Fall 2021

Expanding Temperature Sensing for the Orion BMS 2

Samuel J. Parker
Portland State University

Expanding Temperature Sensing for the Orion BMS 2

By Samuel Parker

An undergraduate honors thesis submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

University Honors

and

Computer Science

Thesis Adviser

Dr. Bart Massey

Portland State University

2021

Table of Contents

Acknowledgments

The development of this project was severely hindered by the COVID-19 pandemic. Completion was only possible through the help of my advisor, Dr. Bart Massey, and countless hours with the Viking Motorsports Electronics lead, Braeden Hamson.

I also owe a debt of gratitude to the Viking Motorsports team for the opportunity to work with an amazing group of people; Mark Morrissey, for his continued professional and academic support; my parents, John and Jennifer Parker, for inspiring me to complete this degree; and Brianna and Jasper Avery, for their dedication to the students of Portland State University's University Honors College.

Abstract

Formula SAE (FSAE) is an annual collegiate design competition that takes place across the globe. Portland State University's team, Viking Motorsports, was committed to designing an Electric Vehicle (EV) for the 2021 FSAE competition. The team designed a completely custom lithium-ion cell battery that is managed by an Orion BMS 2 battery management system. The FSAE rulebook requires a robust temperature monitoring system for any EV power supply. The Orion BMS 2 can only directly collect data from eight temperature sensors, which is not enough to meet FSAE regulation. However, the BMS can be configured to monitor many more sensors through the use of the Controller-Area Network (CAN) protocol. This thesis documents the creation of modules capable of collecting and transmitting data from hundreds of temperature sensors via CAN. Software and hardware designs were driven by requirements established at the start of development. Once complete, modules went through a series of tests to ensure all requirements were met.

Introduction

This thesis documents the development of temperature-sensing modules capable of collecting and transmitting data from hundreds of temperature sensors for a Formula SAE Electric Vehicle. These modules were developed from November of 2020 through April of 2021, using in-house software and circuit boards and off-the-shelf microcontroller units. Modules were required to pass a test plan to ensure they met all safety and regulatory requirements.

Formula SAE

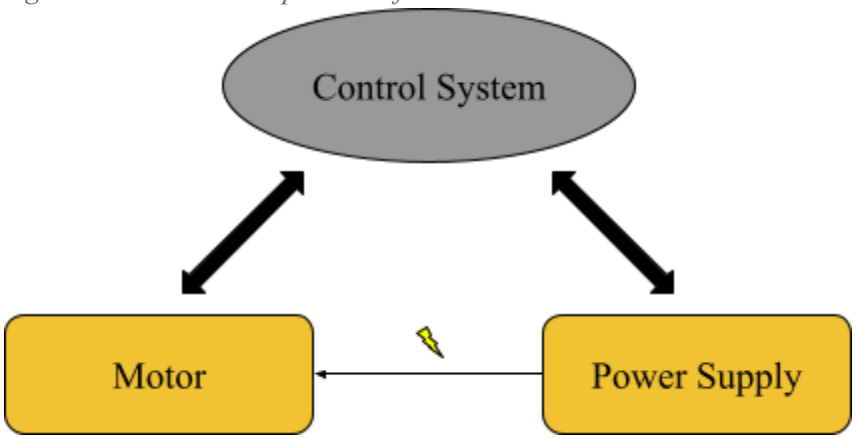
In the early 1980s, the Society of Automotive Engineers (SAE) created a collegiate-level design competition called Formula SAE (FSAE). The competition began with only six participating schools, but has grown to nearly 400 universities around the globe. Teams are instructed to design and manufacture a formula-style race car as if they were contracted by a real-world manufacturing company. Teams then compete in a series of eight events that are broken into two categories: static and dynamic. The three static events focus on the production of the car: teams are evaluated on design, costs, manufacturing methods, and general presentation of the vehicle. The teams must then defend their design decisions to a panel of judges consisting of industry professionals. The dynamic events, on the other hand, focus on the physical performance of the vehicle. Teams participate in rigorous driving scenarios, including timed racing events and G-force tests, to evaluate handling, endurance, acceleration, and grip. Teams must abide by a strict rulebook that is typically revised every year. Traditionally, FSAE teams have designed Internal Combustion (IC) cars for the competition. In recent years, however, it has become more popular to compete with Electric Vehicles (EVs).

Viking Motorsports (VMS) is Portland State University's FSAE team. The team set out to build an EV for the 2021 competition. The team historically competed with an IC car, but transitioned to an EV in 2014. After years of struggling to complete a car, and with rising pressure from the university to compete, the team went through a significant organizational restructuring in late 2019. New leadership roles were established, and the team was broken into the following subsystem teams: Chassis, Suspension, Electronics, Electronic Drive-Train, Business/Sponsors, and Manufacturing. Although COVID-19 restrictions limited progress significantly due to the lack of access to the team's lab, tools, and other resources, the team made significant progress through 2020. Unfortunately, the car was not completed in time for competition, but became a solid foundation to improve upon for the coming years.

Monitoring the VMS Power Supply

To understand the work reported in this thesis, you must first understand the main components of an EV, as shown in Figure 1. The power supply is typically a large battery that supplies electrical energy to the motor. The motor then transforms that energy into mechanical energy that drives the vehicle's wheels. Meanwhile, the control system is responsible for monitoring these components to ensure effective operation and managing issues as they arise.

Figure 1: The main components of an Electric Vehicle



Over the last two years, VMS has been designing a completely custom power supply for the team's EV. The team's battery consists of six segments containing 144 lithium-ion cells each: 864 cells in total. An Orion BMS 2, an off-the-shelf Battery Management System (BMS), monitors the battery's state through a series of sensors, including temperature probes and voltage taps.

