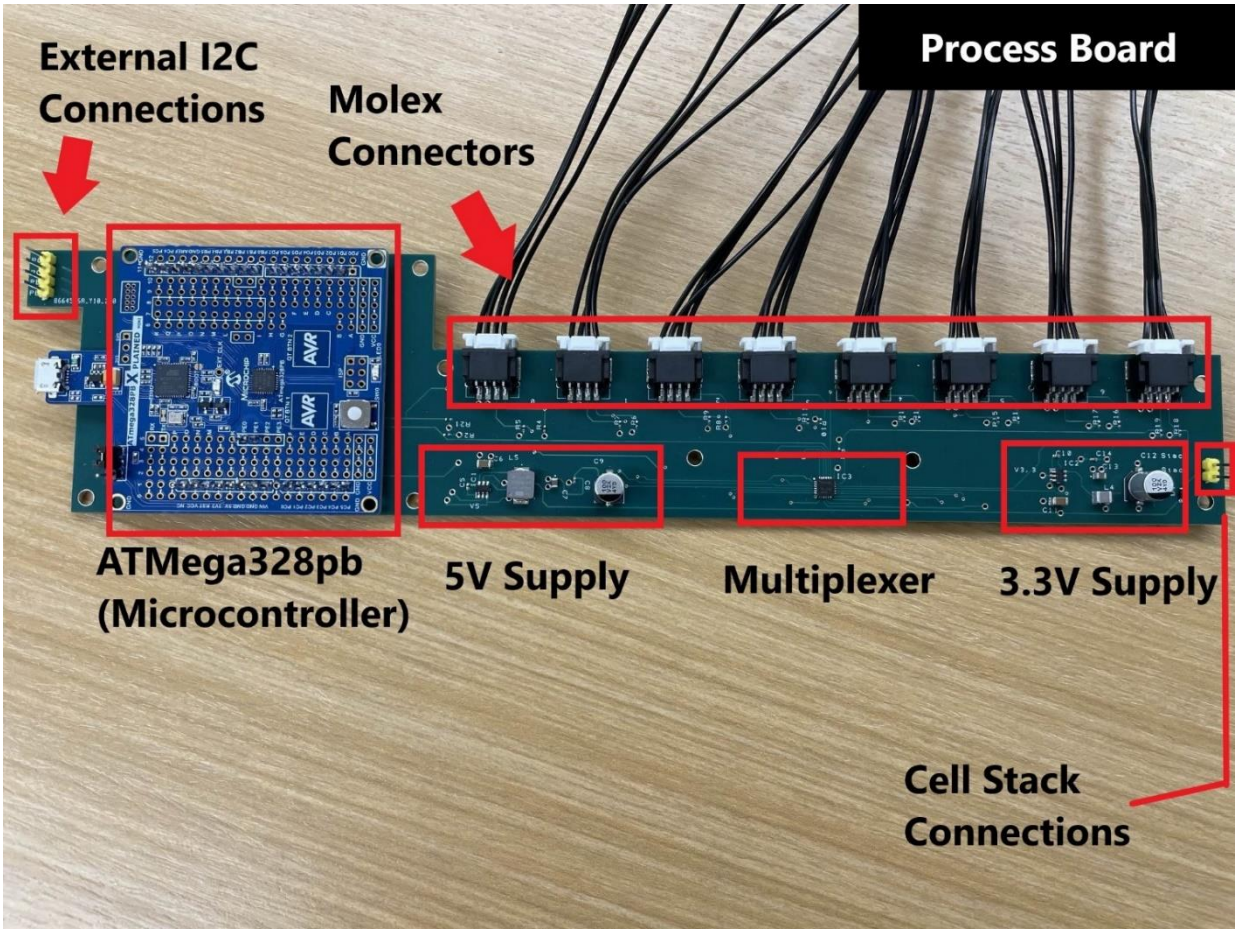

Figure: Subsection view of the process board

2.1.3 Design and Functionality

PCB Composition

The PCB is a 4-layer board with a signal – ground – 3.3V power – signal stack-up. The ground and power planes help reduce noise and improve signal integrity. The board has a standard thickness of 1.6 mm, with 0.5 oz inner copper and 1 oz outer copper to balance conductivity and thermal performance. A 4-layer design was chosen to provide cleaner signal routing and reliable power distribution.

ATMega328PB Evaluation Board

The ATMega328PB serves as the system's main processor. It runs the system software, manages communication with the sensors, and processes incoming data before passing it to the BMS. This microcontroller was chosen for its low power consumption, adequate processing capability, and built-in dual I2C capabilities, which simplifies communication with the temperature sensors and the BMS.

Sensor Connectors

The system includes eight 4-pin Molex connectors, which serve as the interface for the external temperature sensors. Each connector provides 3.3V power, ground, SDA, and SCL, supporting I2C communication. Each connector can accommodate two sensor boards, totaling four sensors per connector, allowing for efficient expansion while minimizing the number of required connections. The connectors were chosen to provide a low profile and secure interface.

Power Regulation

The power system includes 5V and 3.3V switching regulators, chosen to meet the voltage requirements of different circuit components. Both regulators take in the cell stack voltage and step it down to the necessary levels. The 5V supply powers the ATMega328PB and serves as the pull-up voltage for I2C communication with the BMS. The 3.3V supply is connected to the PCB power plane and powers the rest of the system. A switching regulator was chosen because it met the power requirements, offered a well-documented datasheet for easier integration, and provided greater efficiency compared to a linear regulator.

Sensor Multiplexer (MUX)

The multiplexer (MUX) allows the microcontroller to communicate with multiple sensors using a shared I2C bus. Since the I2C protocol has a limited number of available addresses, the single MUX enables the system to manage 32 sensors without address conflicts. This ensures that the microcontroller can selectively communicate with each sensor as needed, optimizing data collection.

Additional Components

- Resistors: Used as pull-up resistors on the I2C lines to maintain signal integrity and ensure reliable communication.
- Capacitors and Inductors: Used for filtering on the 5V and 3.3V power supplies, reducing noise and stabilizing voltage output.
- Diode: Provides reverse bias protection, preventing damage from incorrect power connections.

2.2 Sensor Module

This section outlines the materials, fabrication, and assembly processes involved in the design and production of the sensor module. It also explains the design choices behind the module. The sensor module incorporates two TMP112 temperature sensors along with two passive circuit components.

2.2.1 Fabrication and Assembly

Refer to the appendix for PCB fabrication and assembly instructions.

2.2.2 Connections, I/O and testing pads

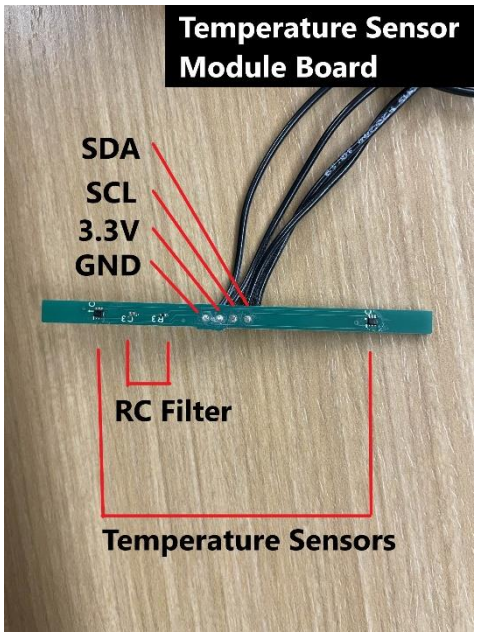

Figure: Subsection view of the Temp Sensor Board

2.2.3 Design and Functionality

The purpose of the temperature sensors is to measure the temperature of the cell. The temperature sensors are positioned at opposite ends of the sensor module to measure the temperature at both the top and bottom of a cell. The module's height matches that of the cell it is designed to monitor. The sensors are not placed at the extreme top or bottom of the module, as those areas are likely occupied by external structural materials that secure the sensor module in place.

The filter subsection is designed to minimize electrical noise affecting the temperature sensors. Its components were selected to be as small as possible to ensure they do not interfere with the contact between the temperature sensors and the thermal interface pad, which sits between the sensor module and the cell.

The I/O subsection is designed to accommodate small wires reaching the sensor module, as there is insufficient space for a standard connector and the module’s current draw is minimal. Given that the sensor module may be positioned in the middle of the battery pack, where space is extremely limited, through-hole connections were chosen as a practical solution.

3. Software

3.1 GitHub

3.1.1 Repository Overview

Repository: https://github.com/rhit-jaegerba/Temperature-Sensing-and-Data-Delivery-for-EV-Battery-Management

The GitHub repository for this project contains all of the code for the full usage of this system, the code necessary to replicate the tests that have been run on the system, and simple example files to validate functionality of individual components in the system.

3.1.2 Repository

- NormalOperation
  - Arduino file to be flashed onto the process board
  - Explained under 3.4 Software Normal Operation
- TestingDataAndFiles
  - All files pertaining to the testing we put our system through
  - DataLogger.py - software to log temperature data on raspberry pi
  - Team_9_Smooth 1.wpf - Thermal chamber profile
  - data2-2-2025.csv - Sensor data collected by the DataLogger.py
  - Team_9_Smooth_02012025_165947.csv - Oracle Data from thermal chamber
- DemoPlot.py
  - Python script to plot Min, Max, Avg works with NormalOperation
- DepreciatedCode
  - Old code that may be a helpful reference.

3.2 ATMega328pb Bootloader

3.2.1 Required Tools and Downloads

Microchip Studio (v7.0.2594: https://www.microchip.com/en-us/tools-resources/develop/microchip-studio

Arduino IDE (v2.3.4): https://www.arduino.cc/en/software

Atmega328PB Mini Core Bootloader: https://github.com/stefanrueger/urboot.hex/blob/main/mcus/atmega328pb/watchdog_2_s/external_oscillator_x/16m000000_hz/%2B%2B19k2_baud/uart0_rxd0_txd1/led%2Bb5/urboot_m328pb_2s_x16m0_19k2_uart0_rxd0_txd1_led%2Bb5.hex\

ATMega328pb Board: https://www.microchip.com/en-us/development-tool/atmega328p-xmini

3.2.2 Installation Instructions

1. Flashing the Bootloader

- **Reference Sources:**
  - Preparing Atmega Xplained Mini for Arduino IDE (Windows)
  - Bootloader Installation via Arduino
- Open **Microchip Studio**, connect the **ATmega328PB** via micro USB and make sure the jumper is in the programming position, and open the programming menu.
- Select **mEDBG** as the tool and **ATmega328PB** as the device, then click **Apply**.

