Critical Design Review Report

ECE 492 - Spring 2016

Table of Contents

- Table of Contents
- Acceptance Test Plan
- Safety Plan
- System Design
  - Overall Schematic
  - Grounded Low Voltage
    - Overview
    - GLV Power/Scada Interface/Safety Loop
    - TSI
    - CAN Bus
  - Tractive System Voltage
    - Overview
    - Displays, Indicators and Controls
    - Data Acquisition
    - Charging and Low Current Output
  - VSCADA
    - Overview
    - Design Details
  - Software Architecture
    - HTTP/WS Server
    - Document Logger
    - Maintainability and Configuration
    - Displays, Indicators and Controls
  - Motor Characterization and Dynamic Model
- Analysis of System States
- Analysis of Communication Links
  - CANbus Communication
  - USB Communication
  - Lafayette Network Communication
  - I2C

Interface Control Specifications
QA Test Plan
Cost Analysis
Schedule
Appendix A: Schematics
Appendix B: Bill of Materials
Appendix C: Fabrication Specifications

Acceptance Test Plan

The ATP is a document detailing how the design is determined to be successful in meeting the requirements laid out in the SOW. It contains items that are split between physical verifications and written documents and analysis confirming requirements. Each requirement has strict Pass/Fail criteria marked to indicate key components that need to be signed off on the signify a complete. The actual signatures will be included as part of the ATR delivered a day after FDD. A copy of the ATP is attached to this document.

Safety Plan

This is the safety plan as outlined in ATP. Attached is a safety plan document further details.

- All documents and code have a location for the name of the creator, and the name of a reviewer that was not directly involved in the creation of the document.
- The creator creates the document in accordance with GPR005 in the SOW.
- The reviewer reviews the content of the document with respect to GPR005.
- Safety Plan was accepted by the team and the ECE Director of Laboratories
- Documents are reviewed by person not directly involved in their creation. Attach a signature sheet to ATR to document a reviewers approval. The sheet is comprised of a table similar to the following:

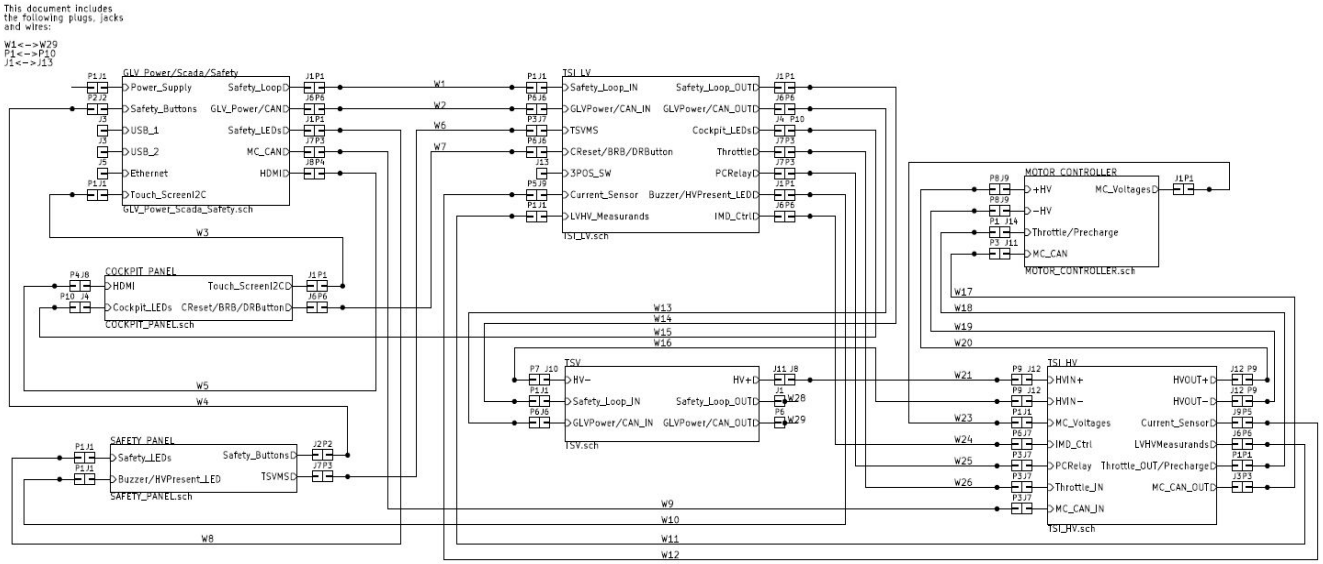
Document File Name	Creator	Reviewer

- Document reviewer writes down any revisions that are suggested and submits them to the professor to show that it was reviewed in earnest.
- A safety plan is generated for any subsystem operating with potential differences greater than 30V. It includes:
    - Analysis confirming any parts operating with greater than 30V are appropriately chosen.
        - Specifically for TSI and TSV
    - A process for properly closing the subsystem prior to operation.
    - A list of precautions that are followed to ensure the safety of participants.
    - ECE Director of laboratories, or an alternate designated by him, reviews and accepts all electrical safety plans before any such assembly is closed or tested. The Director or alternate must sign a label that is placed on the assembly. The label includes the following:

Approved Assembly:_____
File Name of Safety Plan: _____
Approval Signature:_____ Date:_____
Approval Expires(Date, if Applicable): _____

System Design

Overall Schematic

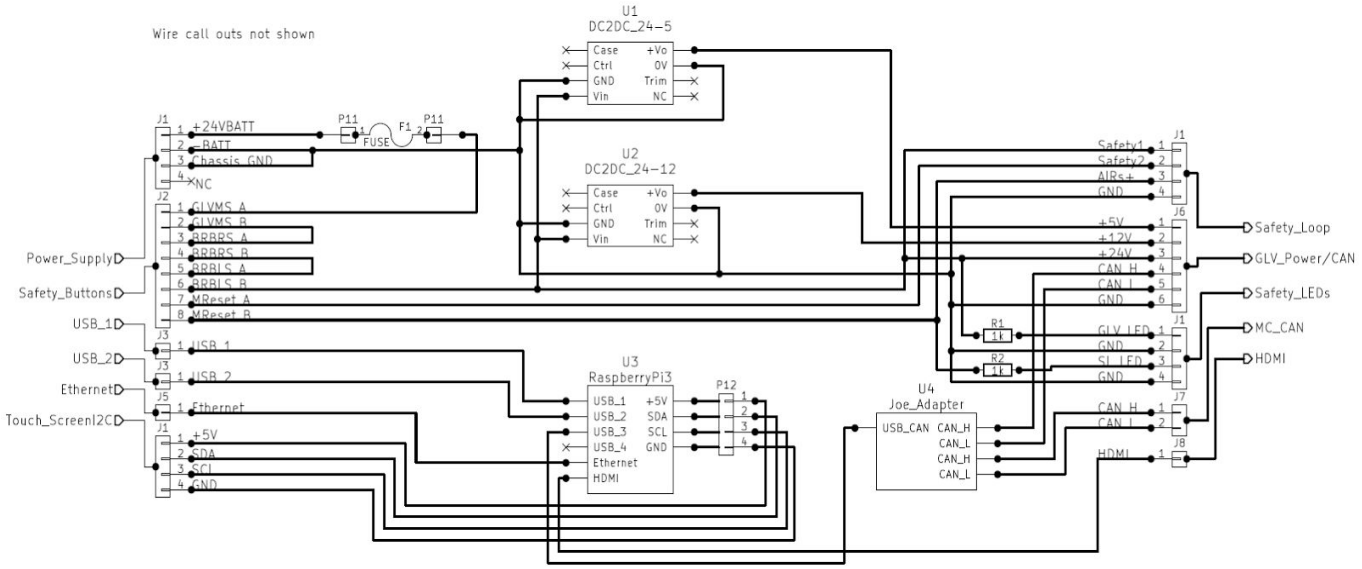


Grounded Low Voltage

Overview

The Grounded Low Voltage (GLV) system provides chassis-grounded low-voltage power to all low-voltage components on the car. This includes the safety loop, Tractive System Interface (TSI), John Gehrig Can Communication Board (JGB) and various other components.

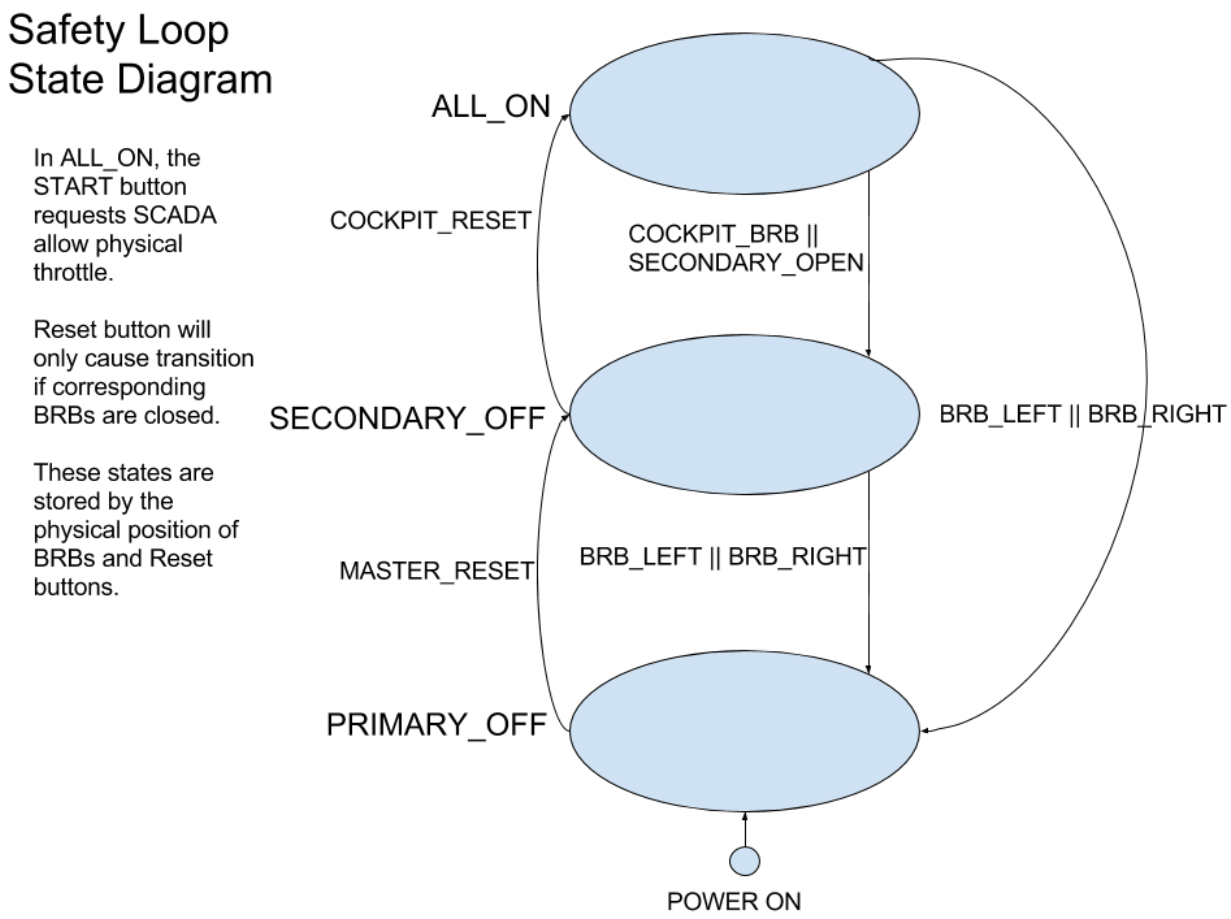
GLV Power/Scada Interface/Safety Loop



Safety

This system utilizes a four-wire, nested double-loop safety loop in order to ensure that the driver is able to disable the vehicle in case of an emergency and that the system is able to disable itself in the case of the TSV being shorted to ground. In this design, there is a primary safety loop, which contains the GLV master switch (GLVMS), a left and right side Big Red Button (BRB) that can each cut power to the overall GLV system, and a reset button to re-energize the system after a BRB is reset; this system is not accessible to the driver.

The secondary safety loop contains the cockpit BRB, the cockpit reset button, the insulation monitoring device, and the TSV master switch (TSVMS); a fault in this loop does not trip the primary loop, but when the primary loop is disabled, the secondary loop is as well.



When GLV power is initially hooked up to the system, nothing will actually receive power until the GLVMS is turned on. From there, the system starts with both loops uninitialized and unpowered. To enable the primary safety loop, the left and right BRBs must be set to their unpushed states and the master reset button must be pressed.