A custom temperature-reporting system needed to be developed for the Orion BMS 2. From the factory, the BMS allows eight directly-connected temperature sensors at a time, but this is not enough to meet FSAE regulations. Orion provides an off-the-shelf Thermistor Expansion Module (TEM) which is capable of monitoring extra sensors; however this module is too expensive to fit the team's budget. This thesis documents the development of custom TEMs by

1. Outlining requirements introduced by the FSAE Rulebook and the Orion BMS 2
2. Explaining how the hardware and software designs meet those requirements
3. Reporting results from a test plan motivated by safety and those requirements
4. Reflecting on accomplishments, complications, and future work for the project

Requirements

In order to ensure effective and safe functionality of the TEMs, they must fulfill the requirements set forth by the FSAE Rulebook and the Orion BMS 2 interface specification.

The FSAE Rulebook

Safety is the foremost priority when designing FSAE vehicles. The consequences of poor design could result in severe injury or even death of the vehicle's driver. Fortunately, FSAE provides a strict rulebook to safely guide the development of teams' cars. Sections EV 8.3 and EV 8.5 of the 2021 FSAE Rulebook [1] include the following rules relevant to this project:

- EV 8.3.4: If the BMS detects an error (including temperature values outside the allowable range), the Shutdown Circuit and BMS Indicator Light must engage
- EV 8.5.2: Cell temperatures must remain below 60℃
- EV 8.5.5: 20% of all lithium-based cells must be monitored (The team decided to increase that proportion to 25% of all cells, for 216 sensors total, to improve error detection and reliability of data)

The Orion BMS 2

The Orion BMS 2 requires TEMs to communicate with it via the Controller Area Network (CAN) protocol. CAN is a network protocol developed in the early 1990s that is fast and highly resistant to noise, making it one of the most popular automotive communication

methods even today [2]. The Orion BMS 2 expects to receive CAN messages every 100ms containing the following information [3]:

- Average temperature of the cells
- Highest recorded temperature
- Lowest recorded temperature
- Number of failed sensors
- An ID for the transmitting module

If temperatures exceed a configurable threshold (in the case of this project 60℃ per EV 8.5.2) or messages are delayed for longer than 100ms, the BMS engages the Shutdown Circuit and the BMS Indicator Light per EV 8.3.4.

Design

Decisions regarding the software and hardware were guided by the requirements outlined in the previous section. These decisions were handled carefully due to their broad implications for the capabilities and safety of the system at hand.

Hardware

Hardware specifications determine the performance and feature limitations of a system. The speed of a computer chip, for instance, is a harsh limitation that cannot be changed without changing the chip itself. Hardware, then, must provide the functionality and performance needed to meet all requirements.

This project was largely designed around existing hardware, with the most notable decision being the selection of a microcontroller unit. Bare-metal decisions such as circuit-logic,

building materials, etc. were left to part manufacturers or other members of the Electronics team. This section describes the parts used and explains any crucial decisions made during development.

Selecting an MCU

The microcontroller unit (MCU) is a crucial component in any embedded system. MCUs are computer chips with characteristically low speeds, small amounts of memory, and intended for small-scale applications. Two critical parts of an MCU are its communication interfaces and peripherals. Communication interfaces are hardware-implemented protocols and ease the ability of an MCU to communicate with other computers. Peripherals, on the other hand, provide general functionality for the software to utilize.

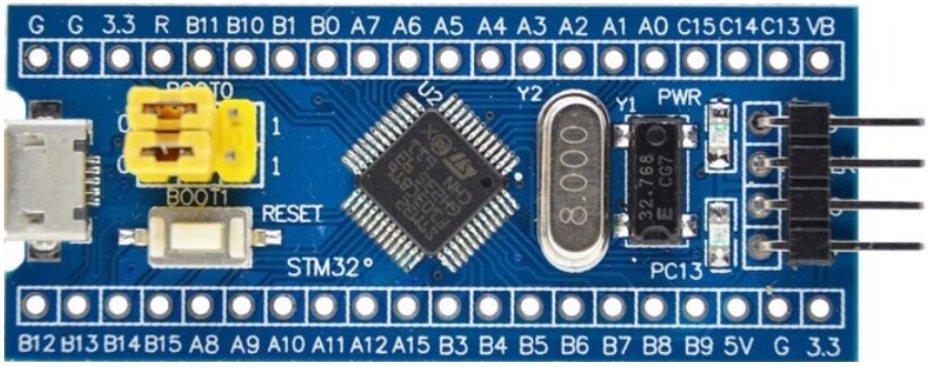
Specifications, development tools, and quality of documentation can vary greatly from one MCU to the next. Several requirements were determined before selecting an MCU for this project.

First, the MCU needed to support the necessary peripherals and communication interfaces to collect and transmit temperature data to the BMS. Although many MCUs could have been programmed to suit this project's needs, hardware support was prioritized to ease the development process. Included in these peripherals and communication interfaces were an Analog to Digital Converter (ADC) and a CAN controller. The ADC is responsible for reading voltage measurements from the temperature probes which the MCU then uses to convert to temperature values. Meanwhile, the CAN controller manages transmitting and receiving messages in the CAN network.

Another requirement was that Software Development Kits (SDKs) must be available for the selected MCU. An SDK is a combination of programs and tools designed to ease software development for a specific MCU or range of MCUs.

The STM32F103C8T6 MCU by ST Electronics [4] was selected because it suited these requirements well. These chips contain 16 ADC channels and a CAN controller, both of which are easily interfaceable through ST's Hardware Abstraction Library. To avoid overcomplicating the design process, the chips were purchased in the 'Blue Pill' package: a 1" by 2" circuit board with the chip, clocks, debugging port, and other external interfaces already mounted as displayed in Figure 2.

Figure 2: STM32 Blue Pill top-view image [5]



MCU Alternatives

Carlson [6] outlines many of the pros and cons of modern MCUs and heavily influenced this project's MCU selection. The boards most carefully considered were the Texas Instruments MSP430, the Arduino Due, and the STM32F103C8T6 Blue Pill. Refer to Table 1 for comparison between boards.