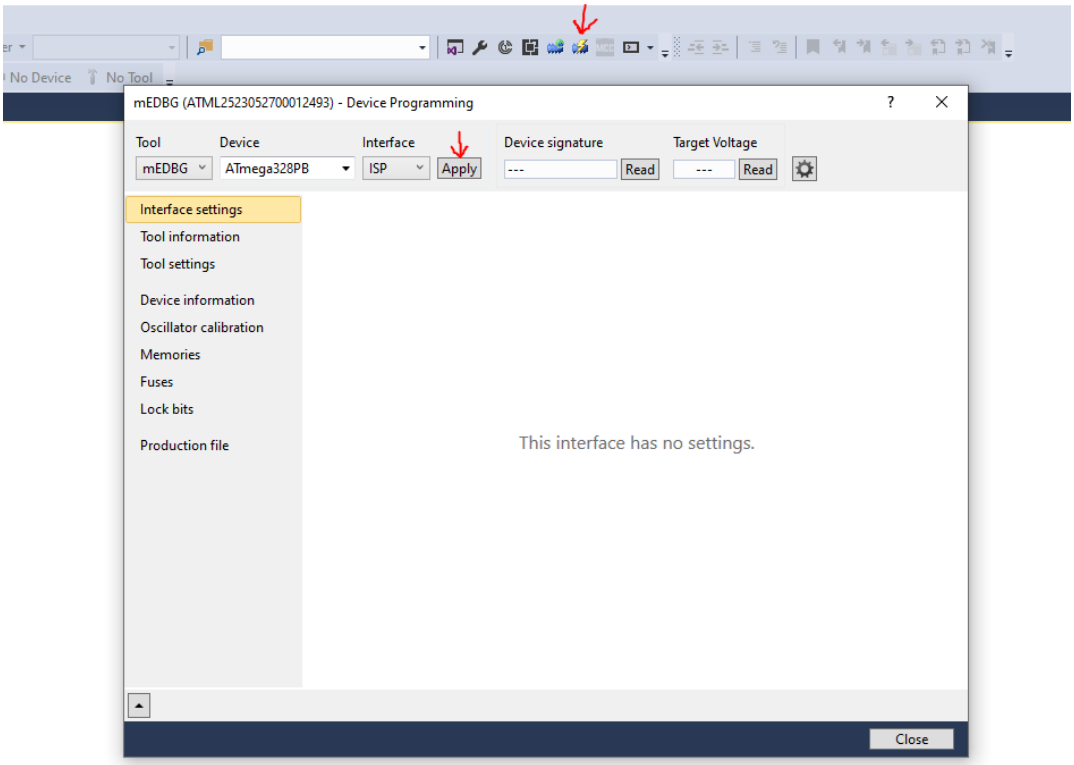

Figure: Device Programing Window

2. Configuring ATmega328PB Fuses

Enable the following fuses:

- **Brownout detection:** 4.3V
- **Bootloader size:** 256 words (512 bytes)
- **Clock source:** External (mEDBG from AVR Xplained Mini)

Fuse Values for ATmega328PB:

Fuses\Product	atmega328PB
Low	0xE0
High	0xDE
Extended	0xFC

Click **Program** to apply the fuse settings.

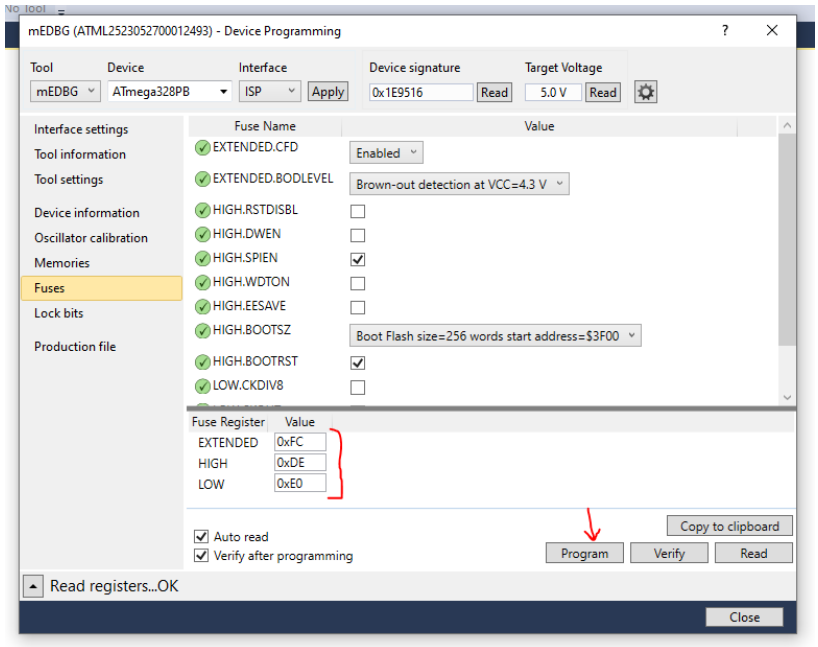

Figure: Device Programming Fuses window

3. Flashing the Bootloader

- Open the **Memories** menu in Microchip Studio.
- Select the downloaded MiniCore Bootloader HEX file:
  - `urboot_m328pb_2s_x16m0_19k2_uart0_rxd0_txd1_led+b5.hex`
- Click **Program** to flash the bootloader.
- Once completed, you can close **Microchip Studio**.

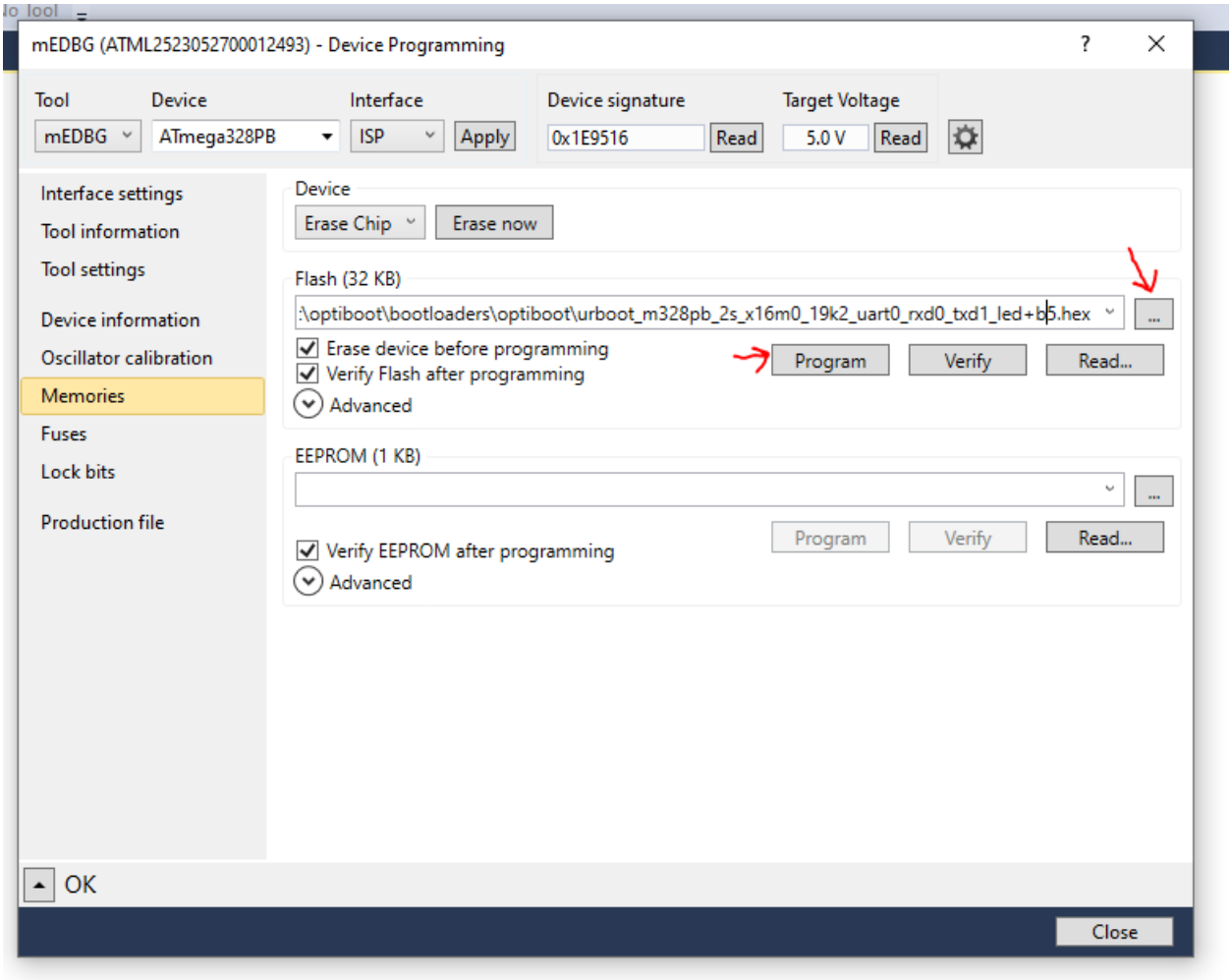

Figure: Device Programming Memories window

4. Setting Up the Arduino IDE

- Open the **Arduino IDE**, then go to **File > Preferences**.
- Under **Additional Board Manager URLs**, paste the following link:
- `https://mcudude.github.io/MiniCore/package_MCUdude_MiniCore_index.json`
- Click **OK**.