Once in the secondary state, the system can regress to the primary state via a press of either the left or right BRB. Also, for the system to proceed to the final state, the cockpit BRB must be set to its unpushed state and the cockpit reset button must be pressed.

In this final state, the AIRs can be engaged upon the TSVMS being turned on. The system can also regress to the secondary state if the cockpit BRB is pressed or if the system JGB disables the secondary system due to a SCADA or IMD fault. Additionally, it can return to the primary state if either the left or right BRB is pressed.

GLV Power

The GLV Power/Scada Interface/Safety Loop System Box is responsible for producing the Car GLV system's voltage(s). The current layout supports 5 volts, which is responsible for powering the scada computer(Raspberry Pi 3) and the VCI (Raspberry Pi Touch Screen 7") in addition to

several other systems. The GLV system also supports 12 volts. The initial system design was done with only 12 volts as GLV power and continues to exist to support these systems.

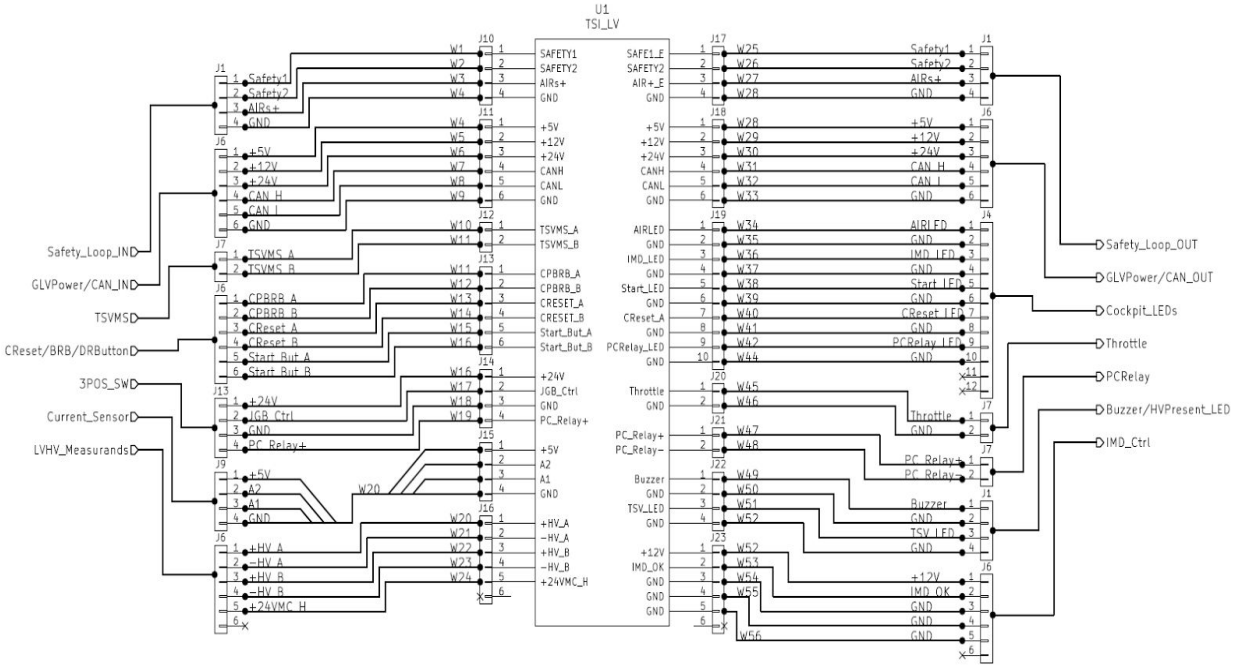
Scada Interface

The Scada Interface connections in the GLV Power/Scada Interface/Safety Loop System Box include CAN via a USB to CAN converter. The system also supports two USB inputs and Ethernet, which are access ports for access to the Raspberry Pi for troubleshooting. The system also has an HDMI output, which can be used for supporting a monitor. However, the HDMI connection is designed to support the vehicle to computer interface and must connected to the Cockpit Panel for VCI usage.

TSI

The TSI is a system that allows tractive power to flow into the accumulator packs, given the safety loop is not tripped. It is here that the IMD would be housed, as the IMD monitors both GLV and TSV. However, the sections allotted for the GLV and TSV are designed such that they remain galvanically isolated from one another, as per R003 in the SOW.

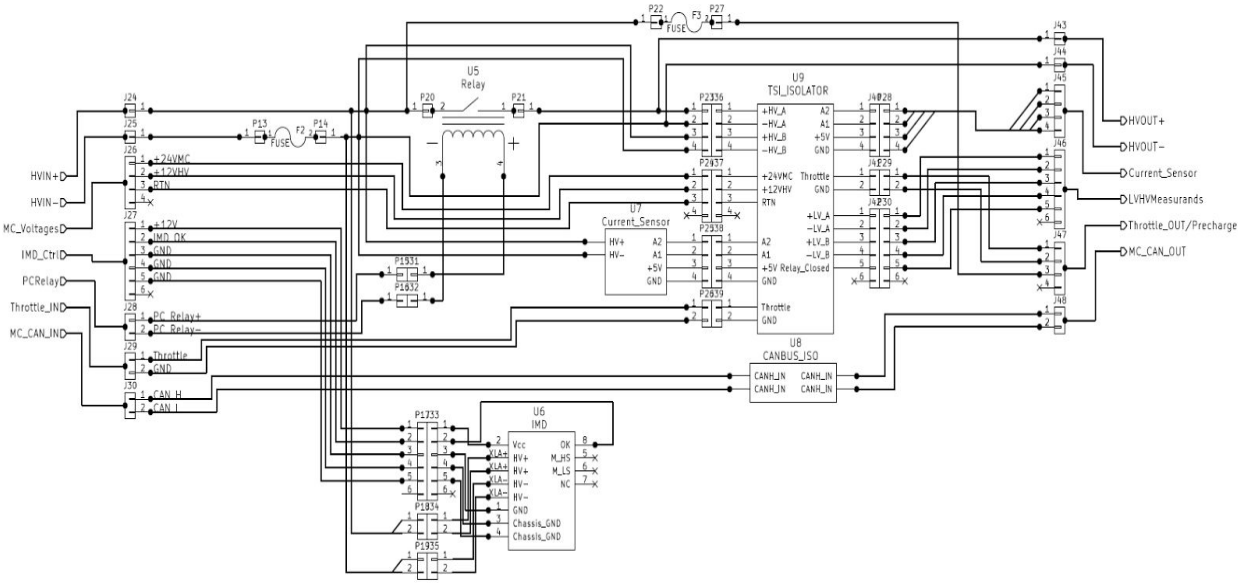
TSI Low Voltage



The GLV team made the decision to completely separate the TSI high and low voltage systems. The TSI low voltage includes a single pc board. This is a modified version of a “JGB,” which has been repurposed and renamed JGB^3. Much of the interface logic for the sensor and safety loop interfaces are added onto this board as well as the connectors for routing wires so external routing considerations are heavily reduced. The TSI low voltage has three basic functions determined by a three position switch: on-where the pre-charge is always closed,

auto-where the pre-charge relays can be closed by scada or user input, and finally off-where the pre-charge relay is always open.

TSI High Voltage



The TSI high section is responsible for isolating the high voltage systems from the grounded low power and logic signals. TSI high voltage, at its core, contains the IMD and the pre-charge relay. The TSI high voltage also, as its name would suggest, is responsible for converting HV side signals to LV so they can be processed by the JGB^3. Namely the TSI HV outputs the equivalent LV signals: current data, two HV+ HV- differential signals, CAN to the motor controller and the motor controller coil side relay voltage. The system will contain a pc board, “TSI_ISOLATOR,” that performs this task.

CAN Bus

The majority of the vehicle CAN bus shall also operate on GLV, although the section into the motor controller operates on the TSV. However, the two sections are separated and interfaced with a CAN bus isolator.

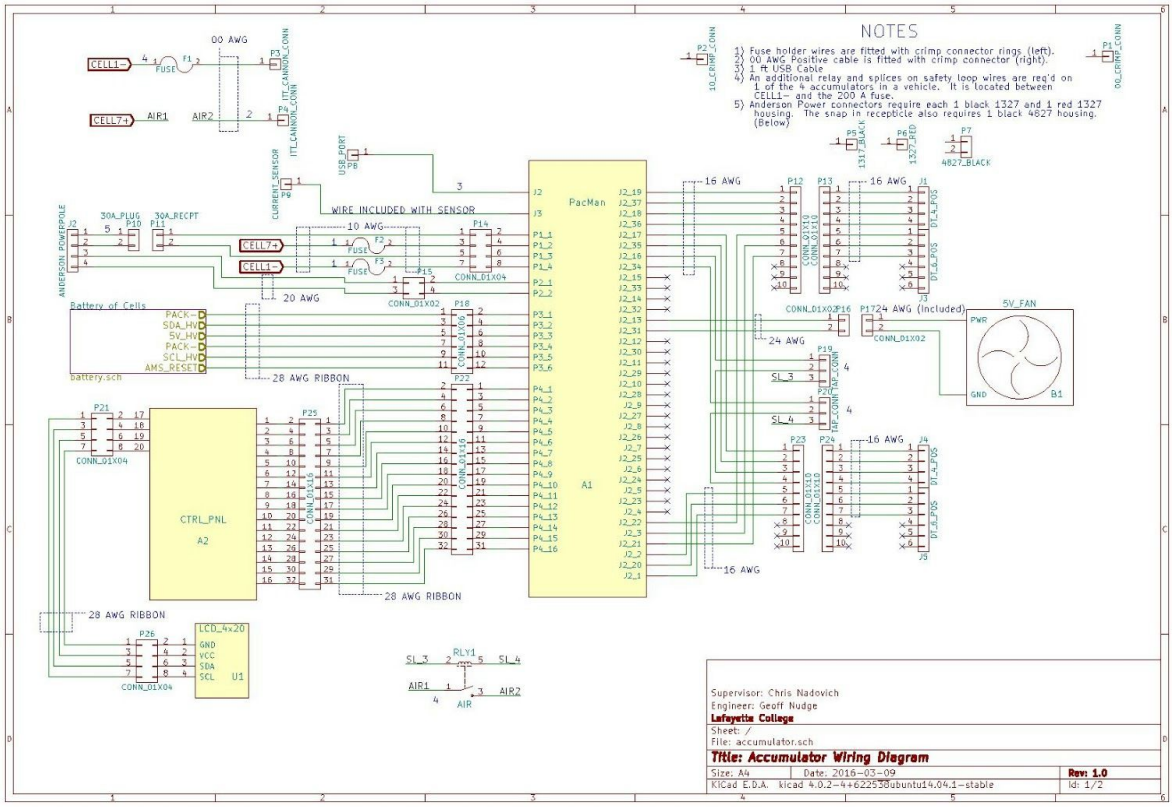
The primary devices that would communicate on this bus would be the SCADA computer, the Curtis motor controller, PackMan, and JGBs. This bus allows for these devices to transmit information regarding the state of the accumulator packs, the safety loop, and so on.

The CAN bus that would be present in the car would utilize a two-wire, shielded, twisted pair set of cables to relay messages.

Tractive System Voltage

Overview

The Tractive System Voltage is provided by four accumulators in series to provide the power necessary to operate the motor. An accumulator is comprised of a battery of 7 LiFeP04 cells (3.2 V nominal) connected in series. Below is a schematic of a single accumulator (see also Appendix A). A Bill of Materials is included in Appendix B.