Table 1: Comparing relevant microcontroller boards

Board	Benefits	Negatives
MSP430	- Accessible development tools - Low power consumption	- Large package - Steeper learning curve - Expensive - No CAN Controller
Arduino Due	- Highly accessible development tools - CAN Controller	- No debugging support from manufacturer - Unrealistic development experience - Less configurable
Blue Pill	- Accessible development tools - Small package - CAN Controller - Highly configurable	- Steeper learning curve

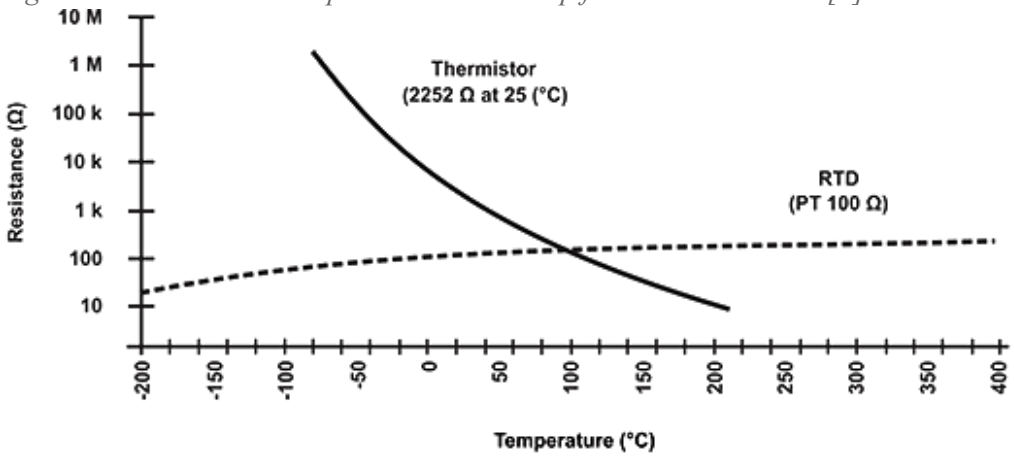
The Blue Pill was chosen because it met the project's priorities better than the other options. Many of the negative characteristics related to the other boards could have been managed with software. However, the lack of debugging support for the Due and the missing CAN controller on the MSP430 ruled these options out.

Temperature Sensors

Negative Temperature Coefficient (NTC) thermistors suit this project well because they are cheap, accessible, and accurate for the temperature range in which they're being used [7]. Thermistors consist of 2 materials bonded together: this bond is strengthened or weakened when exposed to higher or lower temperatures. The strength of the bond determines its electrical resistance: the ability of the material junction to conduct electricity. By measuring electricity flowing through a thermistor you can calculate its resistance, and thus its junction temperature.

NTC thermistors have lower resistance at higher temperatures (as shown in Figure 3), meaning current through the device increases as the temperature increases. (Positive Temperature Coefficient thermistors, on the other hand, have higher resistance at higher temperatures.)

Figure 3: Resistance to Temperature relationship for NTC thermistors [7]



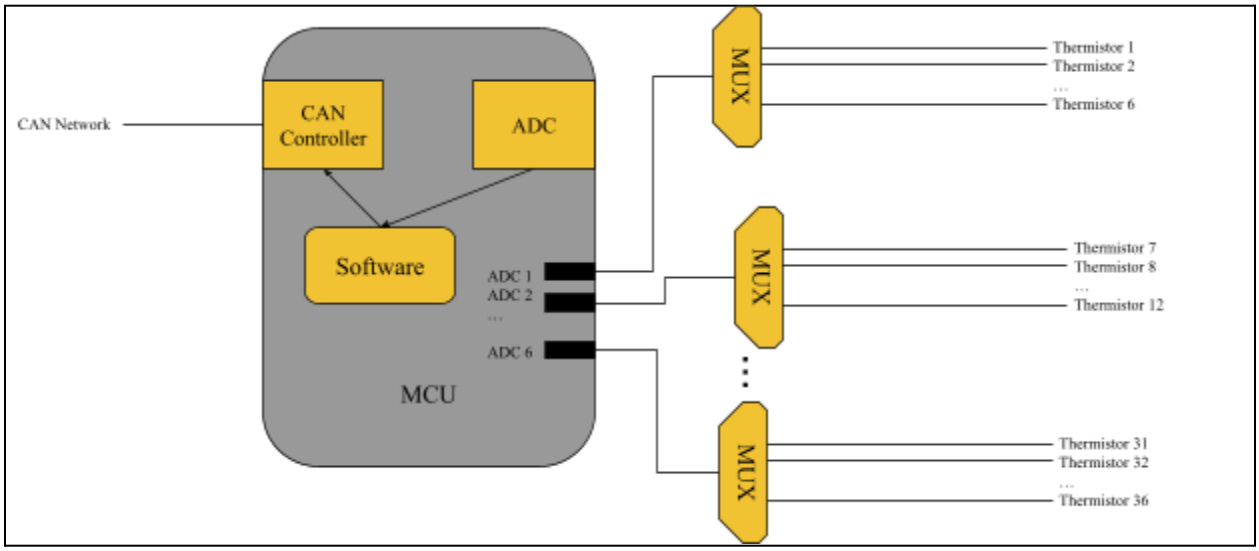
Custom Circuit Boards

The Electronics subsystem team created custom circuit boards to organize thermistors and improve error detection. A total of 216 thermistors needed to be distributed throughout the six segments of the power supply: meaning that each segment required at least 36 thermistors.

The STM32 MCU, however, only consists of 16 available channels to scan temperature sensors. This could be solved by either using more MCUs, or multiplexing all thermistor inputs to one MCU. The Electronics team avoided the latter because wiring all thermistors to one board could drastically increase the wiring complexity by creating a physical dependency between the segments. More than 72 wires exit through a space on the top of each segment. If wires between segments begin to intertwine, it could create additional overhead when constructing or deconstructing segments.