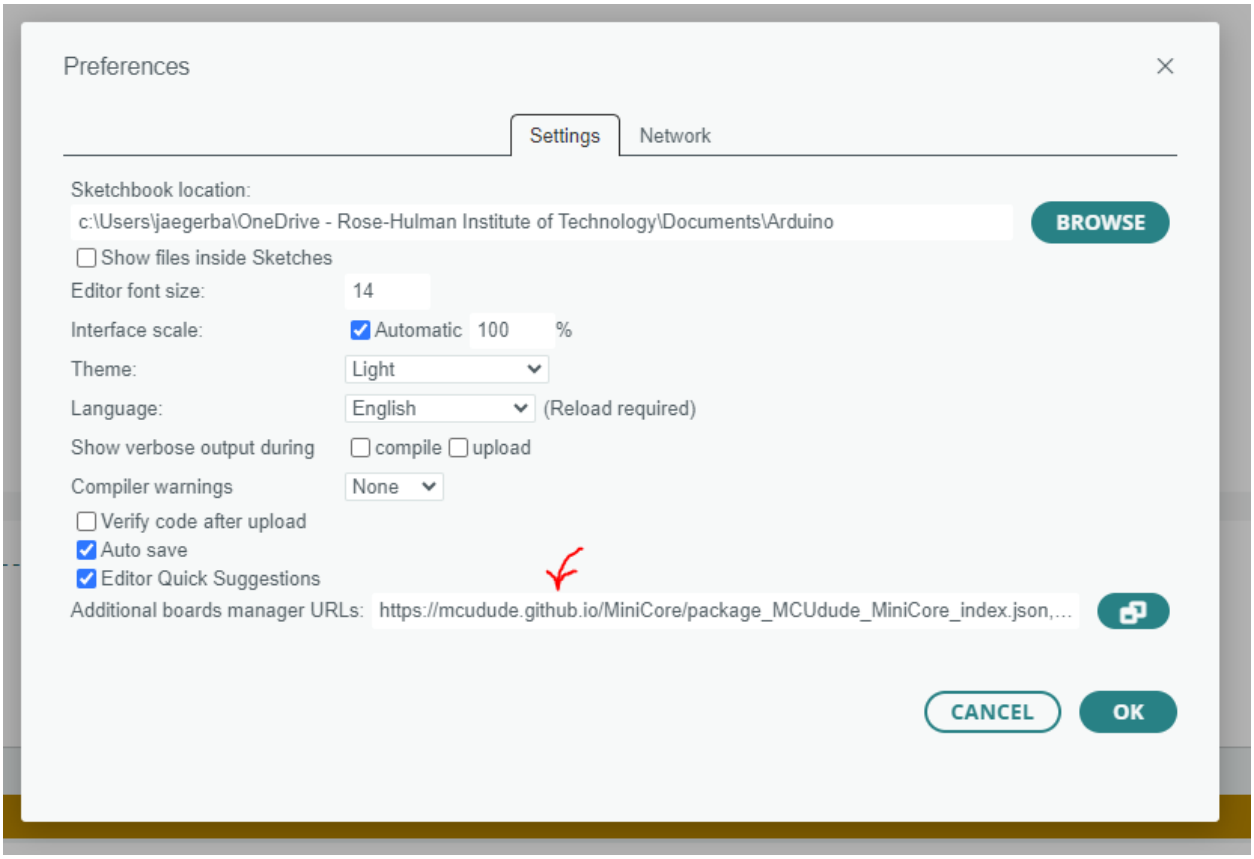

Figure: Arduino IDE Preferences

5. Installing MiniCore

- Open the **Board Manager** from the left menu.
- Search for **"MiniCore"**, then click **Install.**

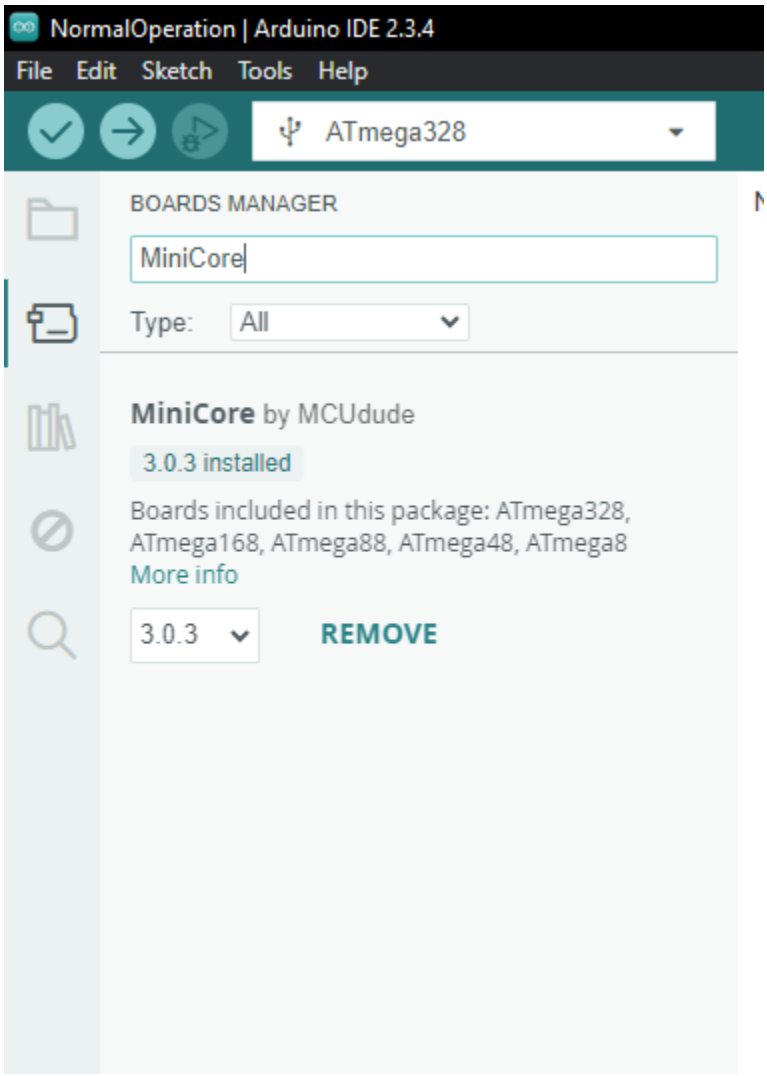

Figure: Arduino Boards Manager

6. Configuring Board Settings

Go to **Tools > Board** and set the following options:

- **Board:** MiniCore > "ATmega328PB"
- **Baud Rate:** Default
- **BOD:** 4.3V
- **Bootloader:** Yes (UART0)
- **Clock:** External 16 MHz
- **EEPROM:** Retained
- **Compiler LTO:** Disabled
- **Variant:** 328PB
- **Programmer:** Xplained Mini

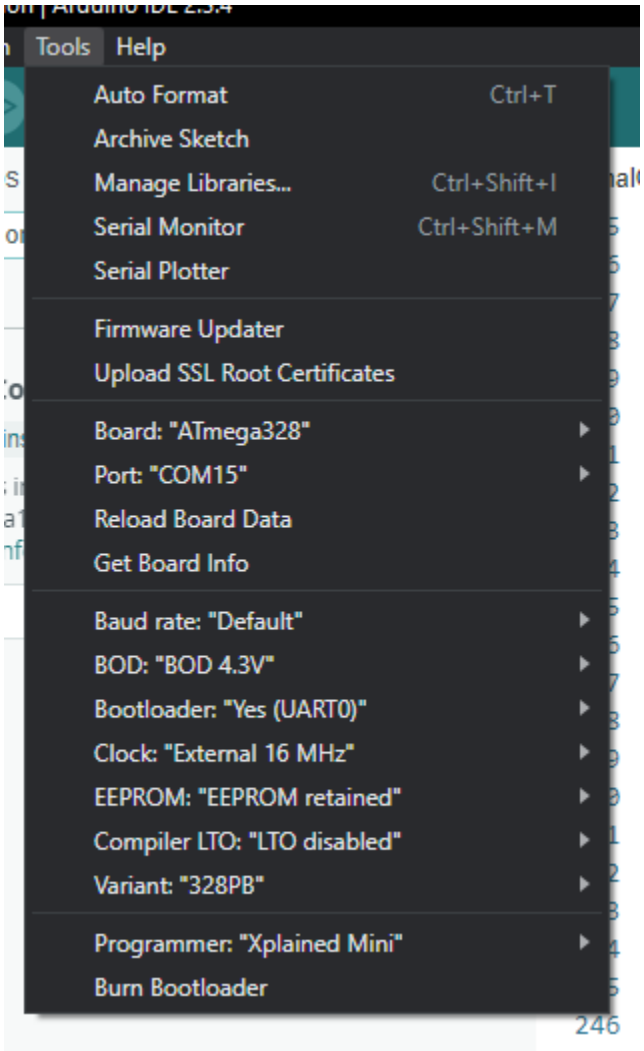

Figure: Arduino Tools Dropdown

7. Selecting Board and Port

- Click the **Board and Port** menu (USB symbol on the top bar).
- Select **ATmega328PB** as the board.
- Choose the correct **Port** that connects to the development board.

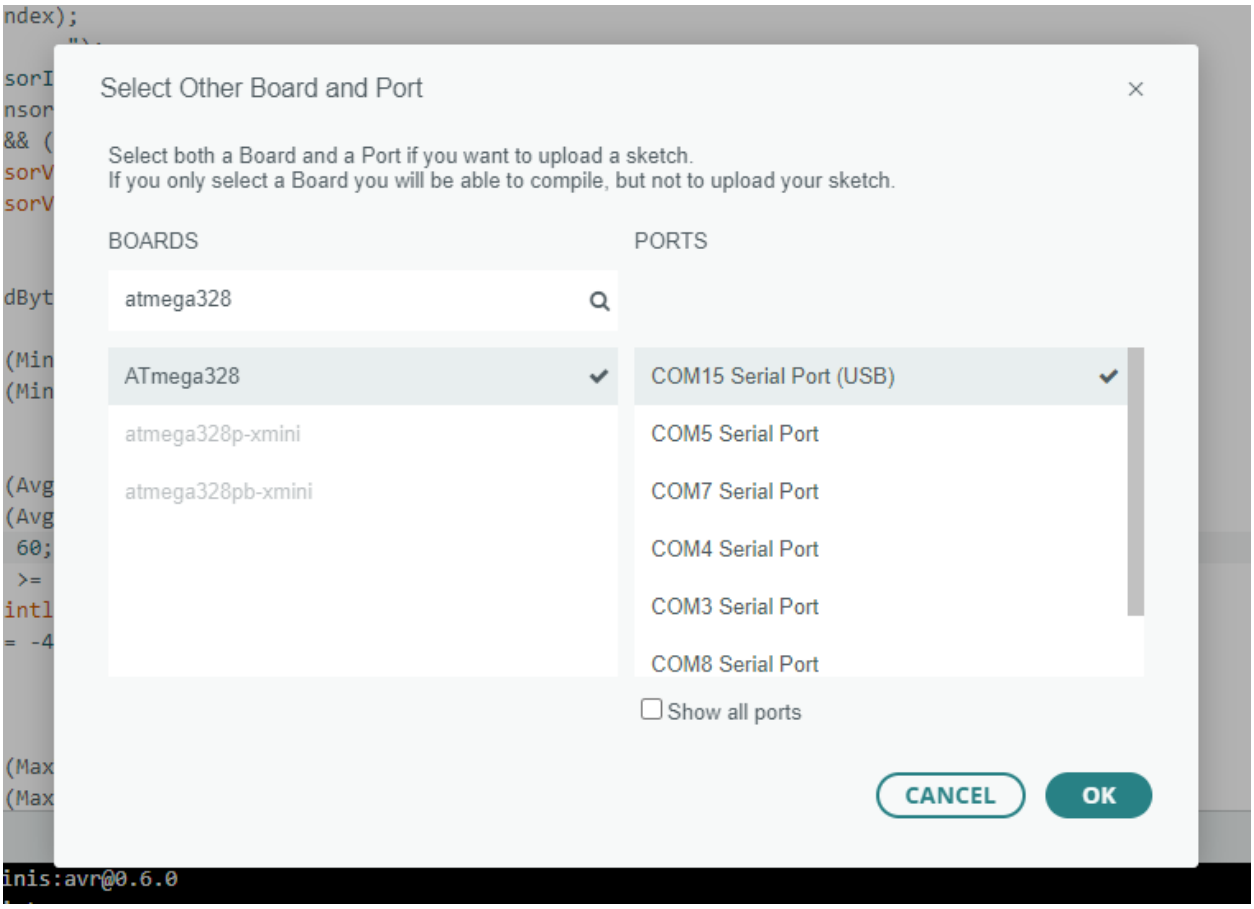

Figure: Arduino Board and Port menu

8. Uploading Code

- Open the desired code using **File > Open**.
- Upload it to the board using **Sketch > Upload Using Programmer**.