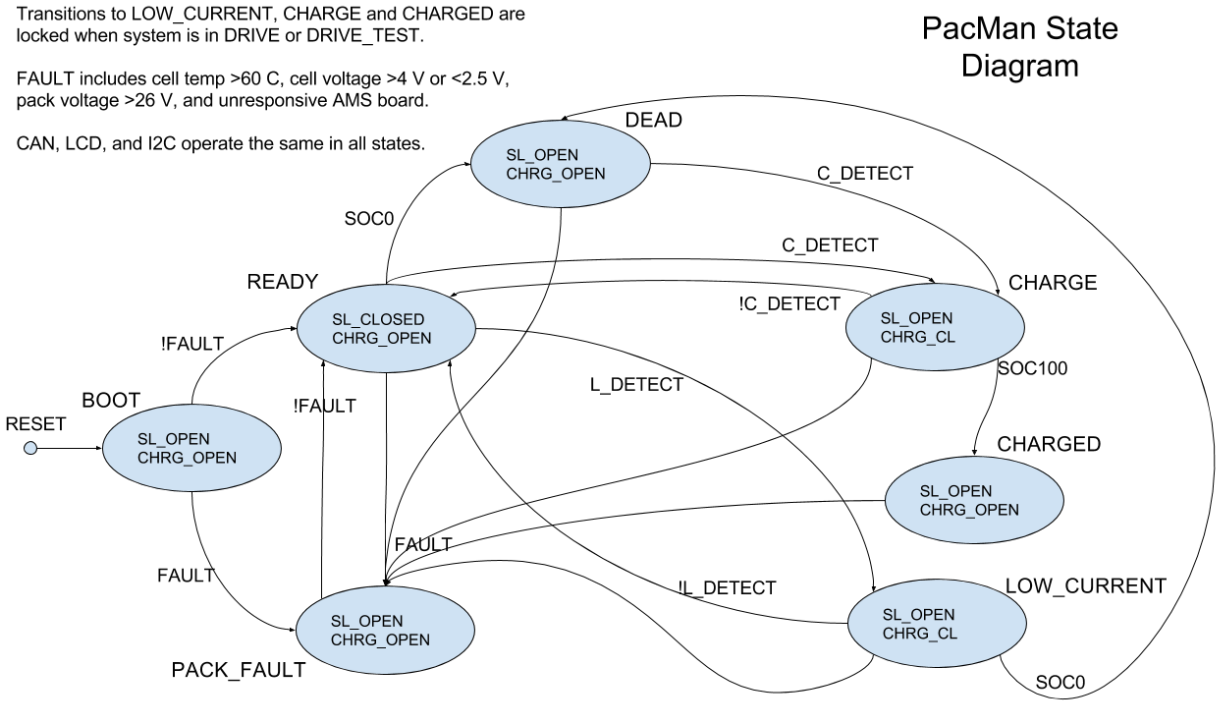


Connections on the accumulator are, a 4 pin safety loop in, 4 pin safety loop out, 6 pin GLV/CAN in, 6 pin GLV/CAN out, a 4 pin low current port, and 2 high current ports.

The accumulator high current output is available through ITT Cannon connectors. The positive connector is attached to the end of a 00 AWG cable to facilitate connection to other accumulators and the motor controller. Accumulator voltage is present only while the safety loop is closed.

A low current output is also available through an Anderson Power connector. This output is limited to 20 A. Charging is also accomplished through this connector, and is similarly limited to 20 A and 30 V.

The functioning of both ports is controlled by a Pack Management Computer (PacMan). A schematic is included in Appendix A (a Bill of Materials is also included in Appendix B). The PacMan utilized an AT90CAN128 Atmel microcontroller. Its system states follow the state transition diagram shown below.



On reset the computer checks for faults. Faults are temperature over limit, voltage out of range, and an unresponsive AMS board. If any of these are present, the pack fault state is transitioned to, and the safety loop relay is left open. If none are present, the pack transitions to the READY state. In this state the safety loop relay is closed and the accumulator is able to provide power through its high current outputs if a closed safety loop is connected the safety loop in port of the accumulator.

From the READY state, a transition to the CHARGE state occurs if a charger is detected. If the accumulator is fully charged while in the CHARGE state, a transition the CHARGED occurs. The safety loop relay is opened in these two states and the charging relay is closed in CHARGE.

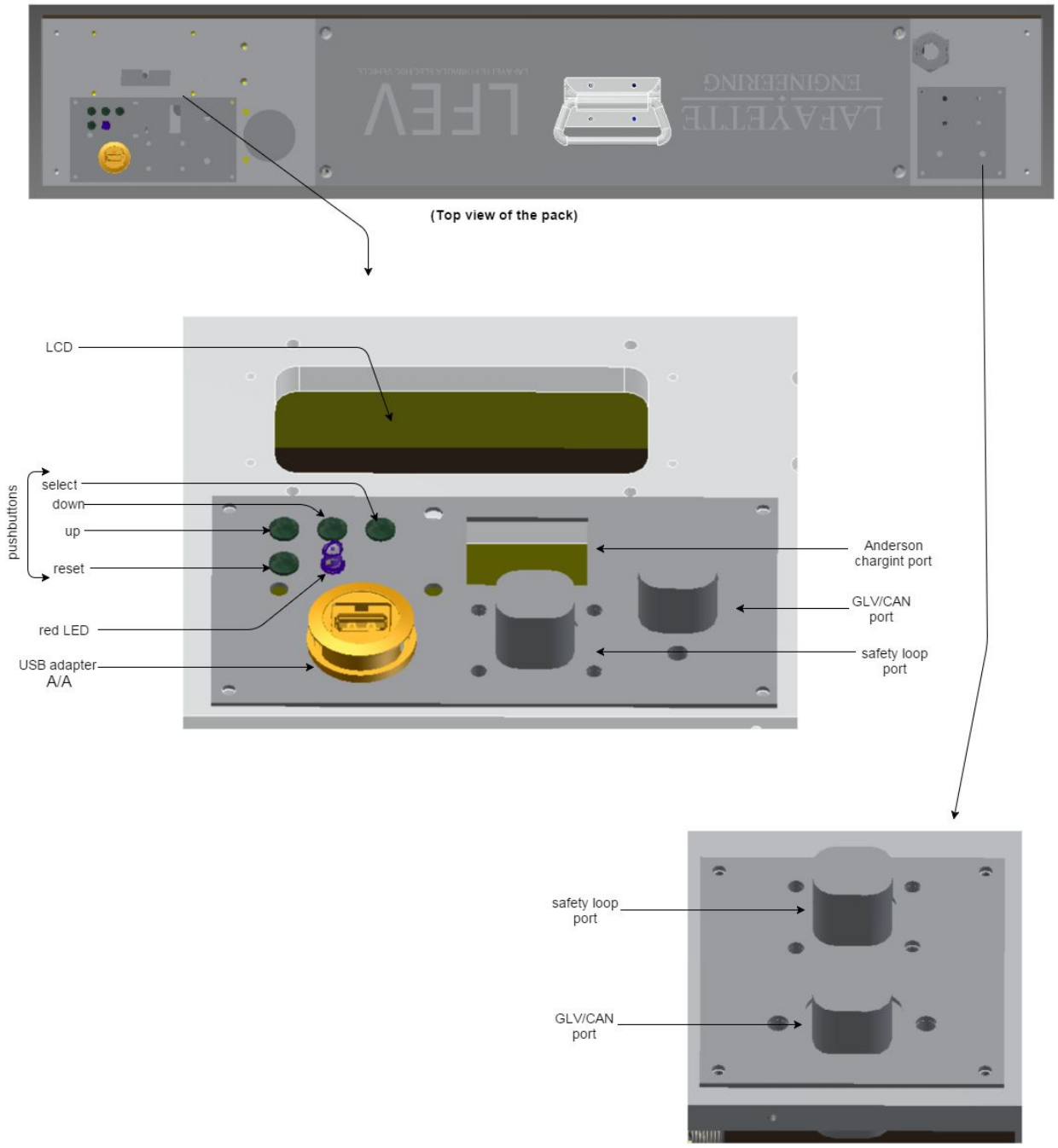
From READY, a transition to LOW_CURRENT occurs if a low current output is detected. Similar to the charging state, the safety loop relay is opened and the charging relay is closed.

All states have transitions to the FAULT state if any of the possible faults are detected. The computer transitions to READY when the fault is no longer present.

Displays, Indicators and Controls

LCD display, controls and indicators are going to be provided on top of the pack.

As illustrated below, controls of the pack are four pushbuttons and an indicator is the red led. Four push buttons are integrated the LCD display. One is for navigating up through the options, one is for navigating down through the options, one is for choosing/selecting and one is for reset. The red LED is an indicator for when AIRs are closed (when the pack is alive).



Managed by PackMan, the LCD will be able to display pack voltage, pack current, cell voltage, cell temperature, state of charge, cell balancing state, charging state, charging history, discharge history and safety loop state. It will also have functionalities of going into sleep mode and choosing calibration factors where they are necessary.

Below is an abbreviated layout of the tree structure that will be implemented for the LCD:

T> (Top level)
- CHST (charging history)>
  - DEL
- C1>
  - CAL>
    - VOFF (Voltage calibration offset)>
      - 1 (Digit 1)
      - 2
      - 3
    - VSLP (Voltage calibration slope)
    - TOFF
    - TSLP
- C2
- C3
- C4
- C5
- C6
- C7
- CAL1>
  - VOFF
  - VSLP
  - AOFF (Amps calibration offset)
  - ASLP (Amps calibration slope)
- CAL2>
  - (Calibration factors associated with state of charge may be accessed here.)

Example outputs of the menu tree structure are shown below:

ACTION	**OUTPUT**	**POSSIBLE OUTPUTS**
Top Menu		
	STATE:RDY SOC: 78% PACV:24.5 PACA:157.4 SAFETY LOOP:CLOSED T> CHST	CHRG, CHRGD, LCO, FLT, DEAD RDY, BOOT CHST, C1, C2, C3, C4, C5, C6, C7, CAL1, CAL2,SLEEP
Down		
	STATE:RDY SOC: 78% PACV:24.5 PACA:157.4 SAFETY LOOP:CLOSED T> C1	

```
Select
          STATE:OK              OK, BYP
          C1_V:3.4 C1_T:47.3

          T/C1> T               T, CAL
Down
          STATE:OK
          C1_V:3.4 C1_T:47.3

          T/C1> CAL
Select
          VOFF:.002 TOFF:2.00
          VSLP:1.02 TSLP:0.99

          T/C1/CAL> V OFF       VOFF, VSLP, TOFF, TSLP, C1
Select
          VOFF:.002 TOFF:2.00
          VSLP:1.02 TSLP:0.99

          T/C1/CAL> VOFF
Select
          VOFF:.002 CURRENT
               .002 NEW
                123 DIGIT
          T/C1/CAL/VOFF> 1           1, 2, 3, CAL
Select and Up
          VOFF:.002 CURRENT
               .102 NEW         0, 1, 2, 3, 4, 5, 6, 7, 8, 9
                123 DIGIT
          T/C1/CAL/VOFF> OK?
Below are example outputs of the remaining menu types.
A Charge History Screen
          LAST DISCHRG%: 17
          LAST CHRG%:100
          CURRENT CHRG%: 78
          T/CHST> T                  T, DEL

CAL1
          VOFF:.021 AOFF:.205
          VSLP:0.97 ASLP:1.04
          KI:  2.54 KCC: 1.05
          T/CAL1> VOFF               VOFF, VSLP, AOFF, ASLP, CAL2, T
```

```
CAL2
          KSL1:.021 KSH1:.205
          KSL2:0.97 KSH2:1.04
          KSM :
          T/CAL1> KSL1              KSL1, KSL2, KSM, KSH1, KSH2,
                                              ASLP, CAL1, T
```

Data Acquisition

The PacMan computer monitors the overall pack current and voltage directly, and individual cell voltages and temperatures via I2C communication with the AMS boards (see spring 2015 design documents). Calibration factors for these values are stored on PacMan and may be modified with the control panel. PacMan also keeps track of accumulator state, state of charge, and the state of the safety loop relay on PacMan. All of these data are regularly sent via CAN frames to the VSCADA computer. The microcontroller communicates on the CAN bus through a Microchip MCP2551 CAN interface IC. More details on CAN communication are provided in the VSCADA section of this document.

Charging and Low Current Output

Charging and a low current output are facilitated by an Anderson Power connector on the control panel. Both are limited to 20 A and 30 V. There are six pins populated on the connector. Two provide the positive and negative contacts for the charger and provide power for low current output. Additionally there are two charge detect pins and two low current detect pins.

If charging is the desired behavior, a connector plugged into the port should have a jumper across the charge detect pins. Likewise, for low current output, a connector should have a jumper across the low current detect pins. These jumpers pull an input pin low on the NXP Semiconductors PCF8574A I2C expander. The microcontroller has access to this expander over I2C and can respond to the connector being attached. This allows the accumulator to avoid excessively discharging the cells with a low current output, and still allow charging through the same port.

In either case, when the port is being utilized, a relay on the PacMan computer closes allowing access to the positive and negative terminal of the accumulator. These connections are fused at the terminals with 25A blade fuses. Current flowing through the charge relay also flows through a 1 mOhm current sensing resistor that is monitored via a Kelvin connection by a Texas Instruments INA 226 current monitor. This IC also allows voltage sensing for the full accumulator voltage.

In the fall of 2015, an IEEE paper that explores a mixed estimation state of charge algorithm. This algorithm combines coulomb counting with a cell model to provide a robust and stable method of state of charge estimation. However, the details of the cell model must be

determined experimentally, and this process will be very time consuming. It is best left to a future research project.