The final decision was to use one MCU per segment. This configuration maintains isolation of the segments, eases cable management, and does not require complex multiplexing. Each MCU needed to collect data from 36 sensors. The custom circuit boards organize thermistors into 6 groups with 6-1 multiplexers (MUXs), resulting in 6 readable channels for the MCU (as shown in Figure 4).

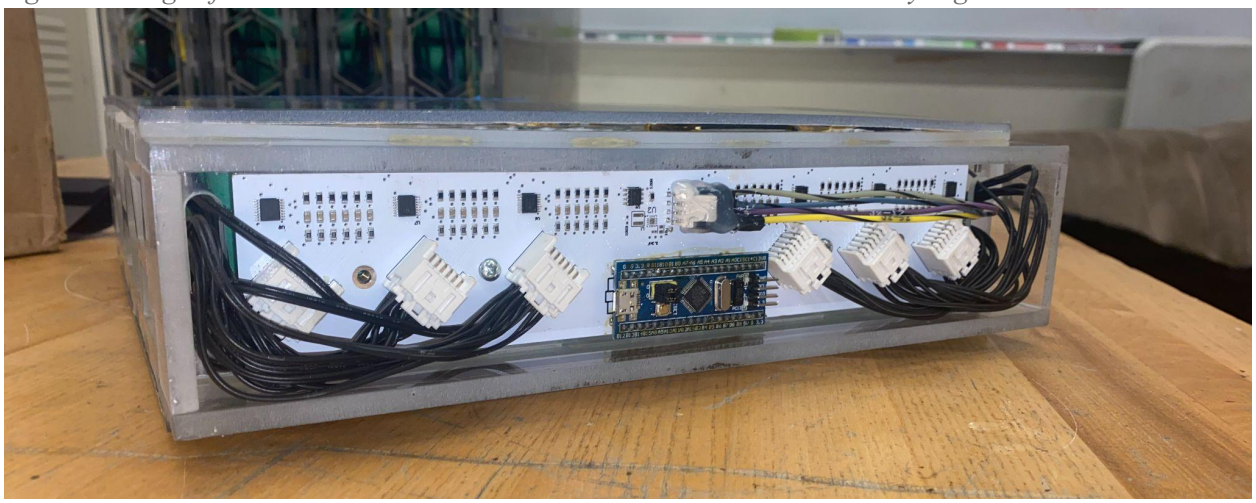
Figure 4: Overview of a TEM



The circuit design also eases error detection for the software. A melted or disconnected thermistor (the most likely failure modes in operation) will no longer provide electrical resistance to the circuit, resulting in a maximum voltage measurement by the MCU. The software, then, is able to detect this value and determine that an error has occurred.

Figure 5 displays the final construction of the hardware. The STM32 can be seen mounted on a white custom circuit board with 36 thermistors attached.

Figure 5: Image of an STM32 and a custom circuit board mounted to a battery segment



Software

As shown in Figure 4, the software plays a surprisingly simple role in the system. The software design was intentionally kept as simple as possible due to rising pressure from the university to finish a car. Prioritizing simplicity ensured a reliable timeline and made the code more readable and maintainable for future team members. The largest software-design considerations were associated with extra features for the boards and were discarded in favor of maintaining simplicity. The software's main responsibilities can be broken into the following stages:

1. *Configuration*: A series of parameters must be configured to determine the functionality of each peripheral
2. *Voltage-Temperature Conversion*: The software needs to periodically collect all of the voltage values from the ADC, and then convert those measurements into temperatures
3. *CAN Message Creation*: The software must periodically create and transmit a CAN message with summary statistics from all of the collected temperature values

This section is dedicated to describing how these stages function, and what tools were used in the process of their development.

Development Tools

This project was written in the C programming language and developed with System Workbench, an Eclipse-based Integrated software Development Environment (IDE) created by ac6 Tools [8]. IDEs provide text editing features, software packages, and a user interface for writing code. They are commonplace in the software development industry because they significantly improve the efficiency of code production. System Workbench includes the following packages:

- *GNU Debugger* [9]: allows developers to step through C code as it executes
- *OpenOCD* [10]: provides an interface for uploading code to a board's memory storage
- *ST's Hardware Abstraction Library (HAL)* [11]: abstracts the features and functionality of hardware into a simplified interface

The GNU Debugger is particularly useful because it enables developers to ensure that each line of code produces the expected result: simplifying the process of troubleshooting coding errors. OpenOCD and ST's HAL, on the other hand, allow the developer to focus on the software functionality without being concerned with the specifics of the hardware.

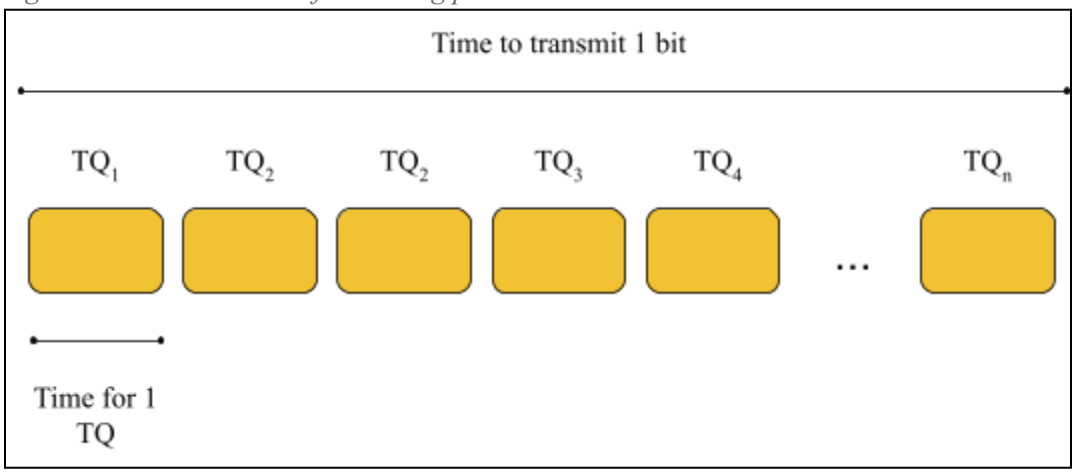
Stage 1: Configuration

Each peripheral consists of a series of parameters that can determine its accuracy, performance, or functionality. Understanding how each parameter impacts a peripheral is critical to ensuring that the system is functioning properly and efficiently.

