Portland State University

Viking Motorsports

ECE Capstone 2014/2015

Digital Dashboard

Noah Erickson
Rishal Dass
Chad Thueson
Jaime Rodriguez
Sean Koppenhafer

www.github.com/noahterickson/digitaldash

Abstract

The increased complexity that high voltage systems introduce into a race car require increased diagnostics to keep the car running safely and correctly. The driver must have the status of all important subsystems visible to clearly see and determine the vital faults. With an updated dash system detailed information can be displayed in real time from the vehicle control unit. The driver can be kept up to date on system functionality both on track or in the pits. The solution to display all this critical information is a small 3.5 inch LCD screen that will receive information from the subsystems of the vehicle and display them in a simple Graphical User Interface. The interface will not only display the warnings and faults but will also display useful real time information such as motor torque, battery level and amperage draw of the motor.

Contents

1 Introduction

1.1 Purpose

The project's objective is to provide the driver/pit crew of the Viking Motorsports Formula SAE electric vehicle with information about the vehicles operation. Information from the on-board vehicle control unit (VCU) will be gathered, interpreted, and sent out for display by a main processing unit. The interpreted information will be displayed on a liquid crystal display (LCD) mounted within the drivers view. Mission critical information, such as warnings about the battery management systems (BMS), insulation-monitoring devices (IMD) and warnings regarding the motor controller information will also be displayed on the LCD.

1.2 Motivation

Currently, the electric race car uses a set of LEDs that light up to indicate that an error has occurred. Although this works, it would be more productive for the driver and the pit crew to know more specific information about the error that has occurred. For example, it would be more convenient for the driver to know exactly how much battery charge is left, instead of just knowing that their batteries are merely low. This extra bit of information could help the driver to decide if they could make it another lap or if they need to make a pit stop.

1.2.1 Vehicle Complexity

The Viking Motorsports team has raced an internal combustion engine vehicle in Formula SAE races for many years. With many years of prior design, their new electric vehicle has been modeled after the internal combustion engine car to compete in the electric class of Formula SAE. An electric vehicle has a more complex electrical system than its internal combustion engine counterpart. The added complexity means that more data must be monitored by the Viking Motorsports Team. The digital dashboard will work with the vehicles subsystems to display relevant information more accurately.

Figure 1: Traditional LED Dashboard

1.2.2 Setting a Standard

A secondary motive for the project is to create a guide that other Formula SAE teams could follow if they wished to build their own digital dashboard. To support this ideal, the Viking Motorsports Team requested that we keep the project open source so that it is easily accessible. Perhaps if enough teams within the electric Formula SAE class adopt a digital dashboard, this project can evolve to support new features outside the scope of this proposal.

Currently the Formula SAE rules require a row of light emitting diodes (LEDs) on the dashboard of any competing electric vehicle.[1] The LEDs are used to indicate when a system failure or warning occurs. If the digital dashboard was adopted by a large majority of the Formula SAE teams, it may be able to replace the LEDs in the rulebook.

2 Problem Statement

The LCD must be readable in all weather conditions. However, readability in sunlight is a primary concern. A complementing waterproof enclosure would allow the device to operate in rainy conditions. The digital dashboard screen

must also be large enough to be readable for the driver, but small enough to fit in the minimal space available in the vehicles dashboard.

2.1 Project Requirements

2.1.1 Operation

- The screen must be visible in direct sunlight.
- The digital dashboard should gather information from the vehicles subsystems and display it in an organized manner.
- The displays interface must be simple and clear so driver can easily read the screen with a quick glance. The interface should include large text and symbols with bright colors.
- Data does not need to be displayed in real-time.
- The electric vehicle must be independent of the digital dashboard and be able to operate without the digital dashboard.
- Display must boot up in less than 30 seconds.

2.1.2 Power

- The digital dashboard must be powered by a 12V or 5V DC power supply.
- The digital dashboards power supply should be able to handle electrical noise present in the vehicle.

2.1.3 Health & Safety

- The digital dashboard unit must be securely bolted to the vehicles frame.
- It must be mounted somewhere inside of the cockpit to comply with Formula SAE rules.[1]

2.1.4 Environmental

- Must be made with non-toxic/hazardous chemicals that would require special disposal.

2.1.5 Maintainability

- Must have minimal to no user maintenance over entire life cycle.

2.1.6 Dimensions

- Must be less than 4 inches in width.
- Must be less than 3 inches in height.
- Must be small enough for the driver to see the screen between the spokes of the steering wheel.

2.1.7 Reliability & Availability

- The digital dashboard must be rugged and water resistant (IP65 standard) as it will be exposed to wet weather conditions.
- Must be easy to install on a variety of vehicle frames.

2.1.8 Price

- Total parts and manufacturing should not exceed $500.
- The resulting product should be inexpensive to compete with the current LED system.

3 Solution Space

The digital dash unit will comprise of two inputs and one output. The unit contains two CAN transceivers that will receive input for the device. Each transceiver receives data from one of the vehicles two CAN busses. The digital dash will process these inputs and output the interpreted data to an LCD display over TTL serial.