This design will utilize coulomb counting, integrating the current flowing through the current sensing resistor to determine the increase in state of charge due to charging as well as the decrease in state of charge due to low current output and the operation of the PacMan computer.

To monitor current flowing through the high current output, a Ametes BBM-01 current sensor is attached to the 0.5 in by 1 in aluminum bar wire that attaches the negative accumulator terminal to the negative terminal of battery of cells. This sensor provides a differential voltage output that is available to the microcontroller over I2C through a Texas Instruments ADS1115 analog to digital converter.

While charging, the voltage of each cell is monitored by the AMS boards (see spring 2015 design documents), and communicated to PacMan via I2C. A cell will be placed in bypass mode when its voltage reaches 3.6 V, allowing other cells to continue to charge without overcharging. This is accomplished allowing some current to pass through a resistor attached to a heat sink instead of the cell. A 5V fan is allowed to run at all times while charging to maintain the ambient temperature inside the accumulator. Once any cell reaches 3.9 V, charging is considered complete. Both bypassing and completion of a charge cycle trigger an entry in a charge history stored on the microcontroller. This data will be accessible in debug via USB, and an abbreviated history is available on the LCD.

Should the accumulator become overly discharged the PacMan computer may not be able to close the charge relay. A 500 mA power source may be connected to the USB port to power PacMan so that the charge relay will close when a charger is connected.

VSCADA

Overview

The ultimate realization of SCADA is an onboard data monitoring and acquisition system. The fully realized version logs data monitored from the car and broadcasts the data to be viewed on a website. Relevant data will also be passed to a 7” touchscreen that will act as the dashboard. For the scada computer itself we have selected a Raspberry Pi 3 for its memory size, wide support, extensive documentation, price point, dedicated graphics hardware, and powerful extensibility. The Raspberry Pi 3 will be connected to sensors and controllers with a CANbus network. The Pi accesses this network via a USB2CAN adaptor.

Design Details

Previous VSCADA code exists but will be largely refactored in order to best accommodate the current requirements and the needs of future Lafayette teams. All UI and Displays will be accessible from anywhere via web page hosted on the VSCADA card. This web page will run a javascript app that improves the current GUI in every way. Any computer or mobile device will have full access to all controls and statuses. Also located on the site will be the dashboard, to which the dashboard screen will auto-navigate to.

The new system will also be designed to be far more sustainable, something not at all accomplished by the existing code base. The current code base is lacking in both clarity and documentation, aspects we hope to focus on. Teams years from now should have no issue looking at our code and figuring things out.

Software Architecture

The software at it's highest level is made up of three primary components.

1. A HTTP and WebSockets server.
2. A long term document-oriented logging system.
3. A system model

HTTP/WS Server

The HTTP/WS server is loaded and, by default, serves a minimalistic web application which dumps the entire system state. A configuration file can be used to serve files from a different directory, allowing a better user interface to be easily loaded. In the event that the system model cannot be created because of an invalid topology, this component is responsible for generating an error page. The state of the system model is transmitted to clients over WebSockets as JSON objects. For implementation details see "SCADA WebSockets API"

Document Logger

This component monitors the system model for changes in state--represented as dictionaries--and logs the changes as documents in a NoSQL database. The concept of a non-relational database can be frightening to some, but we believe this allows for simplicity of implementation and better normalization of data in our use case.

Message Bus

Although not mentioned above, the components communicate through a message bus. Any system can broadcast any kind of message to the other systems. This provides an elegant method of calling multiple handlers at once, and synchronizing the multiple concurrent entities. The broadcaster doesn't have to be aware of who is interested in the message, and listeners

don’t need to be concerned with where a message came from. In our case, the data flow is mostly unidirectional from the models to the logger and web api.

System Model

The system model is created on startup by looking at the `system_topology.yml` file. Details of the format of this file can be found in “SCADAd System Topology.” The system is divided into two classes of subsystem: ‘Virtual’ and ‘physical’. Physical subsystems have a real world counterpart and attempt to mimic the state of the counterpart. They are also associated with logic (methods) to control the state of the real world system. Virtual subsystems are where the system gets all it’s uniquely specific functionality. There is nothing saying a battery pack has to be part of a car, or that a car has to have a safety loop, until a virtual subsystem is written to manage that. Several virtual subsystems are proposed by this team. We attempt to separate concerns and provide clean, modular system, which will be easy for others to maintain. These vehicle specific subsystems are described below.

- **Battery Manager -** Aggregates all physical subsystems of type `BatteryPack` and broadcasts messages to all of them to coordinate global state.
- **Mode Manager -** This subsystem knows that the vehicle needs a power source. A BatteryManager or power supply could satisfy that. It knows that a software throttle should be used in maintenance mode, and the physical throttle used in drive mode. It knows that a safety loop should be present. Et cetera.

The Event Loop

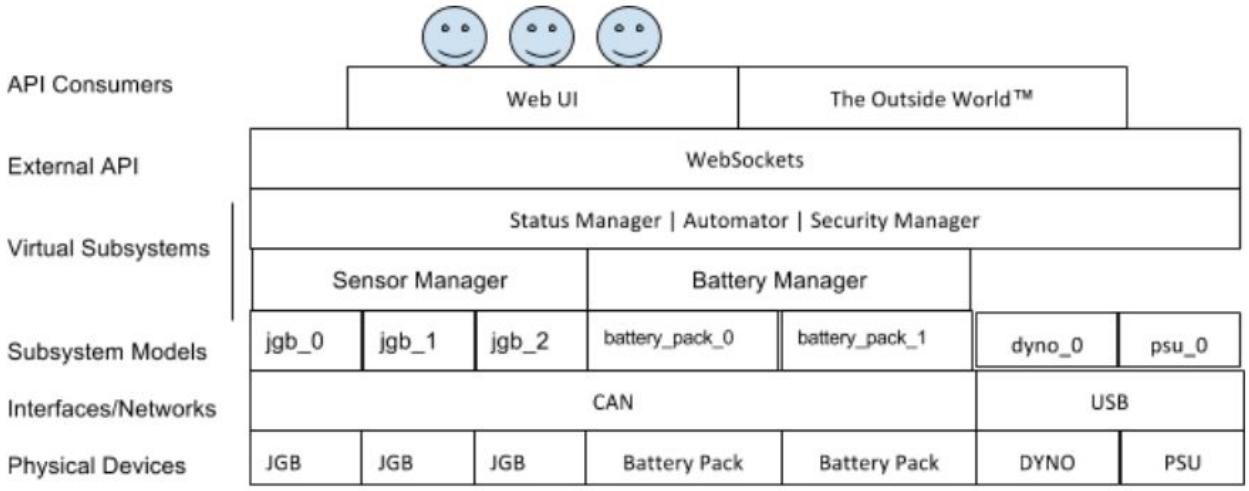
A system like this has a lot of things happening at once. In order to make it easy to express concurrency, the software uses Python3.5's asyncio library. Subsystems typically have a loop which waits for data from CAN, and then processes it. These loops are expressed as asyncio coroutines. They are cooperatively scheduled, which means that the currently executing routine will only change when it is explicitly allowed to. This makes the software more easy to reason about than if it used a threaded approach, which would mean these routines are preemptively scheduled. Python's GIL prevents any true parallelism, so there is no downside from not using true threads. Some blocking libraries may need to be executed in a separate thread, but those can be managed by the event loop.

To accomplish friendliness, we are planning an API approach. A python based “sensor API” is planned. A requirement of making the code sustainable is to make adding new modes, like drive demo, or sensors, such as GPS, as simple as possible. After this year it should not be necessary for any group to touch the core VSCADA code.

Necessary sensor readouts will be organized into managers which can be thought of similar to tabs. These managers will contain similar sensors and provide full readout and control of their parameters. The sensor data will be acquired over CANBus protocols and sent to the web server for interpretation. The

The webserver will also keep logs of all the data in a database. This database will be network transferable and physically transferable via SD card.

The webserver will also be capable of sending commands back to VSCADA in order to adjust parameters as needed. Parameter adjustment can be done manually or it can fed in using a script which will run automated tests for you.



Visual representation of software topology

VSCADA's modes of operations will be divided into three distinct modes which are detailed further in the Analysis of System States section. Other than these states the software is largely stateless. It doesn't care what mode or state it is in or even if it is being viewed, it will always be running monitor programs and logging data. The mode functionality is primarily to control what variables in the system are currently able to be controlled by the user.

Maintainability and Configuration

Looking at previous code, we determined that for the good of the project, system maintainability needs to be of primary concern. SCADA needs to be user friendly, easy to extend, and even easier to configure. As means to this we have elected to move as much of the system topology as possible to a configuration file.

The system topology is specified in a YAML configuration file. The file maps to a hash table and supports lists, numbers, booleans, strings, and comments, which make the configuration file self documenting. There will be no auto detecting hardware because then there would be no way of knowing if the system is completely online.

If the syntax of the configuration file is incorrect, the daemon will serve an HTTP 500 error page, to indicate that the system model cannot be created. If the user neglects to describe parts of

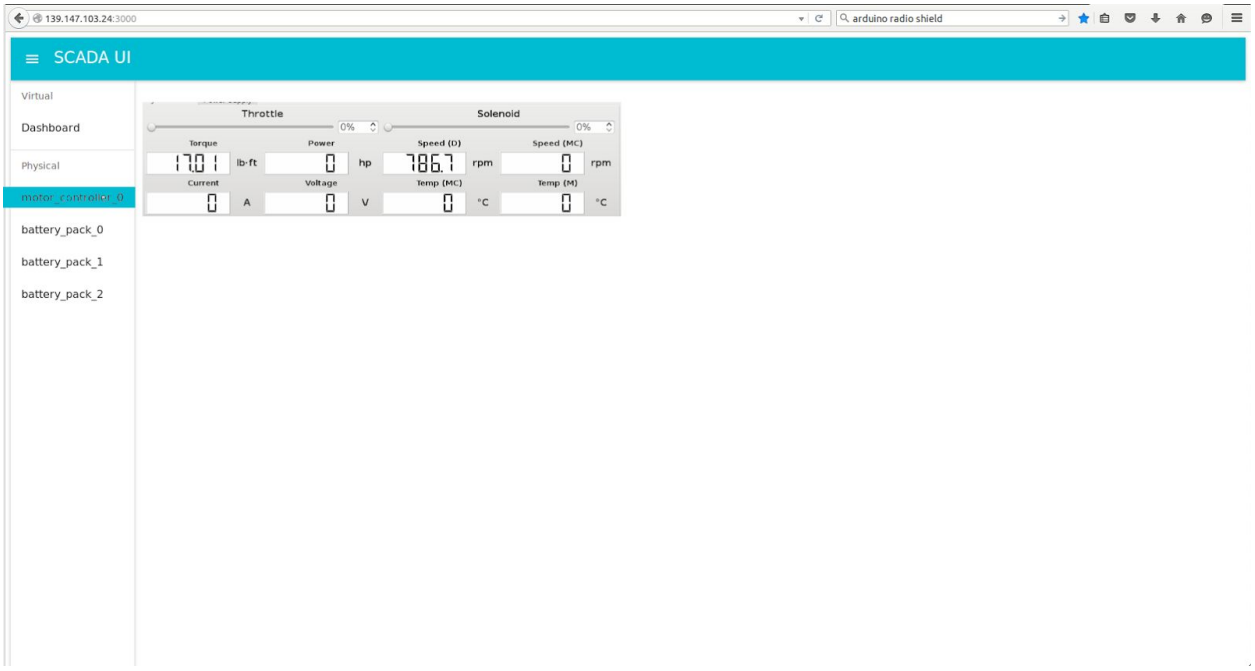
the system in the configuration then the model will also be lacking. If the user describes subsystems that do not exist then the model will contain them, but they will identify as “offline” until contact is made. The configuration file can be modified by connecting to the machine running the daemon over SSH and editing the file with a text editor (nano, vim, etc.)

To clarify the consequences of misconfiguration: Incorrect syntax will result in an error page and no functionality. A subset of the real system will result in an online, but incomplete system, with full SCADA support for the subsystems described. A superset of the real system will result in offline components and full SCADA support.

Software will run on the heavily supported Ubuntu 15, be coded in python, and managed by systemd.