The ADC consists of many configurable parameters, but their default values suffice for this project. The ADC then performs one conversion at a time, making it convenient to frequently switch between ADC channels.

Configuring the CAN controller entails selecting an operating mode and establishing the bit-timing. The CAN operating mode has a number of options, however the only relevant configuration is the 'Normal' mode: allowing the controller to send and receive messages in the CAN network. The CAN bit-timing, meanwhile, determines the amount of time to transmit each bit in the network [12]. This configuration must be consistent across every device, otherwise the CAN controllers are unable to synchronize, leading to misread messages. The bit-timing is determined by two parameters: *n* and Time Quanta (TQ). A TQ is a small but unified time measurement, usually measured in nanoseconds. Each bit consists of *n* TQs, as shown in Figure 6.

Figure 6: Demonstration of bit-timing parameters



Since the amount of time for a single bit is specified by the bit-timing, *n* and TQ also determine the bit-rate of the CAN network. Bit-rate is the amount of bits capable of being sent in a single second. The CAN protocol specifies standardized bit-rates with a minimum of 10 Kb/s, and a maximum of 1 Mb/s. The length of the network, frequency of messages, and number of devices on the network all determine the bit-rate that a network can sustain [13]. Since there are only six devices in the CAN network (1 TEM per battery segment) that transmit every 100ms, and the network distance is less than a meter, the network is capable of sustaining a bit-rate of 1 Mb/s. 500 Kb/s was chosen, however, because it is adequate for the intended use and may be more reliable.

ST's HAL provides useful abstractions of the hardware which ease the configuration of peripherals. Each peripheral is configurable by providing a function with a structure of parameters that the HAL then translates to hardware functionality. A simplified example of configuring the CAN controller using the `HAL_CAN_Init` configuration function is shown below:

```
can_object.Mode = CAN_MODE_NORMAL;
can_object.N = CAN_N;
can_object.TimeQuanta = CAN_TQ;
HAL_CAN_Init(can_object);
```

Stage 2: Voltage - Temperature Conversion

Measuring temperature with a thermistor starts by collecting a voltage value from the Analog-Digital Converter (ADC). This value is then used to solve for the resistance *R* of the thermistor (see Dostal [14] for calculating resistance in a voltage divider), which is in turn used to solve for the temperature as shown in the equation below [7], where T is the current temperature of the thermistor and T_0, $R(T_0)$, and β are values specified by the manufacturer.

$$R = R(T_0) * e^{\beta(1/T - 1/T_0)}$$

Stage 3: Creating a CAN Message

As mentioned previously, the following information needs to be transmitted to the BMS:

- Average temperature of the cells
- Highest recorded temperature
- Lowest recorded temperature
- Number of failed sensors
- An ID for the transmitting module

The average temperature of the cells is calculated by summing the temperatures as they are retrieved and then dividing by the total number of functional thermistors. Meanwhile, highest/lowest temperatures are stored in variables as they are found. Lastly, failed thermistors are detected by checking for voltage measurements close to the maximum value and counted as they are found. A CAN message is then created and transmitted every 100ms in the specified format by the Orion BMS 2 [3].

Pseudocode

Sections one and two are repeated indefinitely throughout the on/off cycle of the vehicle, as presented in Figure 7.

Figure 7: Pseudocode for the general functionality of the software

```
Loop forever
     for i = 1 to 36 (number of thermistors)
          Select an ADC channel
          Collect voltage measurement from ADC

          If an error occurred
               Increase faulty_thermistors
          Else
               Calculate temperature from measured voltage
               If temperature is above max:
                    max = temperature
               Else if temperature is below min:
                    min = temperature
               Add temperature to sum

     average = sum/(36 - faulty_thermistors)

     Fill message_data with max, min, average, and faulty_thermistors
     Transmit message_data to the CAN network
     Delay for 100ms
```

Verification and Validation

The role of the TEM is to accurately report temperatures and detect when thermistors are malfunctioning. Therefore, the following conditions need to be guaranteed:

- Accuracy throughout the expected range of temperatures
- All possible failure modes are detectable
- CAN transmissions are reliable

The Orion BMS 2 owns the responsibility of responding to temperature changes. Therefore, the test plan in Table 2 focuses on the TEM independently, refer to Appendix A for a technical description of these tests. Test results were determined by analyzing CAN messages with a CANdapter [15], a USB device that collects and logs CAN messages in real-time.

Table 2: Test plan for TEMs

Test	**Description**	**Acceptance Criteria**	**Result**
Failed thermistor	Attach a Blue Pill to one of the custom boards with 35 functional thermistors, a single dysfunctional thermistor	CAN message reports an error	PASSED
Failed thermistor connection	Same setup as test 1 but with 36 functional thermistors. Disconnect thermistor connectors one at a time.	CAN message reports an error	PASSED
Accuracy	Same setup as test 2. Verify CAN messages are accurate with a temperature sensing device. Use a heat gun to raise the temperature of a thermistor to above 60℃.	CAN messages maintain accuracy	PASSED
Thermal overload	Use the same setup as test 2. Use a high temperature device to achieve 500℃ at the thermistor.	CAN message reports an error	PASSED
Integration	Setup the BMS to be functional according to Ewert's requirements [16]. Use a heat gun to raise the temperature of at least one thermistor to above 60℃.	Shutdown signal engages BMS reports a 'Pack too hot' error	PASSED