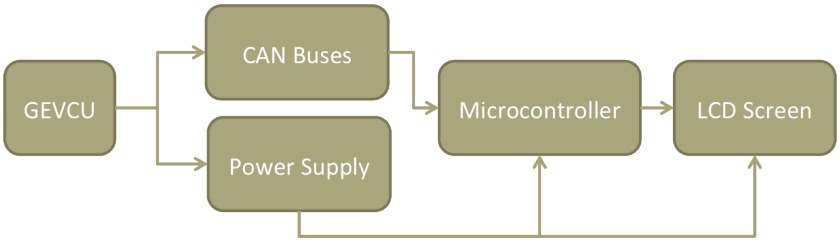

Figure 2: Functional Block Diagram

3.1 Design Decisions

3.1.1 Microcontroller

After many hours of research, our microcontroller choices were narrowed down to these four chips: the Atmega328, the AT90CAN, the Arduino UNO (which uses an Atmega328p chip) and the Arduino Due (which uses an ATSAM3X chip). All of these chips are Atmel brand because we were familiar with their chips from prior classes.

Both the Atmega328 and the Arduino UNO would have required two CAN controllers connected over I2C or SPI to communicate with the CAN busses on the car, so we quickly discarded these as potential microcontrollers. We felt that this would create too much overhead for such a low frequency 8-bit microcontroller.

The AT90CAN chip was the favorite microcontroller for a while. It seemed simple, inexpensive, and included a CAN controller on the chip. However, it turned out to not be as great as we thought. One big issue was that the chip did not support Arduino, which meant that we would not be able to use any Arduino libraries with the chip. While the chip also had a CAN library available from Atmel, it was poorly written and there were no examples of how to use it. The limited examples found online showing how to use the CAN controller on the chip used low level register programming. What finally put us over the edge was the lack of processing power. The AT90CAN is an 8-bit microcontroller that runs at a maximum frequency of 20MHz. With only one CAN bus and a large number of CAN messages to process from the car, we discarded the AT90CAN as

a potential microcontroller.

This left us with the Arduino Due. The ATSAM3X chip is a 32-bit ARM processor that runs at 84MHz which ensured that we would have more than enough processing power.[3] The chip also contains two CAN controllers which can be controlled with a CAN library written by Collin80 on Github [4]. Finally, the chip supports Arduino, which means that we can use Arduino libraries. The Arduino Due and the ATSAM3X chip checked all of our requirement boxes, so it was an easy choice to make in the end.

By using the ATSAM3X chip, the development team was able to quickly begin prototyping the digital dashboard on an Arduino Due.

	Cost	Fab Board	CAN Controller	CAN Library	SPI, I2C, UART
Atmega328	$3	Yes	No	No	Yes
AT90CAN	$8	Yes	Yes	Yes	Yes
Arduino UNO	$25	No	No	No	Yes
ATSAM3X	$12	Yes	Yes	Yes	Yes

Table 1: Microcontroller Choices

3.1.2 CAN Transceivers

Initially, we were certain that the AT90CAN would be the microcontroller that we would use for the project. Because the AT90CAN only had one CAN controller, we were looking at using the AMIS-42700 CAN transceiver because it could combine the two CAN busses on the vehicle into one bus for the single CAN controller. Once the ATSAM3X was chosen as our microcontroller, we no longer needed to look at this chip.

In the end, two MCP2551 CAN transceivers were chosen to interface with the vehicles CAN busses. One transceiver is needed for each CAN controller on the ATSAM3X chip. They are inexpensive, popular, operate with CAN baud rates up to 1Mbit/sec and come in both through hole and surface mount packages.

	Cost	Max Transfer Rate	Multiple CAN Networks
MCP2551	$1.50	1 Mbit/sec	No
AMIS-42700	$6.00	1 Mbit/sec	2 Networks to 1

Table 2: CAN Transceiver Choices

3.1.3 LCD Screen

Finding a suitable LCD was the most difficult part of choosing components. We needed to find something that was sunlight readable, small enough to fit in our limited space, and included a display driver board to offload graphical processing from the ATSAM3X chip. This turned out to be no simple task.

The New Haven display and the SunLCD were never really considered all that much during component selection. The New Haven display was too large for the space that we had to place the dashboard, so it was quickly discarded. The SunLCD was quickly discarded because we could not get in contact with the company to get a price on the display. It also did not have any software support that we could find on their website.

Initially, the EarthLCD was the favorite LCD. It was the perfect size, built for readability in sunlight, and included an Arduino library to interface with the display. It fit in the bill in all departments but price. At $219, it was the most expensive screen by far and would have taken up almost half of the $500 budget.

The more we looked at the 4D systems display, the more we liked it. It was inexpensive, had both a development IDE and an Arduino library for communicating with the screen, and had great support online. The only thing it lacked was sunlight readability.