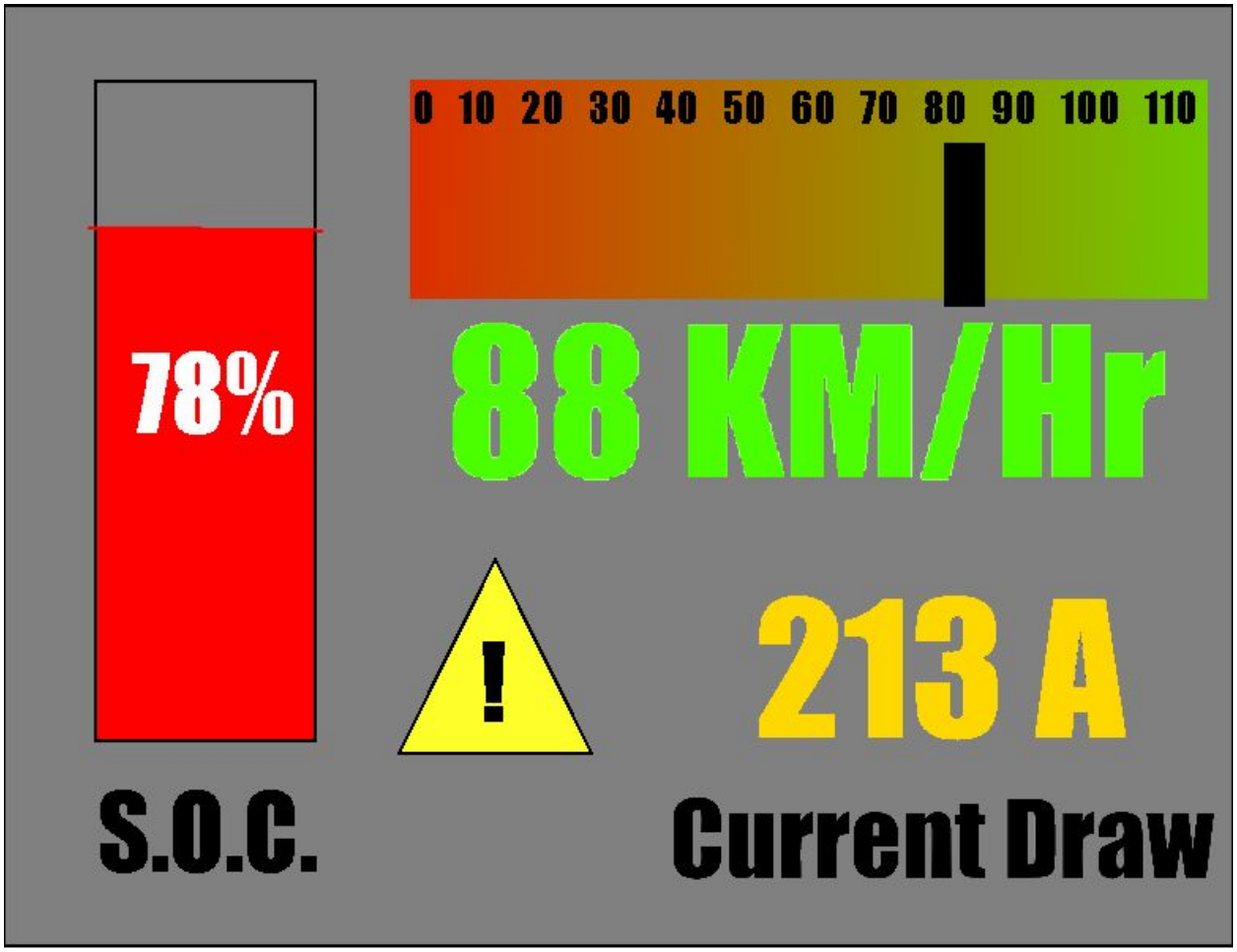
Displays, Indicators and Controls

Displays will be handled by scada-ui program that reads json objects sent out by the scada-d monitoring program. The scada-ui broadcasts a website to display information and also receive commands. The main monitoring system will be a javascript applet similar to the one below. The applet will be available to any browser capable device connected to the same network as the Raspberry Pi. The dashboard will also be a javascript applet that you can navigate to on the 7” dashboard touch screen. The dashboard touchscreen’s computer will be the SCADA Raspberry Pi 3. In drive mode the dashboard should not have any detailed diagnostics according to the formula EV so a warning light comes on when a problem is detected. If you want to know what the warning was check the web page or the SCADA logs. In maintenance mode you may use the touchscreen to navigate the diagnostic website.



Scada display mock-up

7” Raspberry Pi touchscreen acting as dashboard.



Dashboard Mock-up in Drive mode when some error was detected

Motor Characterization and Dynamic Model

Static Characterization and Modeling

Static data has been collected, taking into account inaccuracies measured during torque calibration. Static data plots will determine whether the Motor and controller system can be modeled together as a DC motor. If the static data shows the possibility of modeling this motor as DC then DC parameters will allow the creation of an accurate model for step response testing.

Dynamic Characterization and Modeling

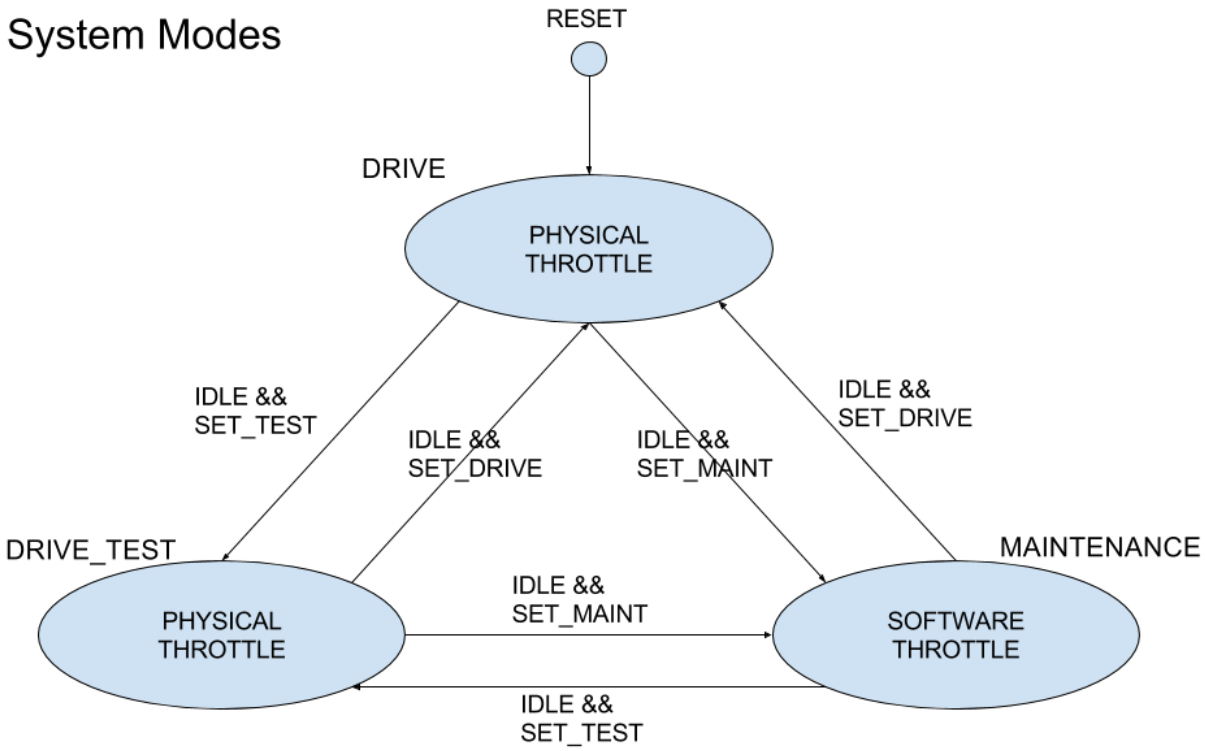
Dynamic data such as the transient response of the motor and motor controller need to be collected. RPM, Temperature and Torque will be measured against time to determine transient parameters of the motor and controller. These parameters will allow the dynamic modeling of the system with polynomial curve fitting.

Efficiency and Cooling

Efficiency calculations are made by comparing torque and rpm data against the power supplies power output(Voltage and Current). The ratio of these two calculations yields motor efficiency data.

Cooling data is being collected both by the motor controller as well as VSCADA-Dyno software. This data, over a range of throttles and loads can help determine the current cooling systems performance.

Analysis of System States



DRIVE_TEST and MAINTENANCE states do not require a closed safety loop for operation.
Transitions to MAINTENANCE require a password to be entered.
A working, closed safety loop is required to provide power to the motor controller in DRIVE.
These states are set and stored by the Raspberry Pi 3.
IDLE is TRUE when no TSV current is flowing in TSI or PacMan, and motor RPM is 0.

SCADA utilizes 3 permanent modes. The main purpose of the three modes is that each mode gives certain restrictions or controls that are not present in other modes. SCADA will attempt to boot into drive mode at the start per EV-rules. Drive mode is characterized by a physical control of the throttle, strict adherence to safety parameters, the inability to operate without the detection of a complete system, and not being able to navigate away from the dashboard on the touch screen. Drive Test mode is similar to drive mode in that it has a physical throttle but does not include a safety check of insuring the entire system is present. Maintenance mode allows the throttle to be set over SCADA and also allows for disabling certain safety features.

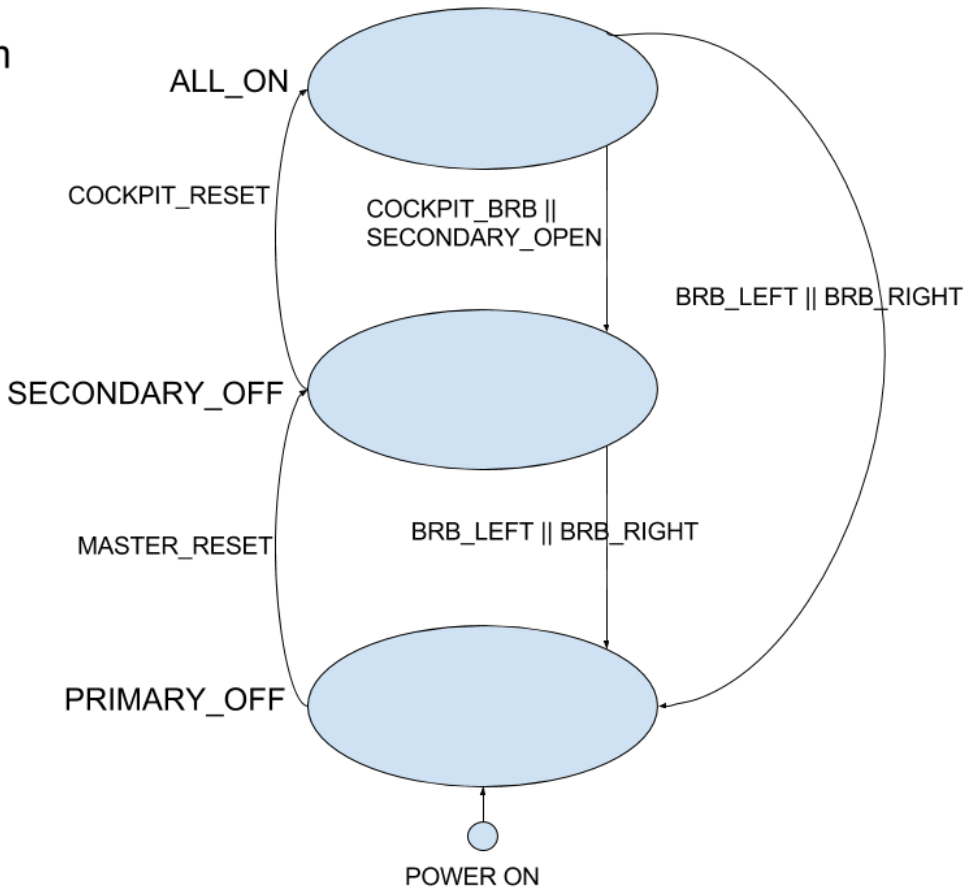
In order to change from the one mode to another two requirements must be met. First, the car needs to be idling aka the throttle is set to zero currently. This prevents the motor from spinning unexpectedly. The second requirement is that the system is told to change modes via SCADA-ui interface. In order to issue a command a password is required. This password ensures that you cannot accidentally change modes as doing so may be catastrophic if attempting to drive the vehicle.