Considering Extreme Conditions

The test plan in Table 2 does not cover extreme real-world conditions, such as when a cell combusts: the team lacks the resources to safely do so. However, the BMS monitors the necessary parameters to prevent a catastrophic failure. These parameters include the battery's input and output current, voltage measurements for each cell group, and cell temperature. A catastrophic failure could only occur under the following conditions:

- A cell group becomes unbalanced, meaning it measures more or less voltage than other cell groups
- A cell overheats
- A cell is punctured

The BMS can prevent the first 2 conditions because it monitors the temperature and voltage values for all cell groups. The final condition, however, could occur unexpectedly, which could result in two conditions: either the voltage tap becomes disconnected from the cell group, or it stays connected to the cell group. If the former occurred, that voltage tap would read an open circuit (0V). If the latter occurred, the BMS would detect a dramatic change in voltage due to the combusted cell(s). In either case, the BMS would immediately engage the Safety Shutdown Circuit and BMS Indicator Light, per FSAE rule EV 8.3.4. Furthermore, the test plan verifies the TEM's ability to detect a cell's combustion temperature and transmit an error to the BMS.

Reflection

Due to the setbacks introduced by the COVID-19 pandemic, the team was unable to complete the car before the competition. The team aimed to finish building the car by April of 2021, leaving three months to test, fix, and adjust components before competition on June 15th of that year. Lack of access to the team's lab, tools, and other resources unfortunately did not make this goal possible. The car was mostly completed before competition, however. The only remaining tasks involved wiring and testing. Considering these setbacks, the university considered the team's progress sufficient to increase their support for the club. This section is devoted to the complications and accomplishments in preparing for the 2021 season, as well as improvements for future seasons.

Complications

COVID-19 introduced a series of complications throughout the development of the car. Due to the state guidelines and the collective interest in the team's safety, contact was very limited between members. Communication was conducted virtually through Discord and Zoom meetings, severely affecting members' abilities to collaborate on their projects. Compatibility among components was difficult to guarantee, creating many issues when constructing the car. Additional costs and setbacks were also created by the restricted access to the team's lab. The lab contains tools, equipment, and space which are crucial to the development of a car. This led to manufacturing many components externally, creating delays while waiting for appointments, added labor costs, and inducing unexpected tolerancing issues. All efforts were made to complete

the car: however, there were simply too many issues for safe and timely completion to be possible.

This project specifically was most impacted by communication strain. The team had attempted embedded projects such as this in the past, however there were no remaining resources. Research was done completely from scratch — issues with software tools, pinouts, datasheets, etc. were handled on a case-by-case basis. Solutions were either discovered independently by the author of this thesis (who began the project with a scant understanding of electrical principles), or in collaboration with the Electronics lead, Braeden Hamson. Virtually troubleshooting issues was exceptionally difficult without the ability for all parties to physically interact with components. For example, building a functional CAN network was delayed about three months because of one issue. Wiring diagrams were carefully followed to construct the network, however messages were simply not received by other devices. Many hours were spent with an oscilloscope or digital multimeter analyzing connections. Ultimately, the Electronics lead discovered that one of the devices had been soldered upside down so connections were incorrect: an easy mistake to notice for someone with years of experience in electrical systems but not even a consideration for the author.

Accomplishments

Overall Success

Figure 8: The VMS team photo at the 2021 FSAE Competition

Many FSAE teams struggled to prepare for the 2021 season. Of the 20 EVs that participated in the 2021 competition, only one car was able to complete all dynamic challenges [17]. Although issues with tolerancing and manufacturing quality arose in the construction of the VMS EV, few extensive redesigns were needed. Rules experts at the competition suggested only two changes regarding the vehicle's electronics: the first being that the high voltage and low voltage lines were crossing, and the second being that a fuse in the power supply needed additional protection; both of which are simple fixes. Otherwise, the car had a functional suspension, chassis, and drivetrain, as shown in Figure 8. Considering the unfortunate

circumstances of the COVID-19 pandemic and the team's progress, Portland State University decided to increase their involvement and support for the team.

Real-time systems, such as this project, have a justifiable purpose in the context of building an EV. However, with the foundations of an EV laid, the team now has the flexibility to build more complex embedded systems with more robust hardware. Projects for future FSAE seasons are now considering single-board computers, such as the RaspberryPi, that are capable of running embedded operating systems such as Linux. One such project is an interactive display embedded in the car's steering wheel.

VMS Restructuring

The team's restructuring in 2019 played a crucial role in the team's progress during the 2021 season. The following practices were added to improve the team's ability to maintain communication and awareness:

- The team met weekly to discuss progress and blockers
- Subsystems met separately to discuss individual projects' progress
- A carefully maintained Gantt chart outlined each subsystem's progression
- The team met with Portland State University faculty on a quarterly basis to ensure safety and plausibility of proposed designs

Although the team was unable to compete during the 2021 season, these processes enabled the team to make the progress it needed. With hopefully fewer setbacks than in 2021, these processes will help ensure the team's success in future seasons.

Documentation Improvements

As discussed in the Complications section, poor documentation, or lack thereof, can create significant amounts of overhead when developing a project such as this. Effective documentation reduces the need to recollect information and eases onboarding of newcomers because the information is already organized in an understandable way. Each step in the development of this project was intentionally documented for the aforementioned reasons. This author wrote various project-specific documents, including a document dedicated to introducing future team members to embedded development. The documents also include contact information if the need for additional mentoring arises.

Future Work

Since the vehicle has yet to be completed, the TEMs have not been tested in a real-world environment. Although it is believed that the tests have validated all conditions, it is possible that issues will arise as the car is completed. Furthermore, as the car continues to be reiterated in future seasons it is possible that requirements and design changes will impact the TEMs. Due to the simplicity of the software, availability of the author, and documentation on this project, however, the team has the resources to be able to address any changes or issues that arise.