Ultimately, we decided to go with the 4D systems display because its cost advantage over the EarthLCD. If sunlight readability was an issue, we figured that we could attach some sort of polarizing shield over the screen to increase its readability.

	Cost	Size	Color	Sunlight Visibility	Software
EarthLCD	$219	3.5”	Yes	Yes	Yes
New Haven	$60	4.9”	No	Yes	No
SunLCD	$???	3.5”	Yes	Yes	No
4D Systems	$89	3.5”	Yes	No	Yes

Table 3: LCD Choices

3.1.4 Power Supply

The digital dashboard needed to draw its power from the electric vehicle. Running the digital dashboard off a battery pack would cause the unit to need regular maintenance to ensure functionality. The electric vehicle has a traditional 12 volt battery to power low voltage systems that the design team originally planned on getting power from. Once the Viking Motorsports Team switched their vehicle control unit to the Generalized Electrical Vehicle Control Unit (GEVCU) we noticed that the GEVCU had extra 5 volt power connections. Since the digital dashboard was already getting CAN data from the GEVCU we decided to draw power from the GEVCU to simplify installation of the digital dashbaord. The power consumption of the digital dashboard was not a concern since it would be minuscule compared to the rest of the systems on the vehicle.

3.1.5 Software

The software for the ATSAM3X was written in the Arduino IDE to provide compatibility and ease of use for future maintenance by the Viking Motorsports Team. The interface for the LCD was laid out in the software IDE provided by 4D Systems. The 4D Systems IDE allows for drag and drop placement of screen objects allowing rapid development of the user interface that even non-technical people can understand. This is a great for the Viking Motorsports Team because it ensures that anyone can easily modify the interface in the future if they so choose.

3.1.6 Enclosure

The enclosure needed to be as compact as possible in order to fit between the triangular space between the steering column supports. Visibility of the digital dashboard was also important, which dictated the enclosures placement. One option was to install the digital dashboard on the right side of the steering column supports. This idea was dismissed because the drivers hand would cover the screen when holding the wheel straight - a prime time for viewing the screen. A second option was to install the digital dashboard in the middle of the triangle created by the steering column support. This would allow the driver to see the screen when holding the wheel straight, but would allow less room overall for mounting the digital dashboard unit.

We decided to go with option two and install the digital dashboard in the middle of the steering column supports. This is the only place where the driver would be able to see the screen while driving straight and it was the only spot we could logically mount the screen on the dashboard.

3.1.7 Communication Protocol

The GEVCU on-board the Viking Motorsports Electric Vehicle must have some integration with the digital dash unit in order to transmit data. There are a number of different protocols that are available to transfer data from the GEVCU to the digital dashboard unit. A few communication protocols the design team considered using were: Wi-Fi, I2C, Serial and CAN.

Since the GEVCU is Wi-Fi capable, this seems like an obvious choice in order to cut down on physical hardware. Implementing Wi-Fi into the digital dashboard unit would be simple as picking apart a Wi-Fi Arduino Shield and integrating it into our custom mainboard. However, Wi-Fi was dismissed fairly early on because of the added complexity for little benefit.

Another option would have been serial communication between the GEVCU and the digital dashboard unit. The ATSAM3X has quite a few serial communication options in the form of UARTs, I2C, SPI and CAN interfaces. The design team

decided against using a UART, I2C and SPI and decided on CAN because the subsystems on the electric vehicle already use the CAN protocol to communicate with each other.

Since CAN is not intrusive and grabs all messages that come across the network, it made sense to use it as our communication protocol. With CAN we would not have to change anything on the electric vehicles to support our digital dashboard, so it would be a plug and play system.

4 Implementation

4.1 Mainboard

The mainboard was designed to be smaller than the Arduino Due that it borrowed its design from. One of the main features implemented along with the smaller size was the addition of a mating header that would plug directly into the 4D Systems LCD screen.