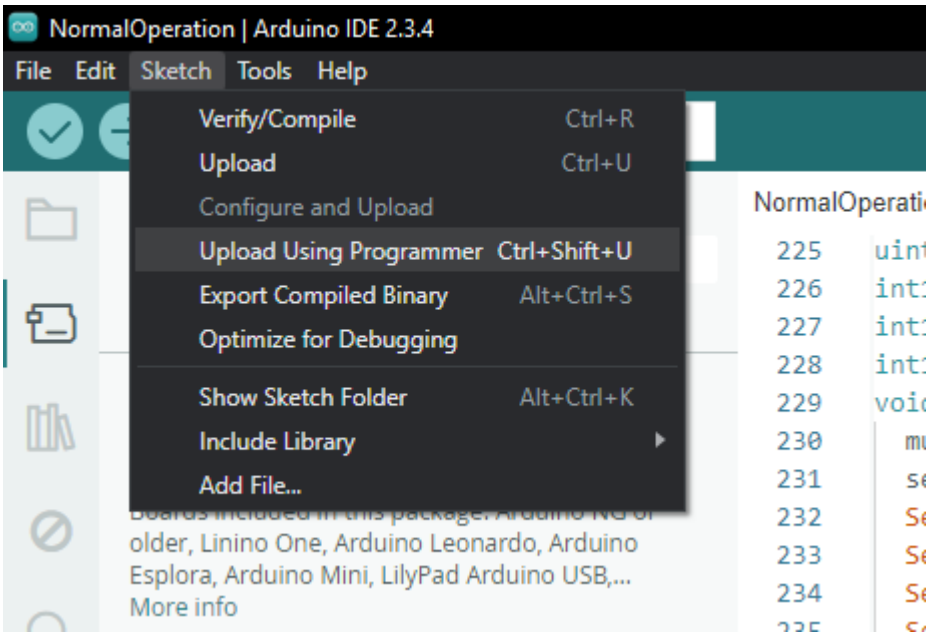

Figure: Sketch Dropdown menu

9. Final Steps

- The board should now be successfully programmed.
- You may see a warning message—this is expected and can be ignored.

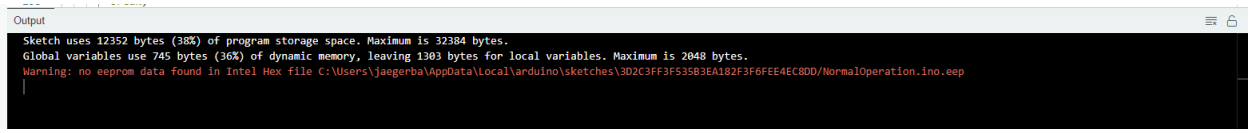

Figure: Arduino successful programming console message

3.3 Software Parameters

Parameters that can easily be modified to alter the operation of the code

connectedSensors: 8x4 array to account for the maximum of 32 sensors. For each sensor, enter a 0 if it is not active, otherwise enter the module that the sensor is associated with. Each process board can support up to 7 modules. Each instance of the system is only aware of the modules it is connected to so there will be multiple instances of modules 1, 2, 3, and 4 throughout the entire battery pack.

```
const uint8_t connectedSensors[8][4] = {
  { 1, 1, 1, 1 },
  { 1, 1, 2, 2 },
  { 2, 2, 2, 2 },
  { 3, 3, 3, 3 },
  { 3, 3, 4, 4 },
  { 4, 4, 4, 4 },
  { 0, 0, 0, 0 },
  { 0, 0, 0, 0 }
};
```

Figure: Example configuration for connectedSensors

WINDOW_SIZE: Configure the size of the sliding window used for digital filtering. The default value is 8. Refer to section 3.4 for more information.

WINDOW_TOLERANCE: Configure the tolerance of the sliding window calculation. The default is 25%. Refer to section 3.4 for more information.

SLAVE_ADDRESS: Set the Process boards I2C address.

3.4 Software Normal Operation

Upon startup the Atmega will automatically initialize both of its I2C buses. Next, all of the connected sensors will be initialized. Any errors that occur will be printed on the serial monitor.

The main loop will start collecting temperature data from the connected sensors. The code will iterate through each connected sensor and collect values from that sensor as determined by the value of WINDOW_SIZE. The average value will then be calculated, and each value will be compared against this average. Any values that are outside of the tolerance set by WINDOW_TOLERANCE will then be discarded and the average will then be recalculated with the new values. This new average will be stored for that sensor until it is recalculated. These values will be continuously updated while the system is running.

If the value for a specific sensor is requested, the last calculated average for that sensor will be sent. When the max, min, or average for a specific module is requested, that value will be found from the last sensor averages calculated and sent.

4. System Integration

4.1 Connecting the Process Board to Sensor Modules

Each sensor module connects to the process board through one of the Molex connectors to optimize space. Each Molex connector supports two temperature sensor modules.

4.1.1 Steps to Connect the Sensor Modules:

1. **Prepare the Molex Pigtail:**
    a. Take a male Molex pigtail, and solder two new leads onto each of the four existing wires, resulting in a total of eight leads.
    b. Use heat shrink tubing over all solder joints.

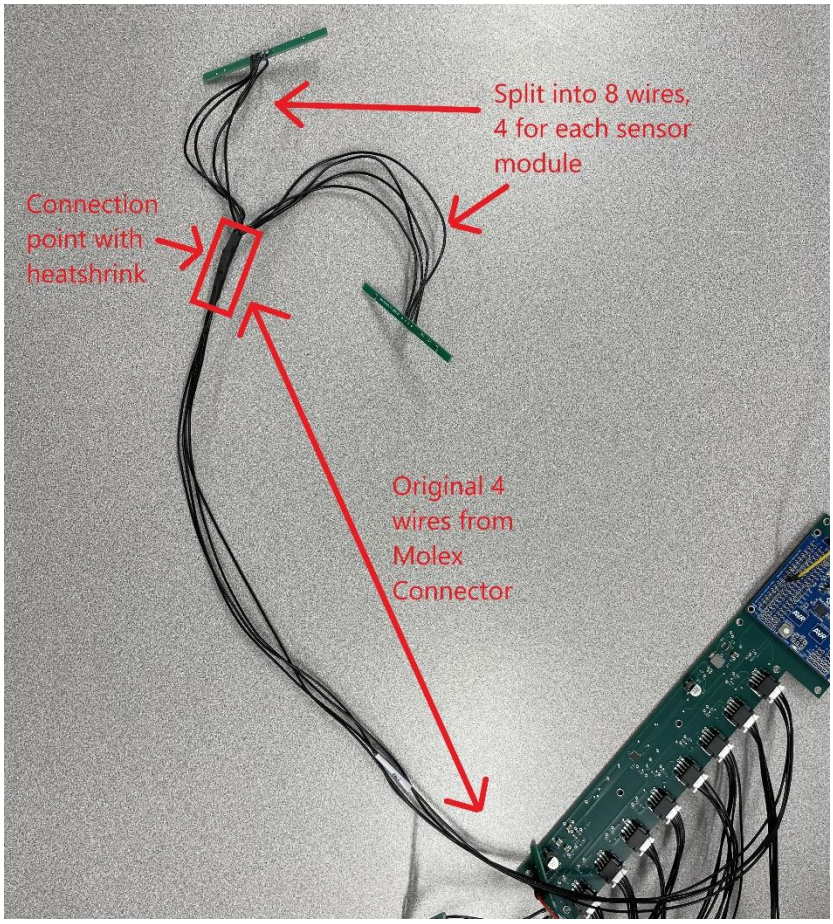

Figure: View of manufactured Molex Pigtail

2. **Establish Wire Connections:**
    a. As a standard, the Molex connector pin with a tag (or the leftmost pin when viewed from the top) corresponds to ground on the temperature sensor module boards. Refer to the figures below for guidance.

Figure: Wire on the Molex Connector that corresponds to GND

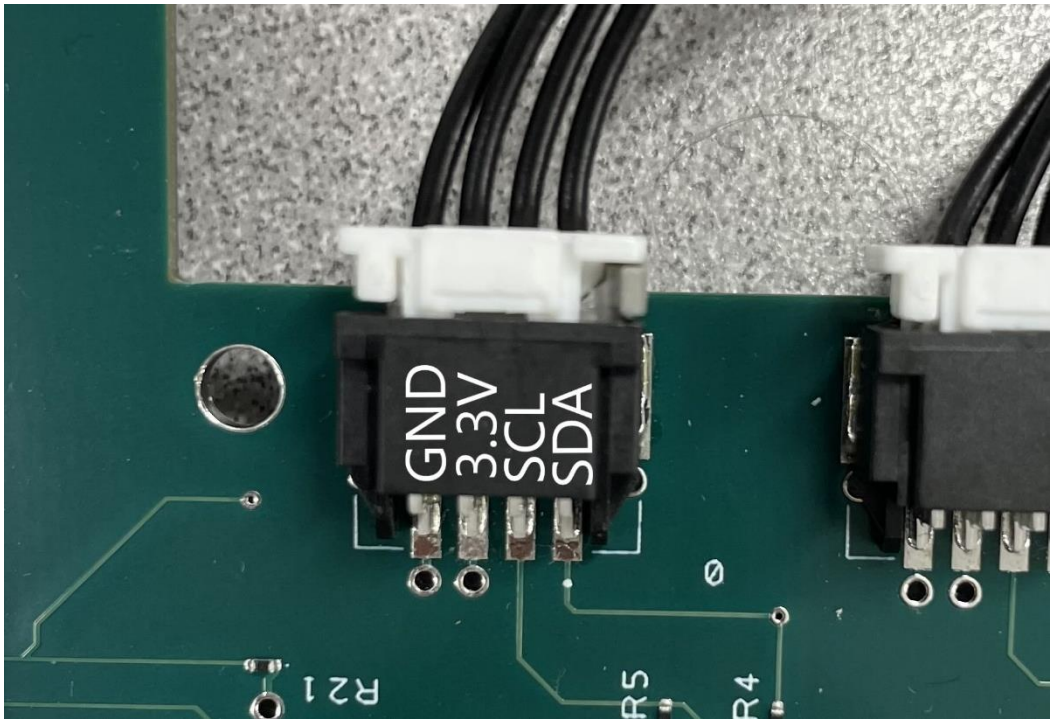

Figure: View of connections on Molex Connector

    b. Solder the new leads to the sensor modules. Utilize the figures below for guidance.