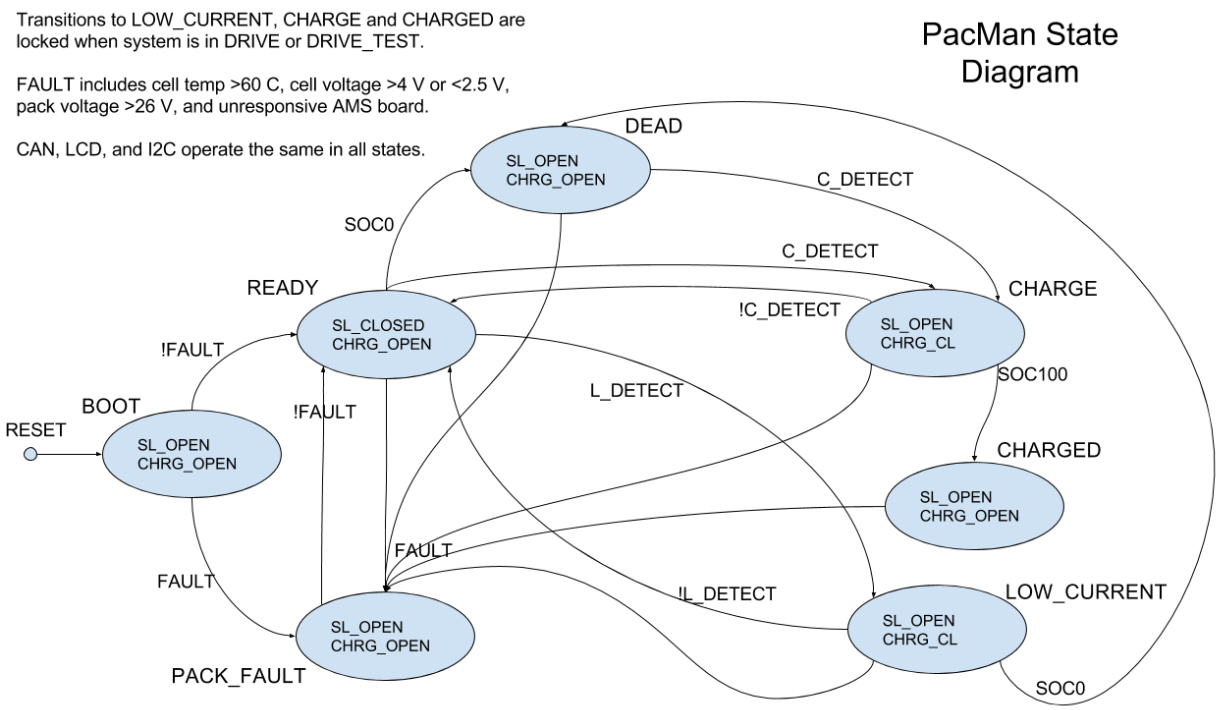
Safety Loop State Diagram

In ALL_ON, the START button requests SCADA allow physical throttle.

Reset buttonw will only cause transition if corresponding BRBs are closed.

These states are stored by the physical position of BRBs and Reset buttons.





On reset, the PacMan computer checks for faults. Faults are temperature over limit, voltage out of range, and an unresponsive AMS board. If any of these are present, the pack fault state is transitioned to, and the safety loop relay is left open. If none are present, the pack transitions to the READY state. In this state the safety loop relay is closed and the accumulator is able to provide power through its high current outputs if a closed safety loop is connected the safety loop in port of the accumulator.

From the READY state, a transition to the CHARGE state occurs if a charger is detected. If the accumulator is fully charged while in the CHARGE state, a transition the CHARGED occurs. The safety loop relay is opened in these two states and the charging relay is closed in CHARGE.

From READY, a transition to LOW_CURRENT occurs if a low current output is detected. Similar to the charging state, the safety loop relay is opened and the charging relay is closed.

All states have transitions to the FAULT state if any of the possible faults are detected. The computer transitions to READY when the fault is no longer present.

Transitions to LOW_CURRENT, CHARGE, and CHARGED are lock out when the system is in DRIVE or DRIVE_TEST modes.

Analysis of Communication Links

CANbus Communication

Overview

CANbus will be the main method of communication across the system. CANbus is a vehicle bus standard designed to allow microcontrollers and devices to communicate with each other in applications without a host computer. It is a message-based protocol, designed originally for multiplex electrical wiring within automobiles. One advantage of CANbus is that it is very easy to add a device to the CANnetwork. It only requires connecting two wires with no additional configuration.

Devices Communicating Over CANbus

- All JGBs send and receive data over CAN
    - JGBs are used in TSV and can be installed to utilize new sensors.
- The motor controller sends measurements and diagnostics over CAN
- PACMAN can send sends measurements and can receive commands over CAN
- SCADA receives and interprets measurements sent over CAN and can also send commands
    - SCADA connects to the CAN network via USB2CAN connector

CANbus Demonstration

CANbus will go through a live demonstration at CDR the results of which will be recorded here.

USB Communication

Overview

When CANBus is not available, usb needs to be used. This is only the case for the Huffbox, which will not be in the final car so USB doesn't need to be a consideration in the final build.

Devices Communicating Over Direct USB

- Huffbox sends data about oil and coolant and receives control information for the physical load on the dyno
- SCADA communicates with the dyno, reading its output and sending the Huffbox commands to adjust the load

Lafayette Network Communication

Overview

This is a local internet connection. It is used for website hosting and displaying. It could be a wireless connection or an ethernet connection. The connected devices can all communicate with each other as long as they are on the same Lafayette Network.

Devices Communicating Over The Lafayette Network

- SCADA's computer will host a webserver that will be able to be accessed remotely to act as the GUI.
- The network will also allow a remote ssh to the SCADA computer for maintenance
- Any cell-phone or PC on the network will link up and be able to utilize the benefits of this network.
- The dashboard will access the GUI across the localhost.

I2C

Overview

Communication within an accumulator is accomplished by I2C. This allows PacMan to monitor all AMS boards, the BBM-01 current sensor, pack voltage and current, and Anderson port connections. PacMan also controls the power to BBM-01 and LCD output via I2C.

Devices Communicating Over I2C

- AT90CAN128 Atmel microcontroller
- DFRobot 4x20 LCD screen
- AMS boards
- NXP Semiconductors PCF8574 I2C Expander
- Texas Instruments ADS1115 analog to digital converter
- Texas Instruments INA226 current monitor
- Silicon Labs Si860 I2C Isolator

Interface Control Specifications

An Interface Control Document has been developed. This document details the cables, the signals carried on them, and pin assignments for connectors at the top level of the system. As the design is finalized additional details will be added.

QA Test Plan

QA Results Reports are generated following any QA test. The comply with requirements detailed in the SOW. Listed here are the QA tests that are required.

QAR001a - Charge Algorithm

1. Mathematical analysis of battery charging/discharging state of charge. The method includes a cell model and coulomb counting.
2. Testing on accumulator test stand.
    a. charging starts appropriately, normal operation
    b. charging stops appropriately, normal operation
    c. charging stops appropriately, all failure modes
3. Charging a discharged TSV accumulator with LiFePO4 cells

QAR001b - Data Acquisition

1. Calibration Accuracy and Analysis (D011)
2. Test I2C messages
3. Test all CAN messages with Lab Terminal, in test stand, in all states
4. Test all CAN messages with VSCADA board, in test stand, in all states
5. Test all CAN messages with Lab Terminal, in Accumulator with LiFePO4 cells, in all states
6. Test all CAN messages with VSCADA board, in Accumulator with LiFePO4 cells, in all states

QAR001c - Displays and Indicators

1. Test all desired displays in test stand, in all states.
2. Test all desired displays in Accumulator with LiFePO4 cells, in all states.

QAR001d - Pack Controls

1. Test navigation to each desired data, or set value, in all states.
2. Set a range of values via controls, in all states.
3. Reset PacMAN in all states.
4. Reset each AMS in all states.

QAR001e - Low Current Output

1. Apply load to draw 20 A.
2. Test Charging functionality.

QAR001f - Delivery of one complete accumulator

1. Annotated photographs of wiring harness.
2. Documentation (Maintenance, User's Manual, BOM, etc.)

3. Demonstration of System States, and availability of TSV power

QAR002b - Safety Loop Integration

1. Ensure that when safety loop is triggered and only when its triggered the UI displays it as such

QAR002m - Measurand Collection

1. Collect data from measurands and compare the results to external instruments to verify proper calibration.
2. When possible reference R005 and R006 requirements to avoid redundancy

QAR002o - Event Logging

1. Show logs of events such as safety loop trigger with expected times of recording

QAR003b - GLV Safety

1. Test for safety loop operation under system faults

QAR003c - Vehicle User Interface Panels

1. Test that buttons and interfaces operate as expected

QAR003d - Tractive System Interface

1. Test that the TSI interacts properly with the safety loop and trips it as needed
2. Monitor that TSV remains isolated from the GLV and ground
3. Test that the Motor/Motor Controller can be engaged and disengaged from driver input

QAR005a - Static Characteristics

1. All specified data measured across full range of operation for torque and RPM
2. Data calibration/accuracy falls within specified tolerances

QAR005b - Dynamic Characteristics

1. All desired model parameters estimated
2. Accuracy analysis determines that parameters are calibrated correctly within proper tolerances

QAR005c - Efficiency and Cooling

1. Motor + controller efficiency and cooling requirements have been successfully measured
2. Tests comparing expected cooling system behavior to measured values are successfully completed