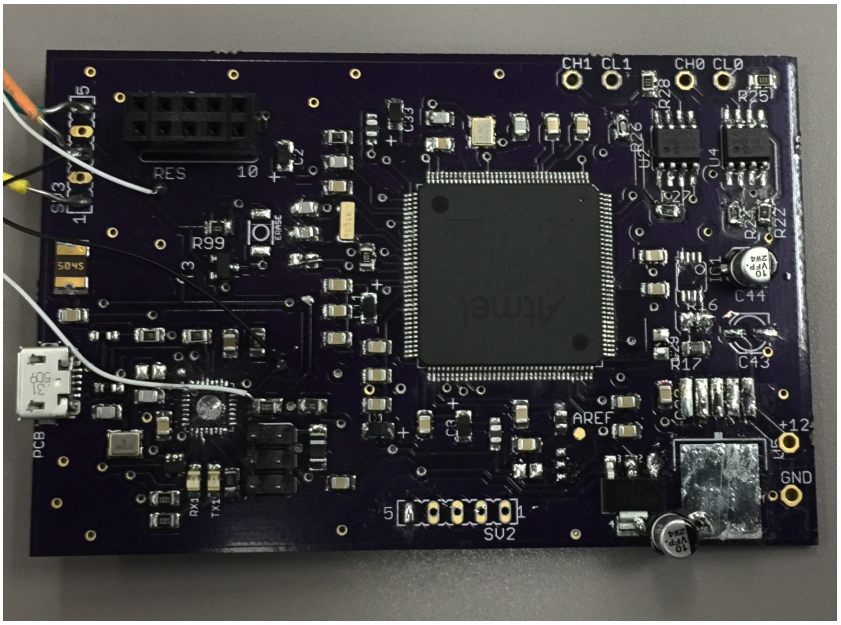

Figure 3: Final Board with Components

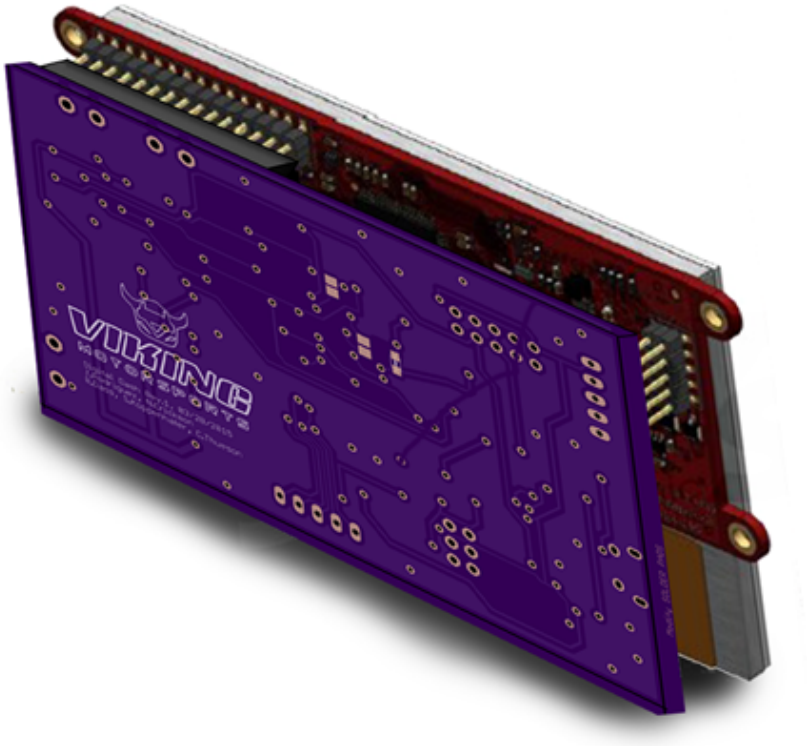

Figure 4: LCD and Board Mating

4.1.1 Schematic

The board schematic and layout were created in EagleCAD. The majority of the ATSAM3X and ATMEGA16u2 functionality were borrowed from the open source Arduino project. [2]

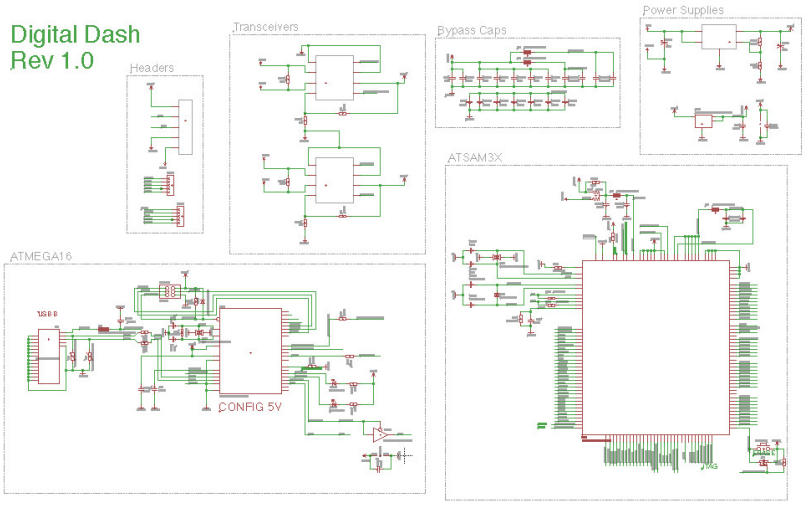

Figure 5: Mainbaord Schematic

The schematic is sectioned into functional blocks:

- Headers - Contains the pinouts needed to mate with the LCD screen.
- Transceivers - Contains the necessary resistances and ICs needed for the CAN transceivers for the ATSAM3X to receive CAN messages.
- Bypass Capacitors - A bank of bypass capacitors.
- Power Supplies - Contains the components needed to drive the MC33269ST Voltage Regulator.
- ATMEGA16 - Hntains the ATMEGA16u2 microcontroller that acts as a bridge between the ATSAM3X and the USB programming port.
- ATSAM3X - This block houses the main functionality of the digital dashboard.