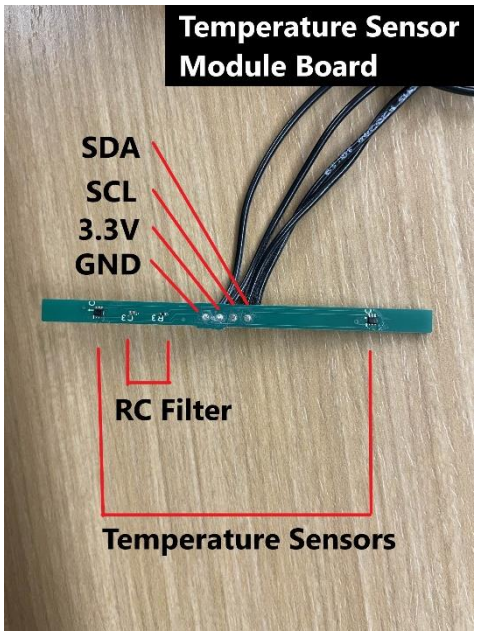

Figure: Subsection view of the Temp Sensor Board

Figure: View from behind of a soldered temperature sensor module

4.2 attaching sensor Module to Cell

The sensor modules are attached to the battery cell using a Thermal Interface Pad (TIP). This pad functions like double-sided tape but has enhanced thermal conductivity, allowing for more efficient temperature sensing.

4.2.1 Application Process:

1. **Preparing the Thermal Interface Pad:**
    a. Cut the TIP to approximately 3 mm x 3 mm for each of the TMP112 temperature sensors to ensure proper coverage.
    b. Ensure the pad is cut cleanly for optimal adhesion and thermal transfer.
2. **Applying the Thermal Interface Pad:**
    a. Place the cut TIP on the top of the case over both TMP112 temperature sensors on the module.
    b. Ensure the pad is securely adhered to the sensor surface.

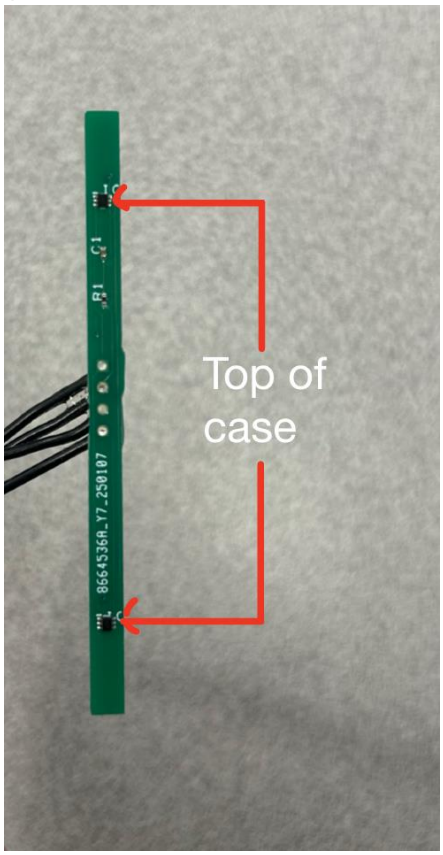

Figure: Image of the front face of the temperature sensor module

3. **Mounting the Sensor Module to the Cell:**
    a. Press the sensor module (with the TIP applied) firmly against the cell.
    b. Secure it in place using zip ties to maintain consistent contact while the adhesive sets.
    c. Allow 24 hours for the adhesive to fully cure before removing the zip ties.

[Insert image of completed sensor-to-cell application]

4.3 Configuring the I2C Communication

4.3.1 Overview of I2C Communication

The Temperature Sensing and Data Delivery system communicates with the Battery Management System (BMS) via an I2C bus, enabling real-time temperature monitoring of the EV battery pack. The microcontroller (ATMega328PB) functions as the I2C slave, responding to requests from the BMS, which acts as the I2C master.

The system supports up to 32 temperature sensors, with data aggregated by the process board before being transmitted over I2C. Temperature readings are made available through a structured register map, allowing the BMS to efficiently poll and retrieve data.

4.3.1 Hardware Connections

Figure: Process board I2C pinout

The Temperature Sensing and Data Delivery system operates with 3.3V logic levels and interfaces with the Battery Management System (BMS) over an I2C bus.

Required Connections:

- SDA (Data Line): Connect to BMS SDA
- SCL (Clock Line): Connect to BMS SCL
- Shared ground or cell negative between process board and BMS.

External I2C Pull-up Resistors:

We highly recommend installing 4.7kΩ pull-up resistors on the SDA and SCL lines.

Additional Debug Connections:

Mux SDA Debug & Mux SCL Debug are available but are not required for standard operation. These pins provide access to the secondary I2C channel used for communication with the sensor multiplexer and are intended for debugging purposes only.

4.3.1 Addressing and Device Selection

The Temperature Sensing and Data Delivery system operates as an I2C slave device with a default address of 0x20. If the address needs to be changed, refer to Section 3.3 - Software Parameters.

To read temperature data, the master must first send a command byte specifying the desired sensor or system metric.

Single Sensor Command Byte Structure:
Bits [7:4] – Multiplexer (MUX) index (0–7)
Bits [3:0] – Sensor index within the selected MUX (0–3)

Example:
If requesting data from MUX port 3 and sensor 2, the command byte is: 0x32 (MUX = 3, Sensor = 2)

Entire Cell Stack Module Command Values:

Bits [7:4] – Cell Stack Module value (9–F) {Module index + 0x9}
Bits [3:0] – Data To request (0-2) {0: Minimum, 1: Average, 2: Maximum}

Example:
If requesting data from Cell Stack Module 2 and you want the Maximum, the command byte is: 0xA2 (Module index = 2, Maximum = 2)

If the command byte specifies an invalid MUX or sensor index or an invalid Cell Stack Module Command, the device will not return valid data.

4.3.1 Reading Data from the Sensor

To read temperature data, the BMS master must:

1. Send a command byte to specify the desired sensor or system metric.
2. Read two bytes from the slave device.
3. Combine the bytes into a 16-bit signed integer.
4. Scale the value by multiplying by 0.0625 to convert to degrees Celsius.

A python example is available here: https://github.com/rhit-jaegerba/Temperature-Sensing-and-Data-Delivery-for-EV-Battery-Management/blob/main/DemoPlot.py

5. Operation Guide

5.1 Powering the System

Refer to the technical specifications section in the appendix for the exact voltage range the process board can accept. Labels near the edge of the process board indicate the correct voltage source connections (“Cell Stack +” and “Cell Stack -”).

5.1.1 Special Notes for Battery Workforce Challenge

We recommend that the Battery Workforce Challenge Team use a single Cell Stack Module (5 Cells in series gives a voltage range of 15-20.75V) to power the process board. If the Battery Workforce Challenge Team chooses to use an ADBMS6830 alongside a process board to monitor multiple Cell Stack Modules, careful consideration must be given to the overall power connections. The Process Board and the ADBMS6830 must both utilize the same ground for safe operation. Ground refers to the point that has the lowest voltage among the Cell Stack Modules connected in this case. For 3 Cell Stack Modules (15 Cells in series), ground would be the negative terminal of the 15th Cell. Refer to the GitHub link below for guidance and examples.

https://github.com/rhit-jaegerba/Temperature-Sensing-and-Data-Delivery-for-EV-Battery-Management/tree/main/Power_Grounding_Diagrams

5.2 Programming the System

The Atmega328PB includes a jumper that allows programming via the micro-USB port (With no jumper the Atmega328PB is in the program position by default). Refer to the figures below for guidance on the correct jumper position.

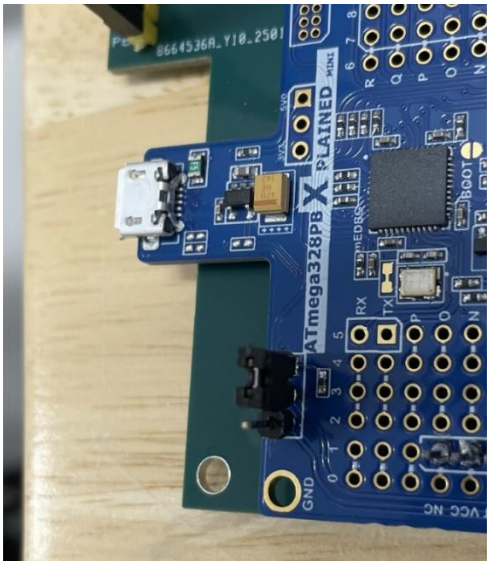

Figure: Atmega328PB jumper in program position

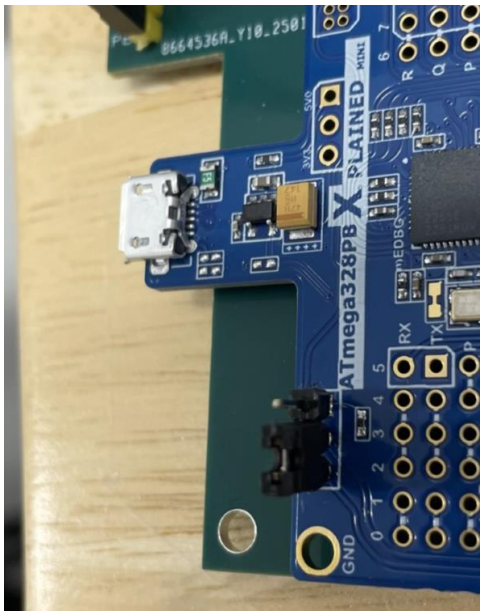

Figure: Atmega328PB jumper in power position

Make sure the jumper is set to the program position, then follow the steps below in the Arduino IDE to upload the correct code to the Atmega328PB.

- Open the NORMAL_FILE using **File > Open**.
- Upload it to the board using **Sketch > Upload Using Programmer**.