Conclusion

This thesis documents the development of a Thermistor Expansion Module for Viking Motorsports' custom Electric Vehicle power supply. Requirements introduced by the Orion BMS 2 and the 2021 FSAE Rulebook guided the software and hardware designs. Once complete, the modules were tested to ensure that all requirements were met. Unfortunately, the team was unable to complete their EV for the 2021 season due to setbacks caused by the COVID-19 pandemic. However, the team laid solid foundations for future seasons.

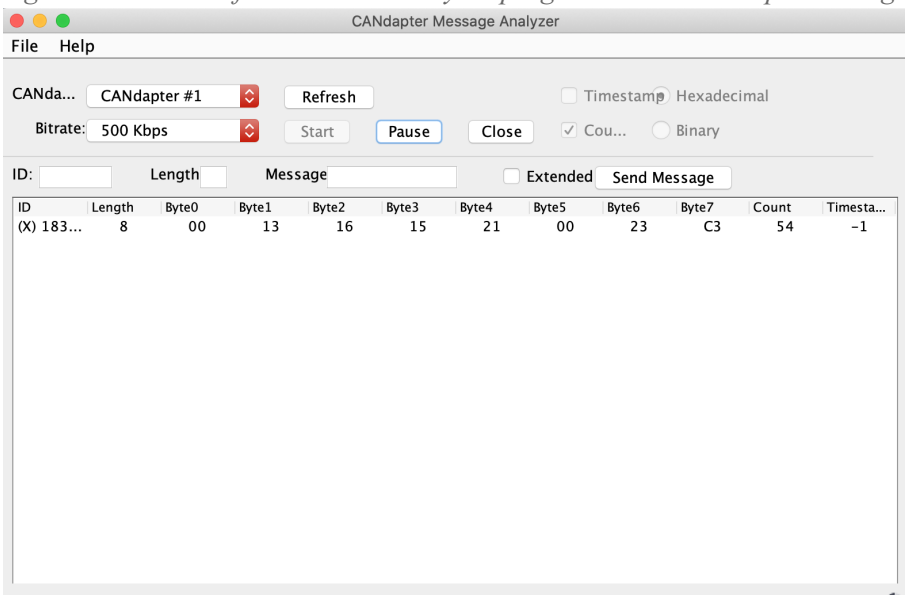
Appendix

Appendix A: Test Descriptions

The first four tests validate the TEM as an individual unit, ensuring accuracy and reliability throughout the various conditions to which a TEM may be used. The final test, however, establishes that the reported data from the TEMs is properly interpreted and handled by the BMS.

Testing was performed with the final hardware configuration, as displayed in Figure 5. A TEM was mounted to a battery segment with 36 thermistors connected. The CANdapter was then attached to the CAN network with an unused CAN connector. Vector's CANalyzer program [18] was used to monitor CAN messages during the tests. Figure 9 displays the CANalyzer user interface with an example message reported in hexadecimal.

Figure 9: User interface in the CANalyzer program with an example message



A TEM is able to report an error to the Orion BMS 2 by setting the eighth bit of the fourth byte in the CAN message [3]. If, for instance, an error had occurred in Figure 9, byte four in the example message would read any value greater than 0x80.

Unit Tests

The tests ‘Failed thermistor’ and ‘Failed thermistor connection’ address the two most common thermistor failure conditions

1. The bonded materials disconnect from the thermistor’s leads (typically due to aging or excessive heat)
2. The leads disconnect from the circuit.

To emulate a dysfunctional thermistor, a thermistor was cut where the lead attaches to the tip. Testing a failed thermistor connection was completed by disconnecting three thermistor connectors and checking the CAN message after each disconnection. Both failure conditions result in an open circuit which the software handles by detecting a maximum voltage value and

setting the error bit in the CAN message, as explained in the Design section. Results were determined by verifying that the error bit was set.

The ‘Accuracy’ test ensures the TEM is providing reliable data to the BMS. Validation was performed by analyzing the CAN message outputs relative to a reliable source. Fortunately, the team had access to an authentic TEM from Ewert Energy. This was connected to an available CAN connector and messages were read from both TEMs through the CANalyzer program. Thermistor tips from both TEMs were held together and then the temperature was raised to 50, 60, 70, and 100℃ using a soldering gun. In all instances, the custom TEM was within zero to two degrees of the authentic TEM. This variation is expected due to differences in calculations between the devices and uneven heating due to the use of the soldering gun.

Lastly, the ‘Thermal overload’ test validates the TEMs behavior in extreme conditions. 500℃ was used as the testing temperature because it is the combustion point of lithium-ion cells. In order to emulate this condition, a soldering gun was set to 500℃ and placed directly on a thermistor tip. Under high temperature conditions, the TEM is not responsible for reporting an error because the BMS will receive and handle the high temperature value. However, since temperature data points in the CAN message are transmitted with one byte each [3], the value range of each data point is only 0 - 255. The software handles this by setting the error bit after detecting a temperature greater than 255℃. Results were determined by verifying that the error bit was set in the CAN message after the thermistor was exposed to 500℃.