4.1.2 Power Supply

The power supply was implemented using a MC33269ST voltage regulator and the necessary components to drive it. The power supply was designed to be used on either a 5 volt or 12 volt input supply to accommodate multiple systems. The power supply input voltage can be changed easily by resoldering a single

connection on the wiring harness from the mainboard to the Deutsch connector. Since the Viking Motorsports Team had a 5 volt source available from the GEVCU we decided to set up the wiring harness for 5 volts.

4.2 Enclosure

The final enclosure was created using the drawing software, Inkscape, and printing to the full spectrum laser cutter in the Electronics Prototyping Laboratory (EPL) at Portland State University. Once the enclosure was cut out of acrylic the edges were shaved down with a belt sander in order to fit into the small triangle made by the steering column support in the vehicle. In order to increase the water resistance of the enclosure the outside was coated with Plasti Dip.

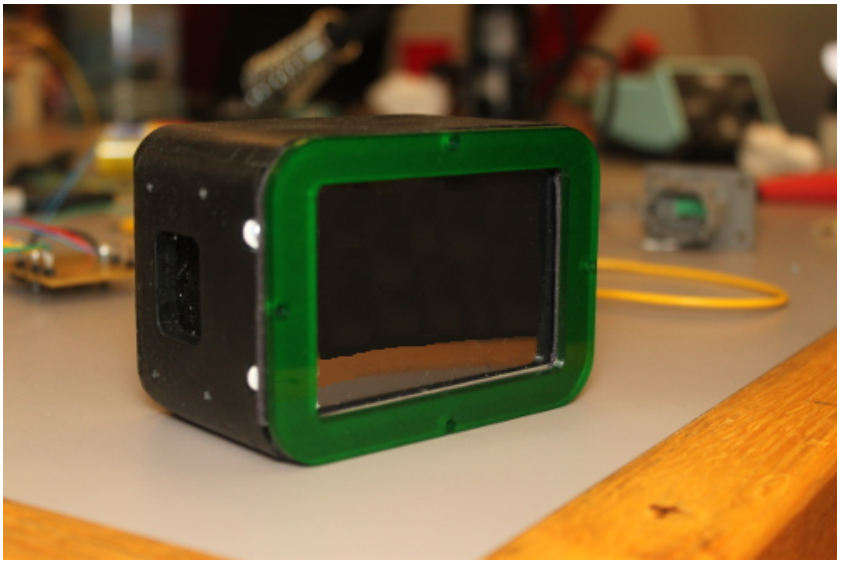

Figure 6: Enclosure

To get a flush fit on the electric vehicle, the design team also took over the task of improving the entire dashboard layout. The old dashboard was a sheet of metal that had been used for the previous two iterations of the electrical vehicle. Since the digital dashboard design team had created the enclosure for the digital dashboard, we knew the exact dimensions of the spacing we needed to get our screen to fit. We cut the dashboard out of bamboo and added an engraving of

the Viking Motorsports Logo.

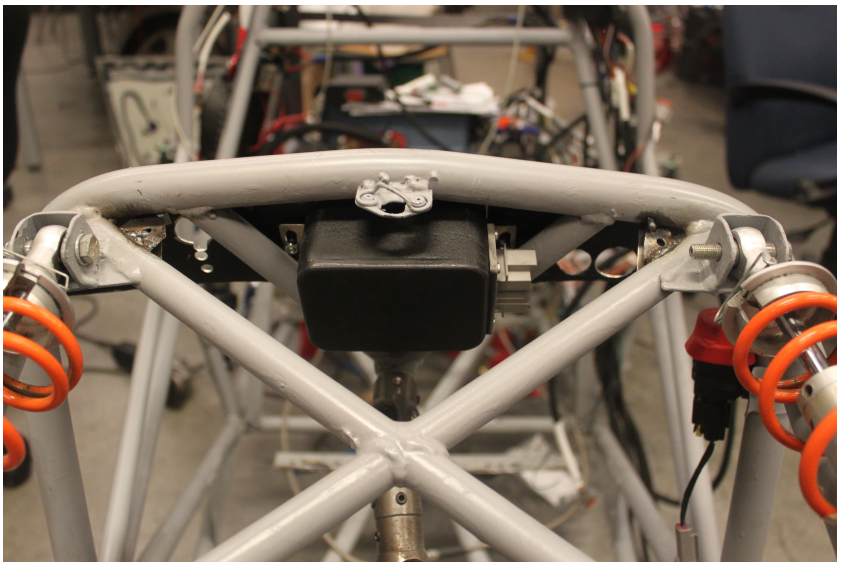

Figure 7: Limited mounting space

Figure 8: Digital Dashboard Integration

4.2.1 Deutsch Connector

The Viking Motorsports electric vehicle is racing in the Formula SAE competition. The design team decided that using SAE standards for communication protocols and wiring harnesses would continue the trend of using

automotive standards on the project. The Viking Motorsports Team had a surplus amount of Deutsch connectors which are an SAE standard. We implemented a Deutsch connector into the enclosure with a wiring harness that connects the pins to the mainboard inside the enclosure.