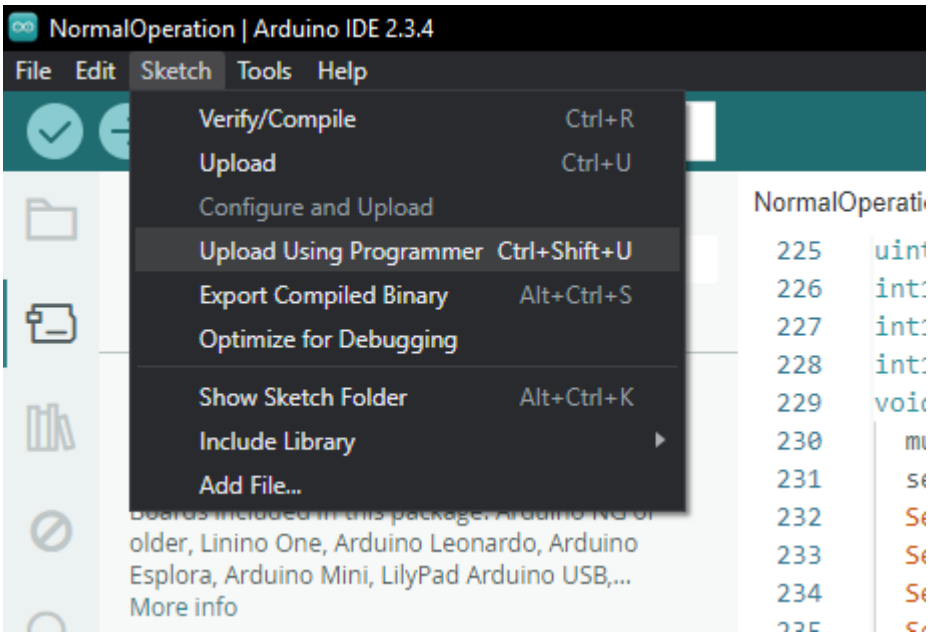

Figure: Sketch Dropdown menu

- The board should now be successfully programmed.
- You may see a warning message—this is expected and can be ignored.

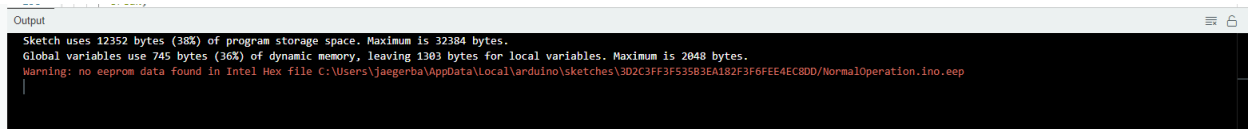

Figure: Arduino successful programming console message

After successfully programming the Atmega328PB, move the jumper to the power position.

5.3 System Setup Successful

The system will function properly if the steps in sections 5.1 and 5.2 are followed. For guidance on system integration, refer to section 4 of this document.

6. Appendix

6.1 Technical Specifications

The table below summarizes the key operating parameters for the Temperature Sensing and Data Delivery system. The "Recommended Operating Features" column lists the conditions under which the system is designed to perform optimally. The "Absolute Maximum" column shows the limits beyond which the system may become unreliable or sustain damage.

Note: These values are based on design targets and sensor specifications. For best performance and longevity, always operate within the recommended range.

Parameter	Recommended Operating Conditions	Absolute Maximum	Notes
Process Board Input Voltage ("Cell Stack Connections", see Hardware Section for location on board)	15.0 V to 20.75 V	4V Minimum 36V Maximum	The system's power circuitry is designed to operate from the voltage produced by a stack of 5 battery cells in series. The Abs Max/Min values come from the switching regulator Datasheet.
External I2C Communication Characteristics ("I2C Connections", see Hardware Section for location on board)	Refer to the Atmega328pb Datasheet (Table 33-9): https://ww1.microchip.com/downloads/en/DeviceDoc/40001906C.pdf VCC=5V	Refer to the Atmega328pb Datasheet (Table 33-9): https://ww1.microchip.com/downloads/en/DeviceDoc/40001906C.pdf VCC=5V	Utilizing the external I2C ports does not require pull-up resistors.
External I2C Communication Speed	100kHz	Up to 400 kHz fast-mode	N/A
Temperature Sensor Accuracy (From –40 C to 125 C)	N/A	±2°C	This is the BWC's requirement for the system and has been verified to by testing through the range of -40°C to 125°C. Temperature values (Avg) obtained from I2C communication

			with our device can be expected to be within ±2°C of the actual temperature in the environment being measured.
System Temperature Range	0°C to 60°C	-40°C to 125°C	This is based off the rating AECQ100/200 rating of all components in the system.
System Temperature Resolution	N/A	0.0625 °C	
Individual Temperature Sensor Polling Rate	Up to 50 Hz	Up to 50 Hz	A 50 Hz rate is optimal for accurate data filtering; higher rates may be used in test modes but could overwhelm the I2C bus or processing routines.
Power Consumption	Active mode: Under 1W	Active mode: Under 3W	Operating within recommended power limits is critical to prevent overheating and preserve battery efficiency.

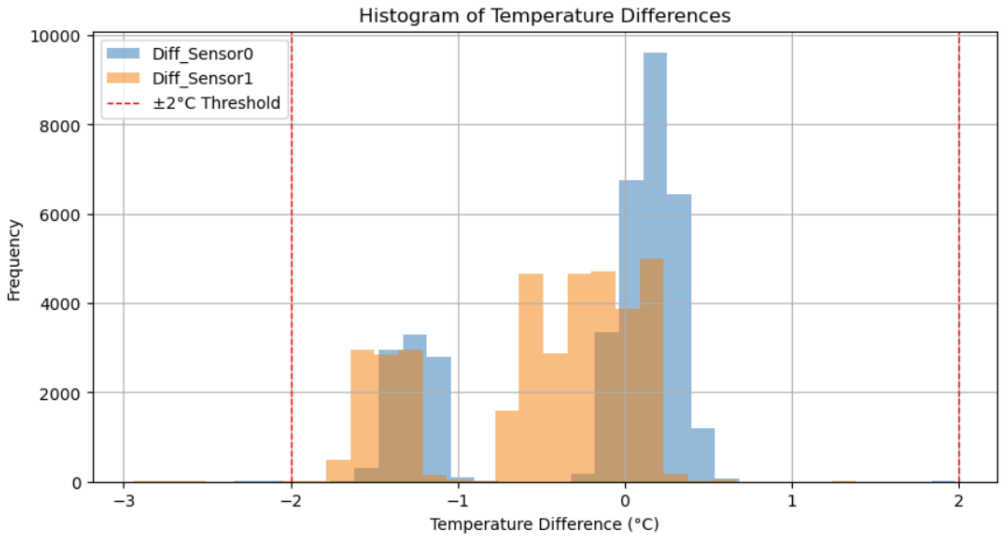

Figure: Absolute difference in temperature between a temperature sensor and the environment