QAR006a - Physics Model

1. Physics model output provides reasonable prediction of fully integrated system performance

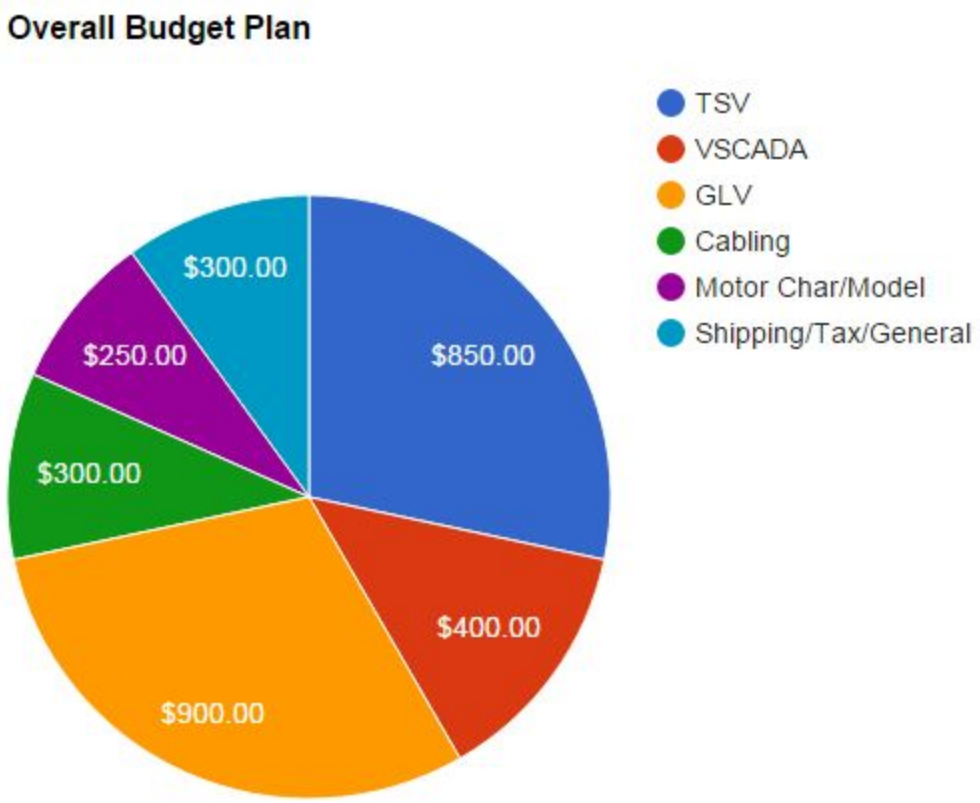
QAR006b - Simulation

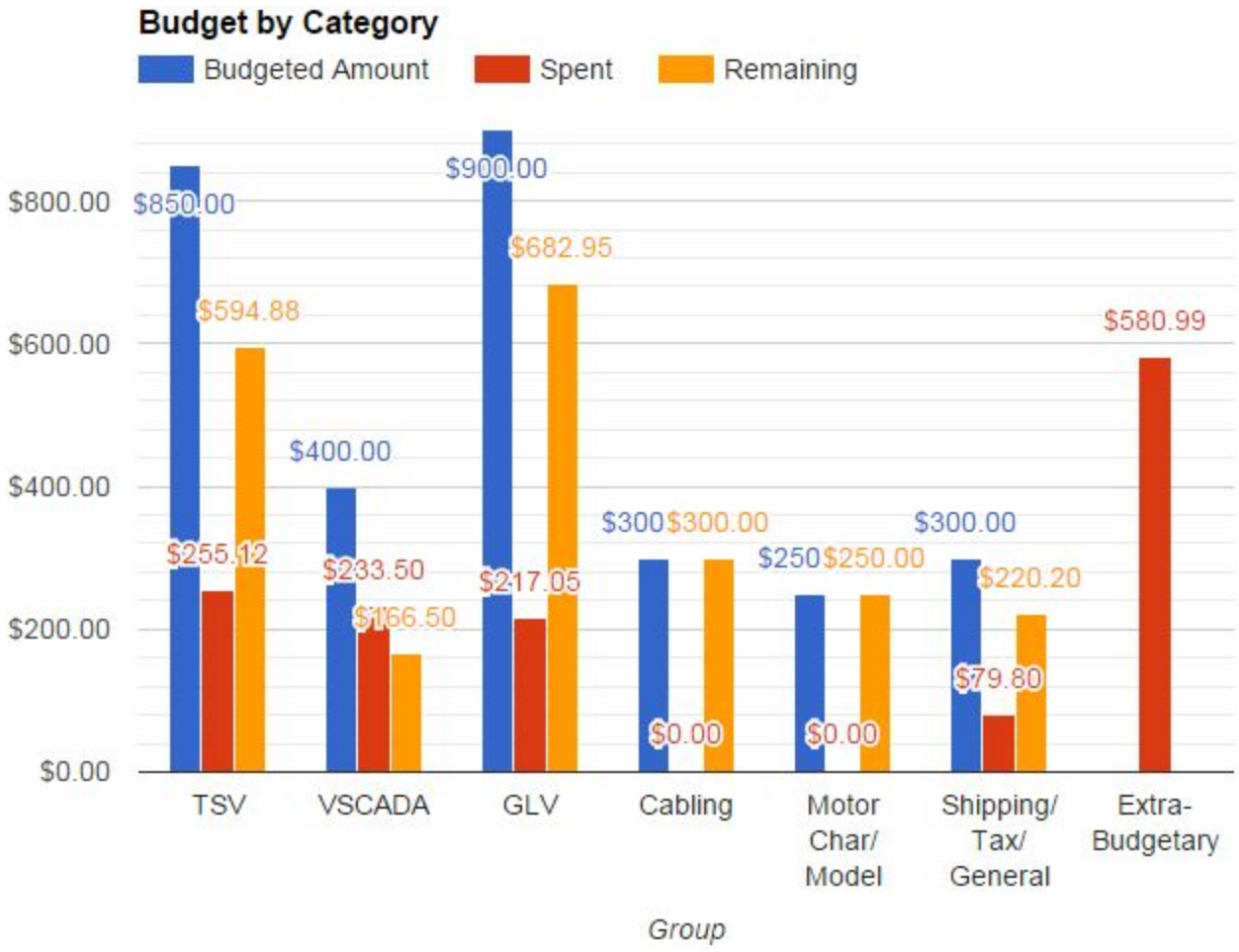
1. Simulation is able to provide outputs expected by the generated physics model
2. Working demonstration to professors successfully completed

QAR006c - Results and Conclusions

1. All data and calculations included in results and conclusions documentation falls within required tolerances, and model provided generates expected outputs for fully integrated system

Cost Analysis



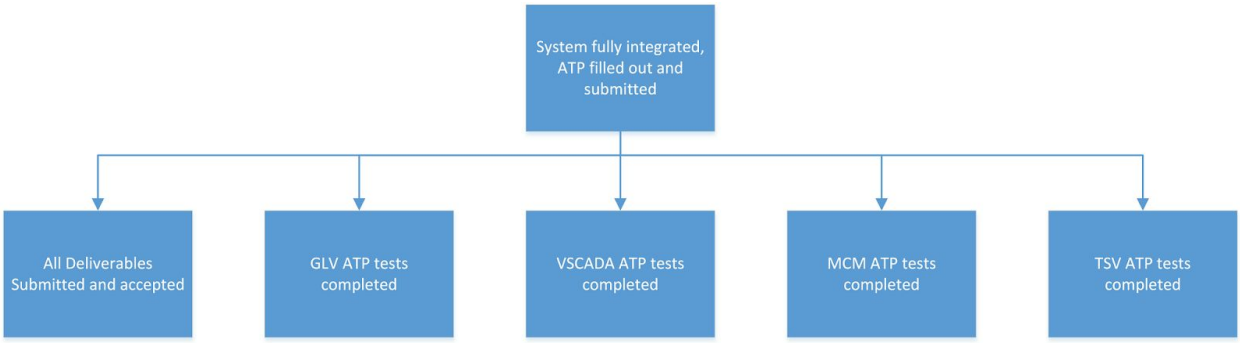

Budget by Category
Budgeted Amount
Spent
Remaining
$800.00
$600.00
$400.00
$200.00
$0.00
$850.00
$255.12
$594.88
$400.00
$233.50
$166.50
$900.00
$217.05
$682.95
$300
$0.00
$300.00
$250
$0.00
$250.00
$300.00
$79.80
$220.20
$580.99
TSV
VSCADA
GLV
Cabling
Motor Char/ Model
Shipping/ Tax/ General
Extra-Budgetary
Group

Thus far into the project, we have spent $743.97 of our $3000 budget, as well as $580.99 outside of the confines of our budget (for the additional accumulator packs beyond the scope of the statement of work).

Schedule

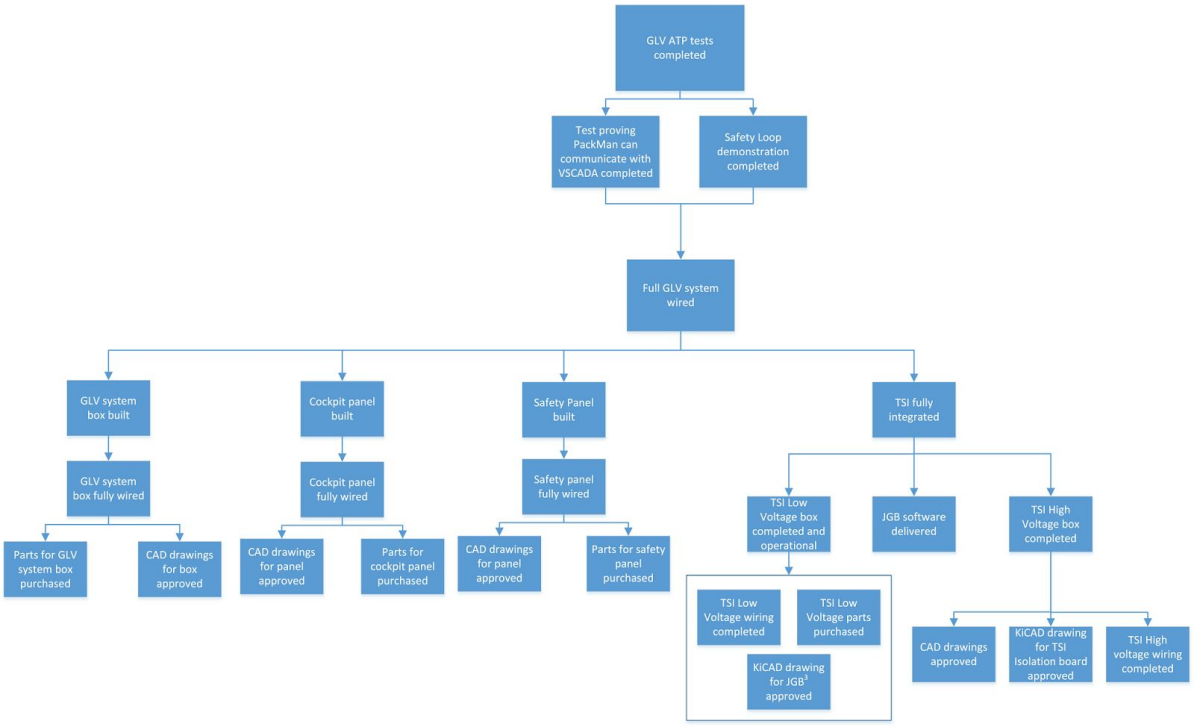
Below is the full work breakdown structure for this project:

Below is the high level work breakdown structure for this project:

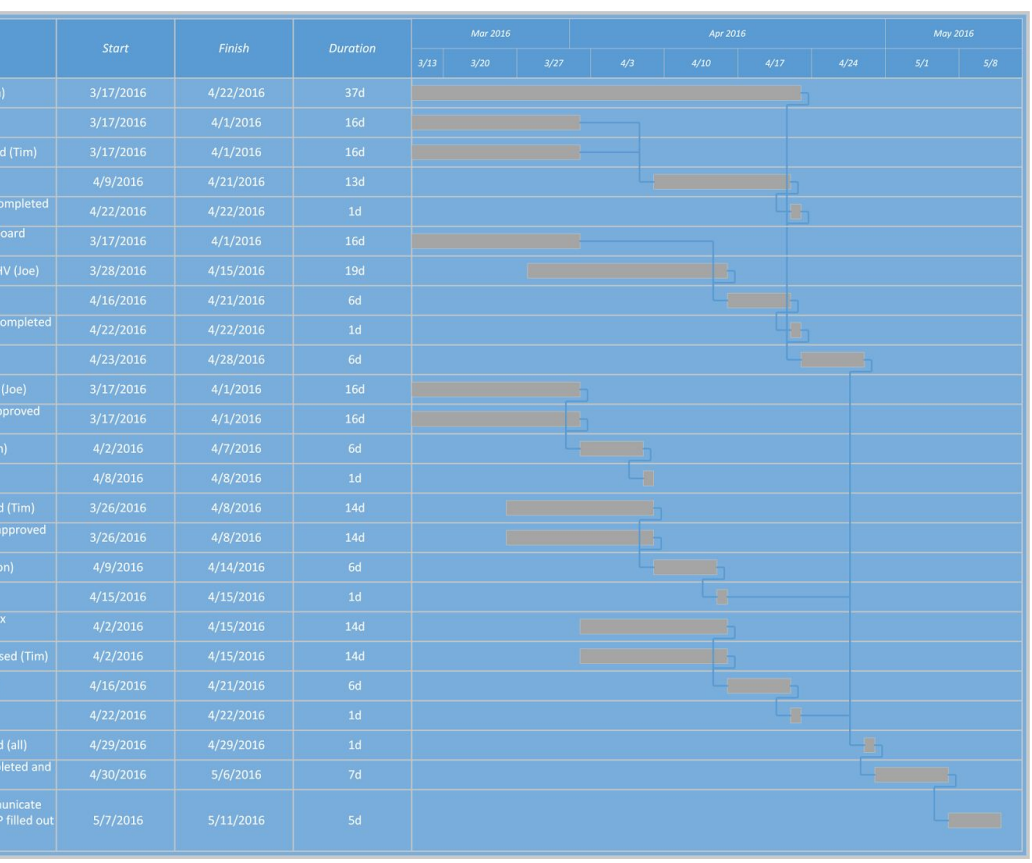


The deadlines for each deliverable will be met as listed in the statement of work.