1. GND
2. 5V
3. RMS_CAN_LO
4. RMS_CAN_HI
5. BMS_CAN_LO
6. BMS_CAN_HI
7. IMD_ERR
8. NC

Figure 9: Deutsch Connector Pinout

4.3 Software

4.3.1 Mainboard

The source code for the mainboard was written in the C programming language using the Arduino IDE. The Due_Can library from Collin80[4] on Github was used to interface with the vehicles CAN busses. The 4D Systems Arduino library was used to interface with the 4D Systems LCD.

The software is fairly simple. In the main loop of the program, the screen is constantly being updated with the most recent data between CAN interrupts. Interfacing with the screen is made easy by using functions defined in the 4D System Arduino library. The main loop also checks the IMD LED line to see if it goes high. If it does go high, we know there is an IMD error and we can change the IMD image to red (which indicates an error).

CAN messages are received using interrupts set up by the Due_Can library.

When a CAN message is received, an interrupt handler function is called for the specific CAN controller. This handler function then calls the specific function that interprets the CAN message that was sent in. During interpretation, the function checks for any error/warning conditions that could occur with the specific subsystem. A data structure is then updated that holds the warnings/errors values for the screen. At the end of every CAN message handler function, the interpreted data is stored in a structure that holds all of the values being displayed on the screen.

4.3.2 Graphical User Interface

The graphical user interface (GUI) uses a black background to maximize contrast and has three large icons in the upper portion representing the insulation monitoring device (IMD) status, the battery management system (BMS) status and the RMS (motor controller) status. If an error is detected for any of these three systems, the corresponding icon will change from the color green to red. Below these icons, gauges for motor torque and amperage are present. Finally, a battery level indicator is also present to the left of the gauges.

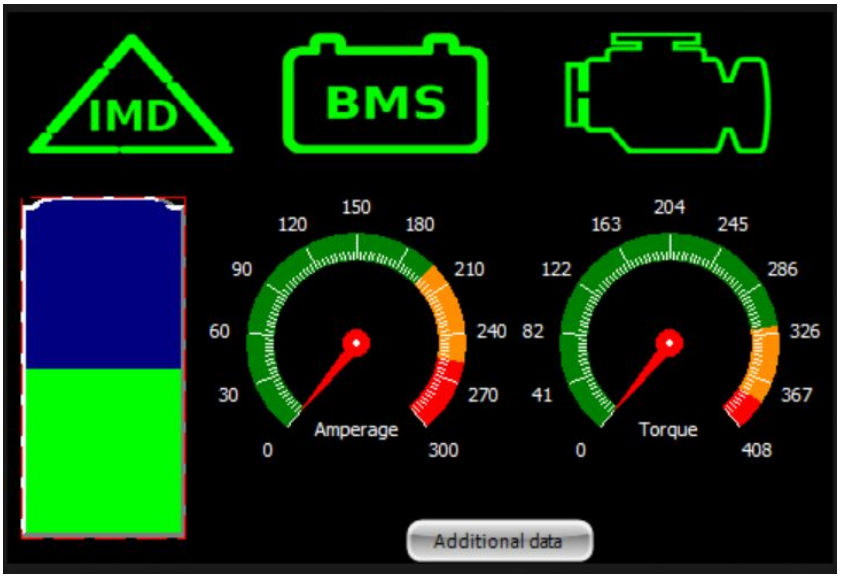

Figure 10: GUI Main Screen

The GUI was designed with a minimalistic approach in order to reduce clutter

and to give the digital dashboard a professional feel. To compliment the simple layout of the main screen, each system status icon has a sub-menu activated via the touchscreen that displays specific error messages for each system. On the main screen of the GUI there is also an Additional Data button, that when pressed, displays additional information not critical to the driver.