6.2 Reference Diagrams and Schematics

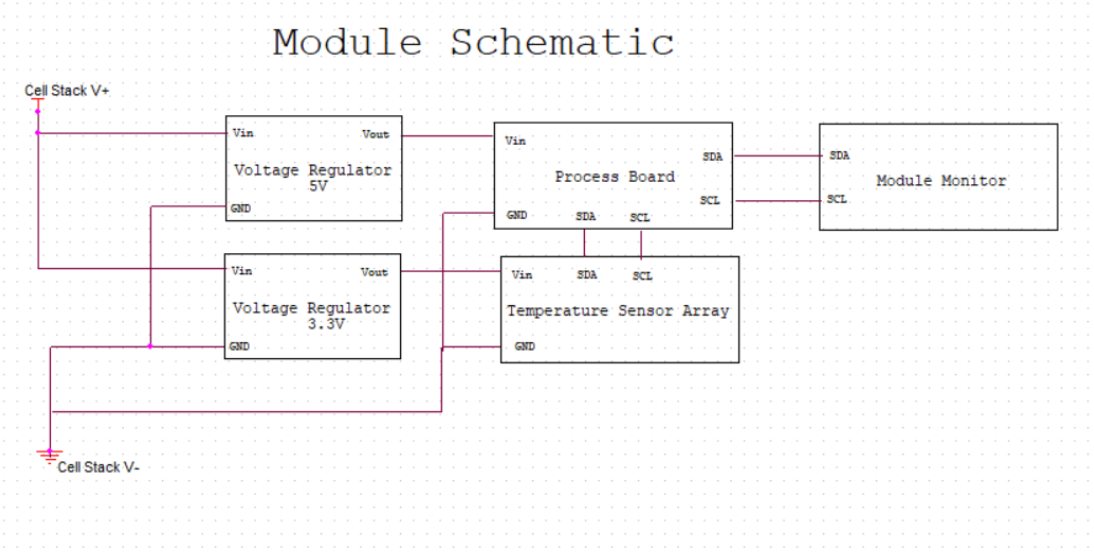

Figure: Overall circuit diagram of system

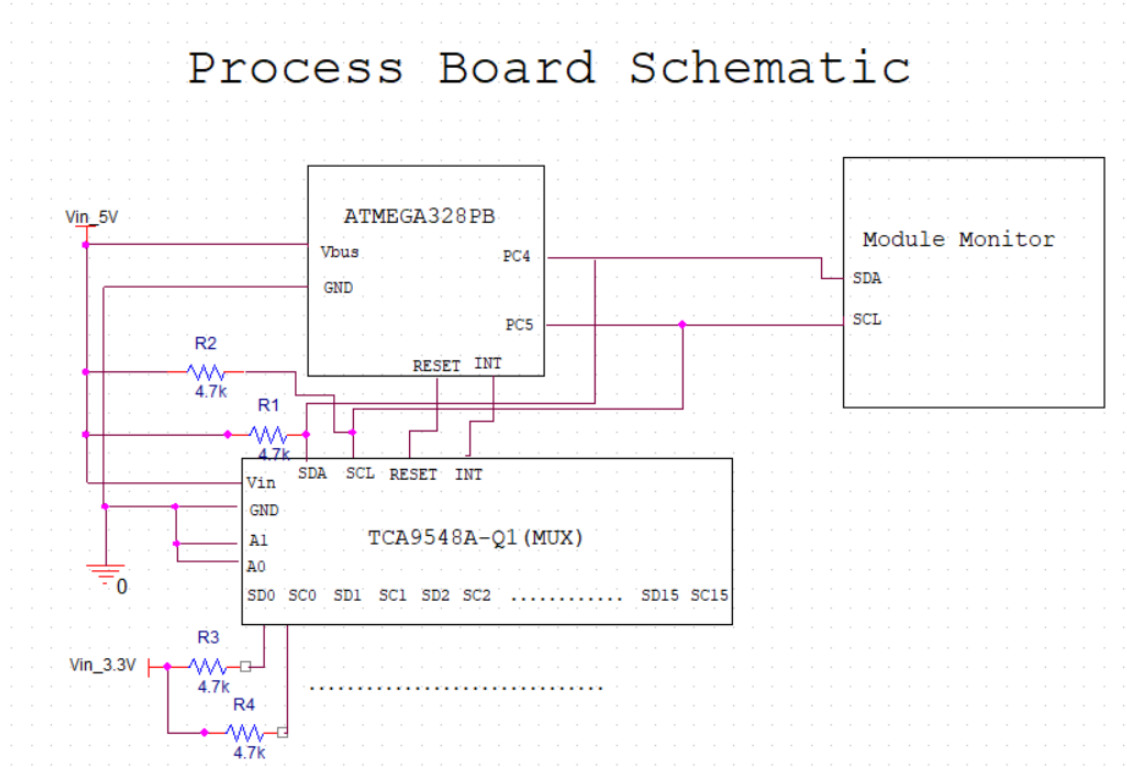

Figure: Diagram of connections between ATMega328PB, Mux and ADBMS6830

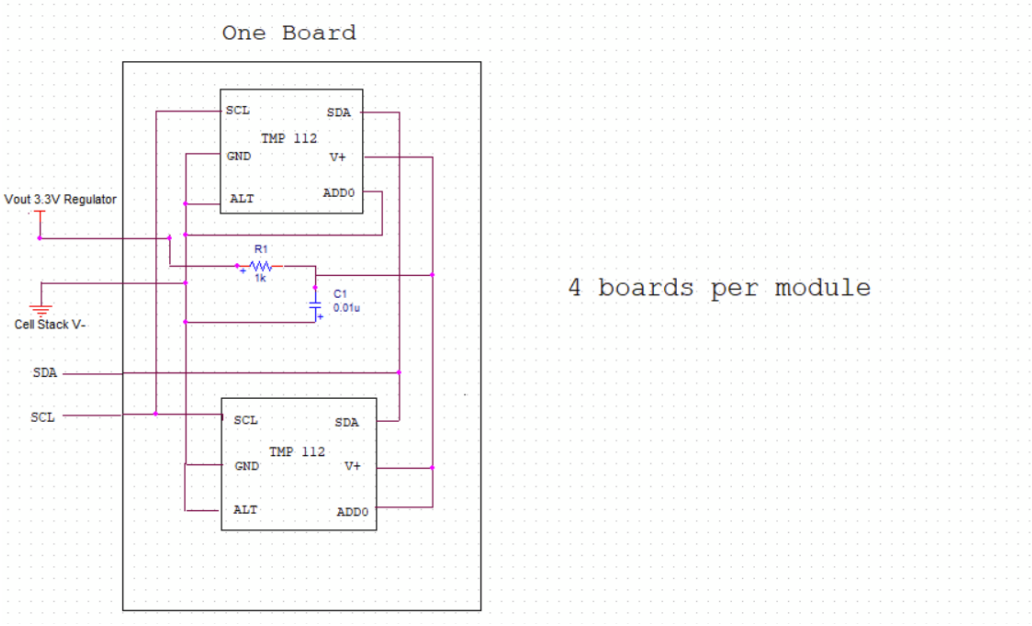

Figure: Circuit Diagram of one temperature sensor board (Two temperature sensors).

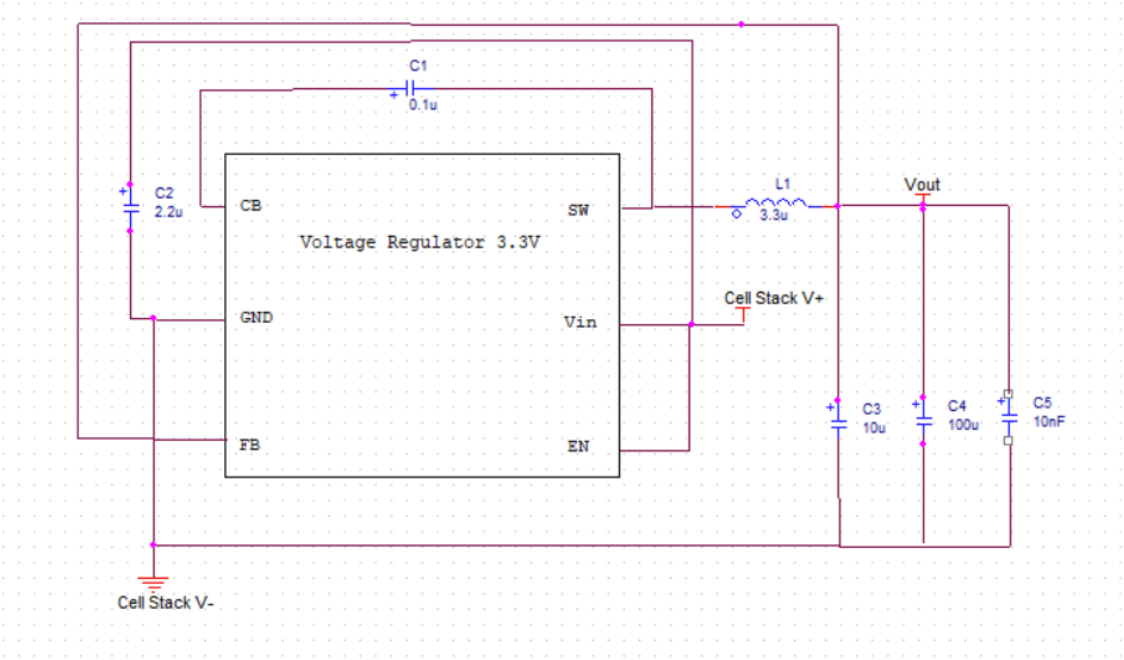

Figure: Diagram of the 3.3V switching voltage regulator circuit.

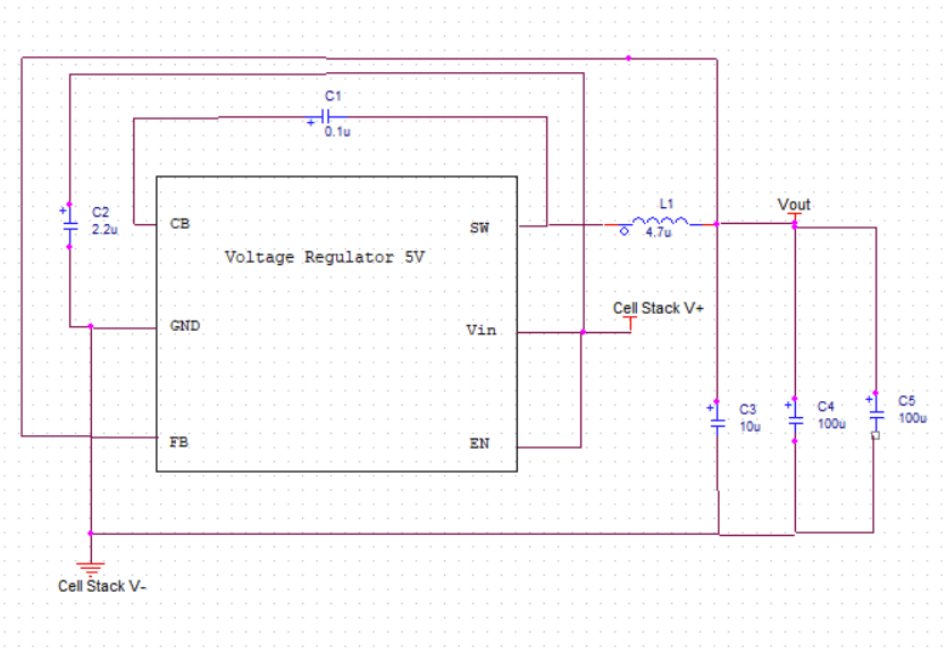

Figure: Diagram of the 5V switching voltage regulator circuit.

6.3 Fabrication & Assembly

6.3.1 Bill of Materials

Before beginning fabrication and assembly, ensure all materials listed in the bill of materials are acquired. The bill of materials includes the exact number of components required to make each board (Process Board and Sensor Module). It is advisable to order spare components, as manual fabrication is not always perfect, and parts may be lost. For expensive components, having one or two extras is recommended, while for cheaper components, keeping an additional ten to twenty is a good precaution.

Bill of Materials:

https://github.com/rhit-jaegerba/Temperature-Sensing-and-Data-Delivery-for-EV-Battery-Management/tree/main/Manufacturing

6.3.2 Fabrication & Assembly Process

For efficient fabrication and assembly of the PCBs used in this product, it is highly recommended to order and use a stencil. A stencil greatly accelerates the manufacturing process. This section of the guide explains the fabrication process with the assumption that a stencil has been obtained for each PCB (Process Board PCB and Sensor Module PCB). Additionally, a microscope is essential for

PCB fabrication and assembly, as many components are extremely small and difficult to place accurately.

6.3.2.1 Applying Solder Paste

Place the PCB on a flat surface and align the stencil so that its holes match the pads on the board. Secure the stencil with tape to prevent movement during solder paste application. Once secured, apply solder paste on top of the stencil and spread it evenly to fill all the holes. Verify that solder paste has been applied to every hole, then carefully lift the stencil and proceed with component placement.

6.3.2.2 Placing Components

Place the PCB under a microscope for better visibility during component placement. While components can be placed in any order, it is recommended to start with the smallest ones first to avoid obstruction from larger components. Use the board's schematics to ensure correct placement, referencing the silkscreen markings (e.g., C05 on the board corresponds to C05 on the schematic). Handle small components carefully, as they can be easily lost. Pay close attention to components with polarity and ensure they are oriented correctly. Additionally, some components must be placed in a specific orientation—silkscreen indicators and datasheets can provide guidance. Continue placing components in their designated locations until all are correctly positioned.

6.3.2.3 Reflow Oven

After placing all components, the PCB is ready for the reflow oven. The oven's temperature settings depend on the type of solder paste used, so refer to the solder paste specifications and the reflow oven manual for the appropriate temperature and duration. Once the reflow process is complete, visually inspect the board to confirm that all components are properly attached and securely soldered to the pads.

6.3.2.4 Atmega328PB

Before mounting the Atmega328PB onto the process board, solder a pin header in the designated area shown in the figure below. This header will serve as a hardware control, allowing the Atmega328PB to operate using either board power or micro-USB power.

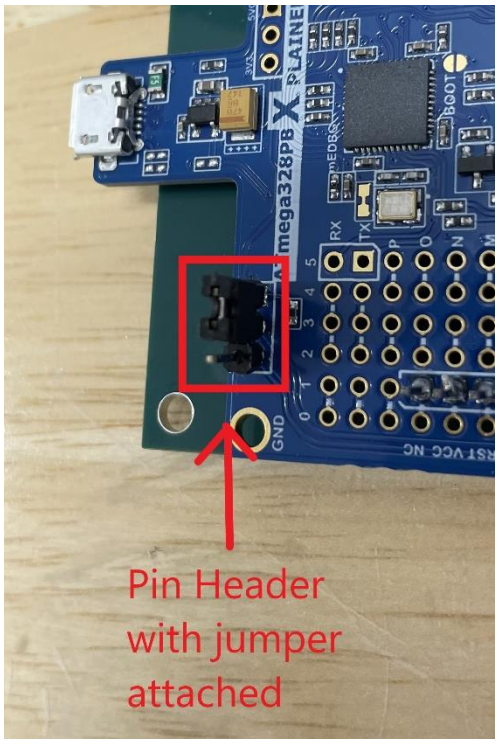

Figure: View of soldered pin header with jumper attached

The Atmega328PB is mounted using pin headers and through-hole connections on the process board, with all connections soldered by hand. Every though-hole pin on the process board PCB should be filled with a pin header. Ensure proper alignment by referring to the pictures below.