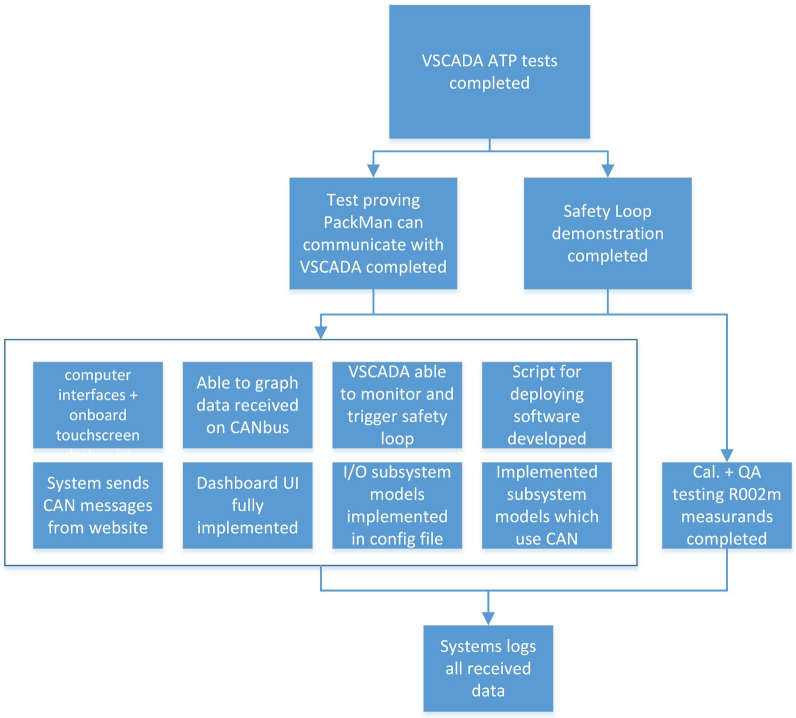
The work breakdown structure and corresponding Gantt chart for GLV are shown below:



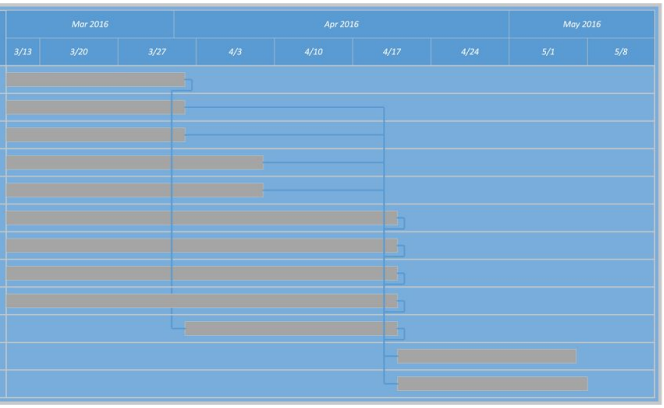
ID	Task Name	Start	Finish	Duration
1	JGB software delivered (Brandon)	3/17/2016	4/22/2016	37d
2	TSI-LV parts purchased (Joe)	3/17/2016	4/1/2016	16d
3	KiCAD drawing for JGB3 approved (Tim)	3/17/2016	4/1/2016	16d
4	TSI-LV wiring completed (Tim)	4/9/2016	4/21/2016	13d
5	TSI-LV box fully integrated and completed (Tim)	4/22/2016	4/22/2016	1d
6	KiCAD drawing for TSI Isolation board approved (Tim)	3/17/2016	4/1/2016	16d
7	CAD drawings approved for TSI-HV (Joe)	3/28/2016	4/15/2016	19d
8	TSI-HV wiring completed (Joe)	4/16/2016	4/21/2016	6d
9	TSI-HV box fully integrated and completed (Joe)	4/22/2016	4/22/2016	1d
10	TSI fully integrated (Tim)	4/23/2016	4/28/2016	6d
11	Parts for safety panel purchased (Joe)	3/17/2016	4/1/2016	16d
12	CAD drawings for safety panel approved (Tim)	3/17/2016	4/1/2016	16d
13	Safety panel fully wired (Brandon)	4/2/2016	4/7/2016	6d
14	Safety panel built (Brandon)	4/8/2016	4/8/2016	1d
15	Parts for cockpit panel purchased (Tim)	3/26/2016	4/8/2016	14d
16	CAD drawings for cockpit panel approved (Tim)	3/26/2016	4/8/2016	14d
17	Cockpit panel fully wired (Brandon)	4/9/2016	4/14/2016	6d
18	Cockpit panel built (Brandon)	4/15/2016	4/15/2016	1d
19	CAD drawings for GLV system box approved (Tim)	4/2/2016	4/15/2016	14d
20	Parts for GLV system box purchased (Tim)	4/2/2016	4/15/2016	14d
21	GLV system box fully wired (Joe)	4/16/2016	4/21/2016	6d
22	GLV system box built (Joe)	4/22/2016	4/22/2016	1d
23	Full GLV system wired/integrated (all)	4/29/2016	4/29/2016	1d
24	Safety loop demonstration completed and ATP filled out (Brandon)	4/30/2016	5/6/2016	7d
25	Test proving PackMan can communicate with VSCADA completed and ATP filled out (Brendon & Geoff)	5/7/2016	5/11/2016	5d



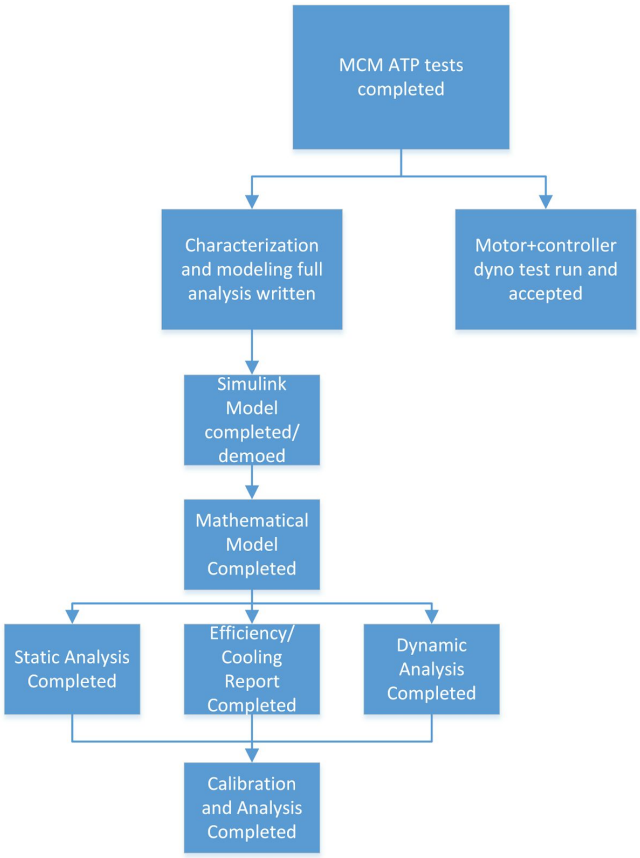
The work breakdown structure and corresponding Gantt chart for VSCADA are shown below:



ID	Task Name	Start	Finish	Duration
1	System logs all received data (Nick)	3/17/2016	4/1/2016	16d
2	System able to send CAN messages from website (Brendon)	3/17/2016	4/1/2016	16d
3	Onboard computer interfaces + onboard touch-screen dashboard exist (Nick)	3/17/2016	4/1/2016	16d
4	Data received from CANbus can be graphed (Nick)	3/17/2016	4/8/2016	23d
5	Dashboard UI fully implemented (Brendon)	3/17/2016	4/8/2016	23d
6	Implemented subsystem models which use CAN (Brendon)	3/17/2016	4/20/2016	35d
7	Script for deploying software delivered (Nick)	3/17/2016	4/20/2016	35d
8	All I/O subsystem models implemented in a configuration file (Brendon)	3/17/2016	4/20/2016	35d
9	VSCADA able to monitor and trigger safety loop (Nick)	3/17/2016	4/20/2016	35d
10	Calibration and QA testing R002m measurands completed (all)	4/2/2016	4/20/2016	19d
11	Integration with other groups / ATP testing completed (Brendon)	4/21/2016	5/6/2016	16d
12	Tutorial and documentation for future groups completed (Nick)	4/21/2016	5/7/2016	17d

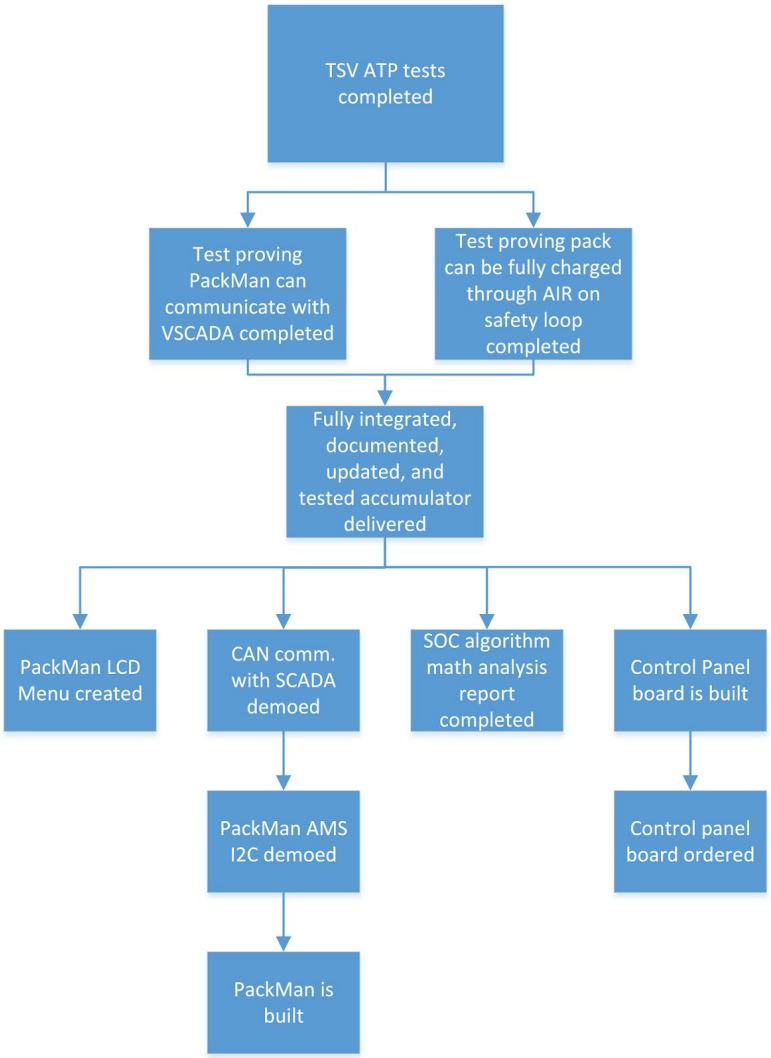


The work breakdown structure and corresponding Gantt chart for MCM are shown below:

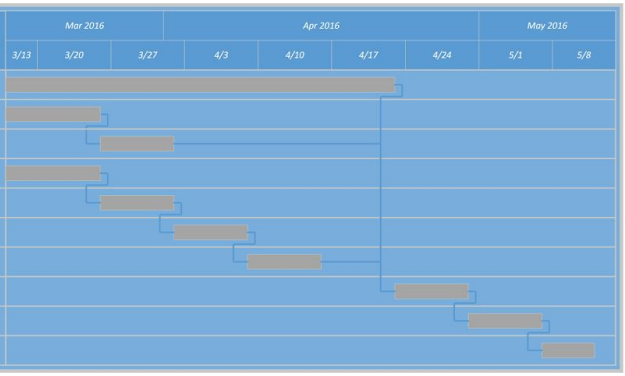


ID	Task Name	Start	Finish	Duration
1	Static Analysis complete, report portion completed (Armen)	3/17/2016	3/25/2016	9d
2	Calibration analysis report completed, calibration factors added (Armen	3/17/2016	3/29/2016	13d
3	Data collected, dynamic characterization (Dan)	3/26/2016	3/29/2016	4d
4	Dynamic analysis completed, report portion completed (Dan)	3/30/2016	4/9/2016	11d
5	Data collected, cooling (Armen)	4/10/2016	4/13/2016	4d
6	Efficiency / cooling analysis completed, report portion completed (Armen)	4/14/2016	4/16/2016	3d
7	Mathematical physics model completed, report portion completed (Dan)	4/17/2016	4/30/2016	14d
8	Motor, controller, dyno test completed, ATP portion filled out (Armen)	5/3/2016	5/3/2016	1d
9	Simulink model completed, able to be demoed (Dan)	5/1/2016	5/7/2016	7d
10	Final report completed, ATP for group fully filled out (Dan)	5/8/2016	5/13/2016	6d

The work breakdown structure and corresponding Gantt chart for TSV are shown below:


TSV ATP tests completed
Test proving PackMan can communicate with VSCADA completed
Test proving pack can be fully charged through AIR on safety loop completed
Fully integrated, documented, updated, and tested accumulator delivered
PackMan LCD Menu created
CAN comm. with SCADA demoed
SOC algorithm math analysis report completed
Control Panel board is built
PackMan AMS I2C demoed
Control panel board ordered
PackMan is built

ID	Task Name	Start	Finish	Duration
1	State of charge alogrithm mathematical analysis report completed (Geoff)	3/17/2016	4/22/2016	37d
2	Control panel board ordered (Jae)	3/17/2016	3/25/2016	9d
3	Control panel board built (Jae)	3/26/2016	4/1/2016	7d
4	PackMan is built (Geoff)	3/17/2016	3/25/2016	9d
5	PackMan AMS I2C demoed (Geoff)	3/26/2016	4/1/2016	7d
6	PackMan LCD Menu created (Geoff)	4/2/2016	4/8/2016	7d
7	CAN communication with SCADA demoed (Geoff)	4/9/2016	4/15/2016	7d
8	Accumulator completely assembled (Geoff)	4/23/2016	4/29/2016	7d
9	All R001 QA tests completed (Geoff)	4/30/2016	5/6/2016	7d
10	Final ATP testing completed (Geoff)	5/7/2016	5/11/2016	5d



Appendix A: Schematics

See Attached

Appendix B: Bill of Materials

See Attached

Appendix C: Fabrication Specifications

See Attached