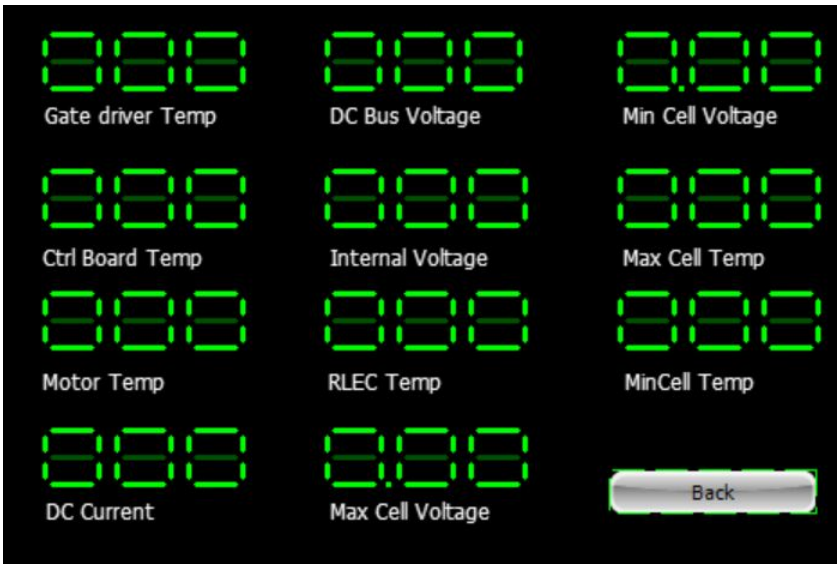

Figure 11: GUI Debug Screen

5 Explanations

5.1 Protocols Utilized

5.1.1 CAN Bus

The Controller Area Network or CAN bus is one of several central networking protocols that are used in vehicles that do not have a host computer. Companies are starting to move towards vehicles that are more technologically advanced and equally more environmental friendly. With these advances, all the individual working modules in the vehicle need to be connected so that they can communicate with each other. For example, the engine reports the vehicles speed to the transmission, which in turn must tell other modules when to shift gears. To be able to connect all these different modules together to work efficiently and

effectively a central networking system becomes necessary.

Having all the modules connected together into one system has numerous benefits. The most significant benefit is the price to performance ratio that it offers. This makes the current CAN bus technology the most reliable and cost effective choice for the automotive companies. If a major event occurs in one of the modules it is shown in the main controller and will allow the rest of the modules to take the proper actions. If one of the modules of the CAN bus happens to fail other modules are not affected unless they are directly related to each other. Diagnostics of the vehicle using the CAN bus becomes easier as well. It can pinpoint what has cause the module to fail and fix that module instead of tearing the whole vehicle apart.

5.1.2 Serial

Serial communication streams data one single bit at a time and was used to communicate with the LCD screen from the microcontroller. Specifically, the 4D Systems LCD screen used TTL serial to communicate with our microcontroller. Common examples of serial communication are USB and Ethernet.

5.2 Tools Used

5.2.1 Arduino IDE

The Arduino IDE was used to develop the software used on the microprocessor. It was also used to program the microprocessor.

5.2.2 Eagle CAD

EagleCAD was used to create the hardware schematic as well as the custom printed circuit board layout.

5.2.3 4DSystems Visi-Genie

The GUI portion of the LCD was created using 4D Systems Workshop software. This IDE integrated nicely with the Arduino IDE with an external library.

5.2.4 Hardware tools

Creating the wiring harness needed tools such as wire strippers, wire crimpers, multi-meters, oscilloscopes and soldering irons.
Creating the enclosure and the demonstration unit included using the laser cutter in the Electronics Prototyping Lab (EPL) at Portland State University. Creating the first prototype PCB required the use of the LPKF PCB Router in the EPL. Creating the final PCB required the use of the reflow oven in the EPL as well as the surface mount soldering station.

6 Testing

6.1 Testing Objectives

6.1.1 Testing Purpose

The following test plan documentation supports the following objectives:

- Identify existing project information and the software and hardware that should be tested.
- List the recommended test requirements (high level).
- Recommend and describe the testing strategies to be employed.
- Identify the required resources and provide an estimate of the test efforts.
- List the deliverable elements of all tests.

6.1.2 Testing Scope

This test plan describes the integration and system tests that will be conducted on the Viking Motorsports digital dash following the integration of subsystems and components identified in the Design Considerations for prototyping document.
The purpose of this test plan is to test the feasibility and performance of the Digital Dash. It is critical that all subsystems be tested before integration into

the system so as to ensure a proper user experience when using the digital dash is integrated into racing vehicle.
The following modules and interfaces will be tested:

- CAN message interpretation
- Data transmission to LCD
- GUI on LCD screen

Critical performance measures to test are:

- Minimal time to capture and interpret CAN data
- Time to display data on LCD screen after data interpretation
- Initialization time before the digital dash is ready to use

6.2 Requirements for Test

The lists below identify items (functional requirements, non-functional requirements and use cases) that have been targeted for testing. This listing represents what will be tested.

6.2.1 Function Testing

- Project Requirements: The system must be powered by a 12V source
- Data from CAN network is being relayed to Arduino Due for analysis
- Data from Arduino Due is being sent to LCD
- Data is being properly interpreted and displayed in GUI on LCD

6.2.2 User Interface Testing

- LCD Screen must be viewable in sunlight
- Project Requirements: "The digital dash should be easily readable, big fonts, non-cluttered display."

- The following variables must be displayed on the LCD screen
  - Main battery voltage
  - PM100 temperature
  - Max cell temperature
- The following warnings must be displayed on the LCD screen
  - Main battery voltage low
  - PM100 temperature high
  - Max cell temperature high
- The following errors must be displayed on the LCD screen
  - Battery voltage below limit
  - PM100 temperature above limit
  - Max cell temperature above high
  - BMS error
  - Insulation Monitoring Device
  - PM100 Error

6.2.3 Performance Testing

Project Requirements: "The unit must boot up in less than 30 seconds."

6.2.4 Load testing

Verify the system response when CAN network is heavily loaded.

6.2.5 Ruggedized Testing

- Requirements Document: "The unit must be weatherproof."
- Ruggedized to survive one season of racing.