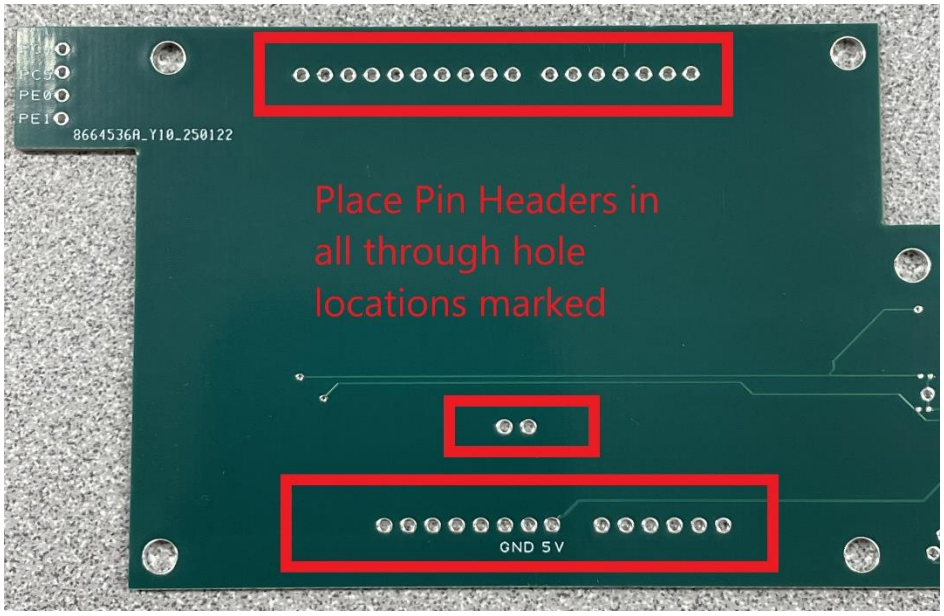

Figure: Empty process board

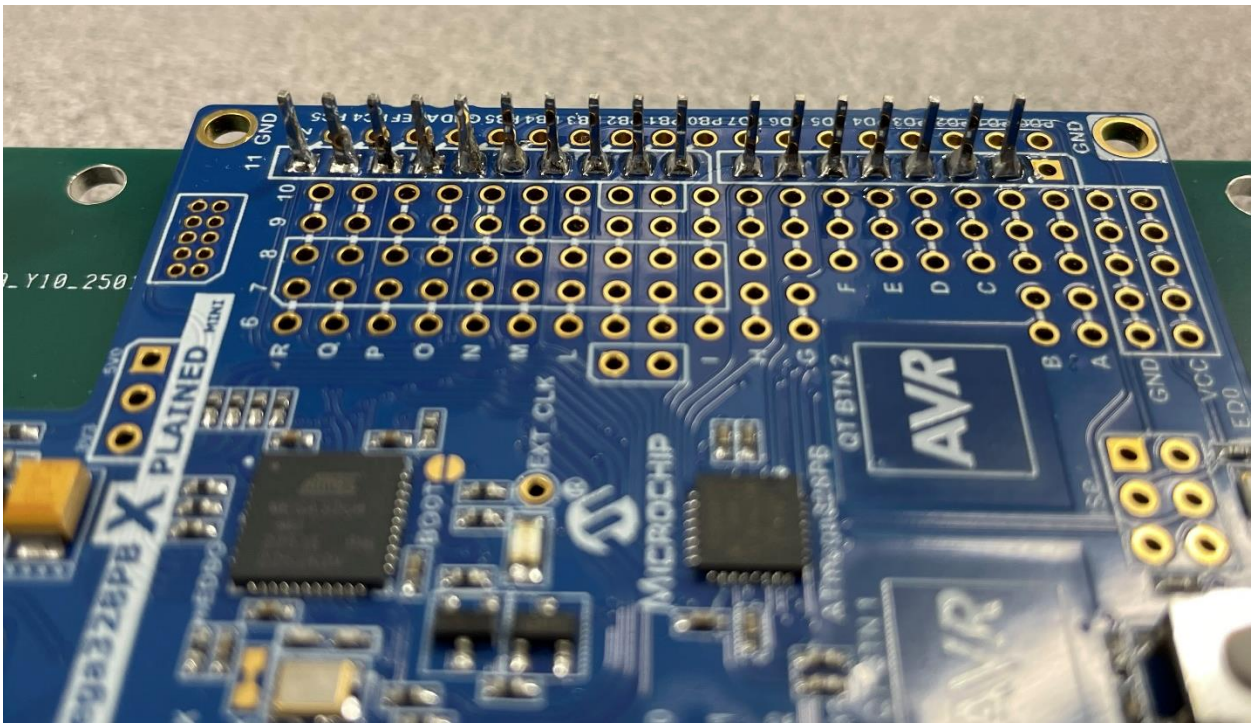

Figure: Pin Header Top View

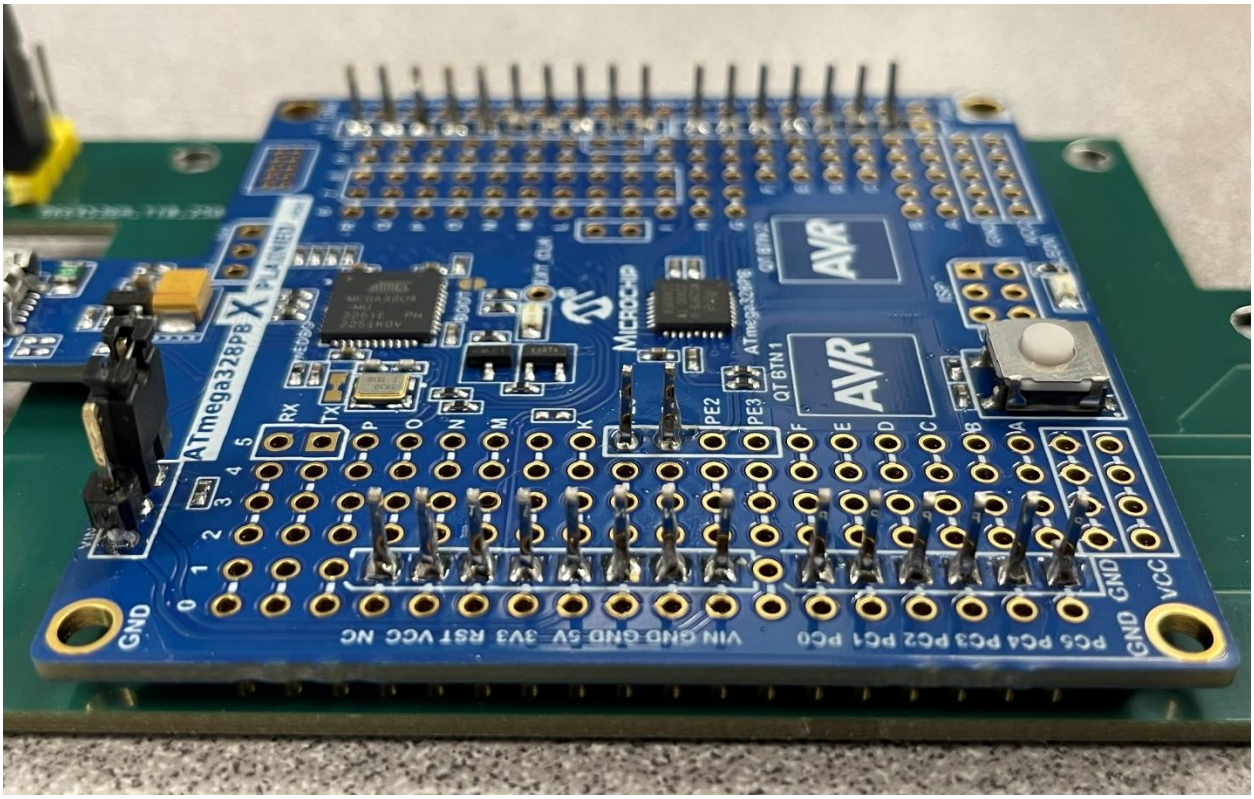

Figure: Pin Header Bottom View

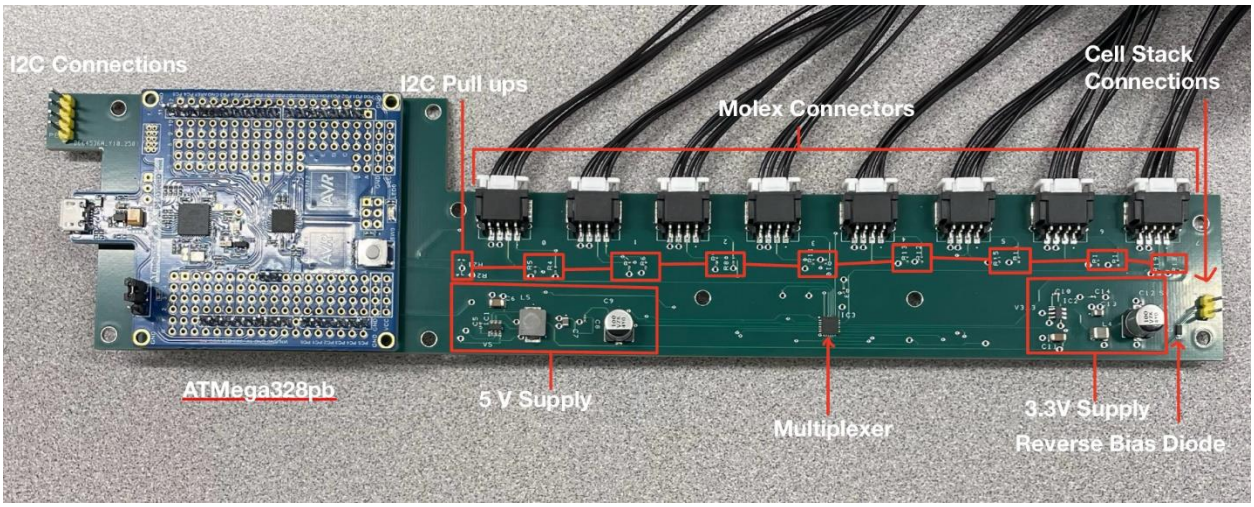

Figure: Subsection view of the process board

6.4 Glossary of Terms

- BMS: Battery Management System
- I2C: Inter-Integrated Circuit communication protocol
- SMT: Surface-Mount Technology
- AEC-Q100/Q200: Automotive Electronics Council standards for component reliability
- Cell Stack Module: A module within the battery pack, containing cells, designed by the Battery Workforce Challenge.
- ADBMS6830: A device that monitors the voltages of cells in a Cell Stack Module