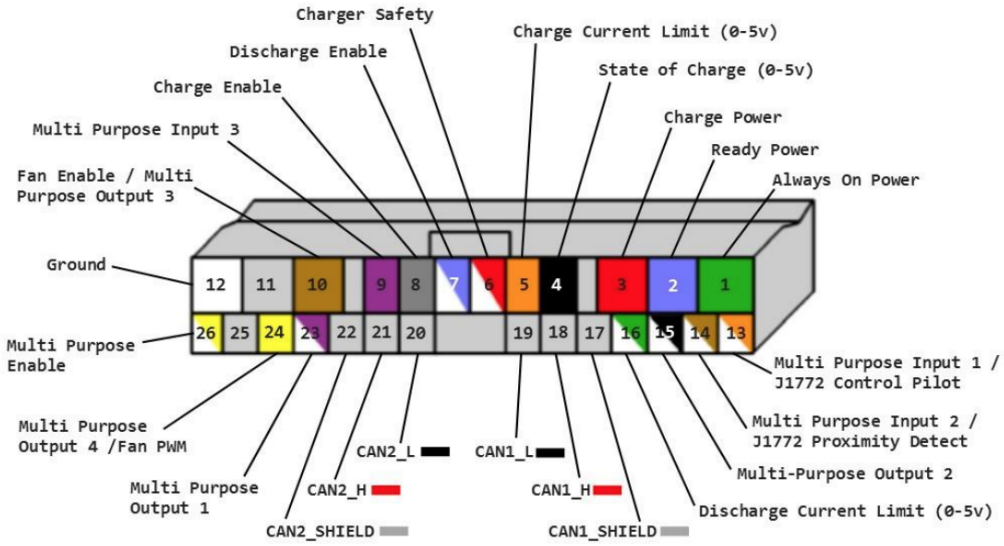
Integration Test

This test verifies that the TEM is sending messages that are readable by the Orion BMS 2. Although this could be validated by plugging the TEM into the CAN network and verifying

that the BMS does not throw any temperature-related error codes, adding a response condition for the BMS further demonstrates the system's functionality in safety-critical conditions.

Unfortunately, wiring for the safety shutdown circuit is not complete so this test is based only on the signal sent from the BMS. Figure 10 shows the pinout for the Orion BMS 2's main connector. The 'Discharge Enable' wire, or pin 7, is used to indicate that the power supply is in safe operation. Therefore, this is also the wire used to indicate an error in the system.

Figure 10: Pinout for the Orion BMS 2's main wiring connector [16]



This test was set up by plugging a TEM into the CAN network, programming the BMS to send a shutdown signal and record a 'Pack too hot' error at temperatures above 60℃, and adding an LED light to the shutdown signal. A soldering gun was then used to raise the temperature of a thermistor to above 60℃. At that temperature the light turned on and a 'Pack too hot' error was recorded, indicating the test's success.

Availability

Source Code

Repository: https://github.com/VikingMotorsports/THERM2CAN

This repository contains the following items:

- Source files
- Dependencies (CMSIS files, HAL Driver, code generated by System Workbench)
- Binaries
- Software License

This code is available under the GNU General Public License V3.0.

Other Items

Other items (hardware designs, documentation, etc.) are not publicly available.

References

[1] "FSAE Rules 2021." SAE International, 2020. [Online]. Available: http://fsaeonline.com/cdsweb/gen/DownloadDocument.aspx?DocumentID=72e6bc4d-a65d-48f7-ac65-2a6a2e1b87d4

[2] R. Bosch, "BOSCH: CAN Specification." BOSCH, 1991. [Online]. Available: http://esd.cs.ucr.edu/webres/can20.pdf

[3] "Thermistor Module CANBUS Protocol." Ewert Energy Systems. [Online]. Available: https://www.orionbms.com/downloads/misc/thermistor_module_canbus.pdf

[4] "STM32F103x8 & STM32F103xB." ST Electronics, 2015. [Online]. Available: https://www.st.com/resource/en/datasheet/stm32f103c8.pdf

[5] M. Koch, *Blue Pill Starter Kit (STM32F103)*. [Online]. Available: https://mecrisp-stellaris-folkdoc.sourceforge.io/bluepill-starterkit.html

[6] J. Carlson, "The Amazing $1 Microcontroller", [Online]. Available: https://jaycarlson.net/microcontrollers/

[7] "The Resistor Guide, your guide to the world of resistors," no. Chapter 3-Resistor Types, p. NTC Thermistor.

[8] "System Workbench for MCU: Embedded Microcontroller Development Environment - ac6-tools." https://www.ac6-tools.com/content.php/content_SW4MCU/lang_en_GB.xphp (accessed Nov. 08, 2021).

[9] "GNU Debugger." https://sourceware.org/git/gitweb.cgi?p=binutils-gdb.git (accessed Nov. 14, 2021).

[10] "OpenOCD - Open On-Chip Debugger / Code / [a498a3]." https://sourceforge.net/p/openocd/code/ci/master/tree/ (accessed Nov. 14, 2021).

[11] *STM32CubeF1 MCU Firmware Package*. STMicroelectronics, 2021. Accessed: Nov. 14, 2021. [Online]. Available: https://github.com/STMicroelectronics/STM32CubeF1

[12] S. Robb, "CAN Bit Timing Requirements." Freescale Semiconductor, 1999. [Online]. Available: https://www.nxp.com/docs/en/application-note/AN1798.pdf

[13] S. Corrigan, “Controller Area Network Physical Requirements.” Texas Instruments, 2008. [Online]. Available: https://www.ti.com/lit/an/slla270/slla270.pdf?ts=1635022881630&ref_url=https%253A%252F%252Fwww.google.com%252F

[14] F. Dostal, “Voltage Dividers in Power Supplies,” *EE Power*, 2019, [Online]. Available: https://eepower.com/technical-articles/voltage-dividers-in-power-supplies/

[15] Ewert Energy Systems, *CANdapter*. Accessed: Nov. 14, 2021. [Online]. Available: https://www.ewertenergy.com/products.php?item=candapter

[16] “Orion BMS 2 Wiring and Installation Manual.” Ewert Energy Systems, 2018. [Online]. Available: https://www.orionbms.com/manuals/pdf/orionbms2_wiring_manual.pdf

[17] “Formula SAE Nevada 2021 EV Results.” SAE International, Jun. 28, 2021. [Online]. Available: https://www.sae.org/binaries/content/assets/cm/content/attend/2021/student-events/formula/fsae_nv_ev_2021_result.pdf

[18] “CANalyzer – ECU & Network Analysis | Vector.” https://www.vector.com/int/en/products/products-a-z/software/canalyzer/ (accessed Nov. 14, 2021).