6.3 Test Strategy

The test strategy is the recommended approach to test the hardware and software. The previous section described what will be tested. This section describes how it will be tested.

6.3.1 Function Testing

The goal of these tests is to verify proper data acceptance, processing and transmission. This testing is based on black and white box techniques. These tests are to check that modules can handle typical situations.

Test Objective:	Ensure proper data entry, processing and transmission.
Technique:	Execute each use case, case flow, or function, data, to verify the expected results occur when valid data is used.
Completion Criteria:	All planned tests have been executed. All identified defects have been addressed.
Special Considerations:	N/A

Table 4: Function Testing

6.3.2 User Interface Testing

The goal of these tests is to verify proper data acceptance, processing and transmission. This testing is based on black and white box techniques. These tests are to check that modules can handle typical situations.

Test Objective:	Verify the following: Ensure the LCD screen displays information properly on the screen and is visible to the driver. Verify that screen is readable in sunlight.
Technique:	Predefined messages are broadcast over the CAN bus and the corresponding information should be displayed onto the screen. Take the unit outside and see if screen is readable.
Completion Criteria:	All user interface modules functioning properly. All identified defects have been addressed.
Special Considerations:	If visibility of screen in sunlight is sub-par a new type of screen will need to be sourced and all interface testing will have to be repeated.

Table 5: User Interface Testing

6.3.3 Performance Testing

Performance testing measures the transaction time and other time sensitive requirements. The goal of of performance testing is to tune the system to ensure that interaction with the system is not excessively slow and doesn't use excessive energy.

Test Objective:	Validate system response time for designated transactions.
Technique:	Measure the time it takes to perform predetermined operations to ensure they complete within a satisfactory time range.
Completion Criteria:	Completion of operation is found to be satisfactory based on predetermined time range.
Special Considerations:	N/A

Table 6: Performance Testing

6.3.4 Load Testing

Load testing measures the ability of the system to function properly under different workloads. The goal of the load testing is to determine and ensure the system functions properly beyond the expected maximum workload.

Test Objective:	Validate system response time for designated operations.
Technique:	Saturate the CAN bus with data to determine if data can properly be captured and processed.
Completion Criteria:	Data still properly displayed on screen without system locking up or becoming unusable.
Special Considerations:	N/A

Table 7: Load Testing

6.3.5 Ruggedness Testing

Ruggedness testing verifies the operation of the unit while meeting Formula SAE Rules requirements and sponsor design requirements.

Test Objective:	Verify unit adheres to requirements of Formula SAE article 7 EV7.33. Ensure meets any additional ruggedness requirements.
Technique:	Physical tests to ensure that enclosure can function after being exposed to water for 240 seconds.
Completion Criteria:	System is still operational after all tests have been completed.
Special Considerations:	Exposed to water means rain like conditions not immersion.

Table 8: Ruggedness Testing

6.4 Testing Equipment

Equipment needed to carry out testing:

- Windows compatible PC
- Viking Motorsports electric race car
- GEVCU Programming Software
- Spray hose

6.5 Resources

Role	Minimum Resources Recommended	Specific responsibilities/ Comments
Test Designer	Chad Thueson	Identifies, prioritizes, and implements test cases.
System Testers	Chad Thueson, Sean Koppenhaffer, Jaime Rodriguez, Noah Erickson, Rishal Dass	Executes tests, log results, recover from erros, document defects
Designer	Chad Thueson	Identifies and defines the operations, attributes, and associations of the test classes.
Implementer	Chad Thueson	Implements and unit tests the test classes and test packages.

Table 9: Resources

7 Conclusion

The digital dashboard performs as intended when simulated with the demo program that we created. Instead of testing all functionally of the screen, the demo program focuses on only the most important functions.
At the time of this report the integration of the digital dashboard and the electric vehicle is underway.

8 Licensing Information

9 Glossary

- BMS - Battery management system
- CAD - Computer aided design
- CAN - Controller Area Network
- EPL - Electronics Prototyipng Laboratory
- GEVCU - Generalized Electrical Vehicle Control Unit
- IMD - Insulation Monitoring Device
- LCD - Liquid crystal display
- LED - Light emitting diode
- PCB - Printed Circuit Board
- RMS - Rhinehart Motion System
- SAE - Society of Automotive Engineers
- VCU - Vehicle control unit

10 References

References

[1] *2015 Formula SAE Rules*. 2014, September 17. URL: http://students.sae.org/cds/formulaseries/rules/2015-16_fsae_rules.pdf.

[2] *Arduino Due*. Arduino. URL: http://www.arduino.cc/en/Main/ArduinoBoardDue.

[3] *Atmel — SMART ARM-based MCU Datasheet*. Tech. rep. SAM3X/SAM3A. Rev. Atmel-11057C-ATARM-SAM3X-SAM3A-Datasheet_23-Mar-15. Mar. 2015.

[4] Collin Kidder. *Due CAN Library*. URL: https://github.com/collin80/due_can.