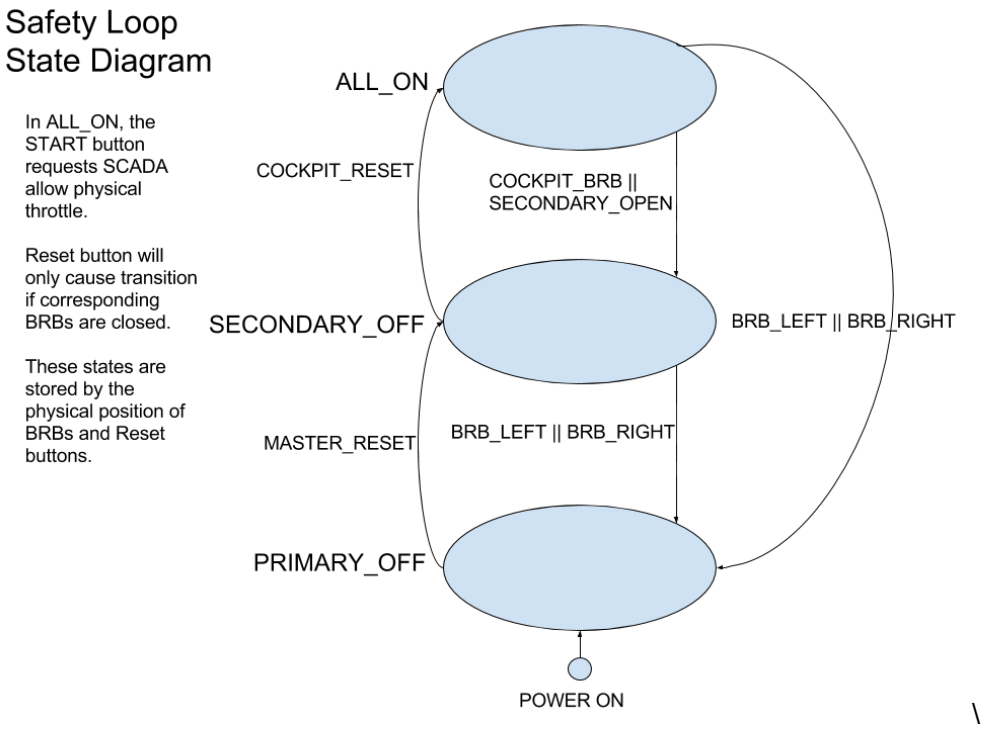
LFEV Spring 2016 Maintenance Manual

Do not be alarmed this is merely a draft

Table of Contents

GLV

Safety Loop



\

After Opening the Safety Loop

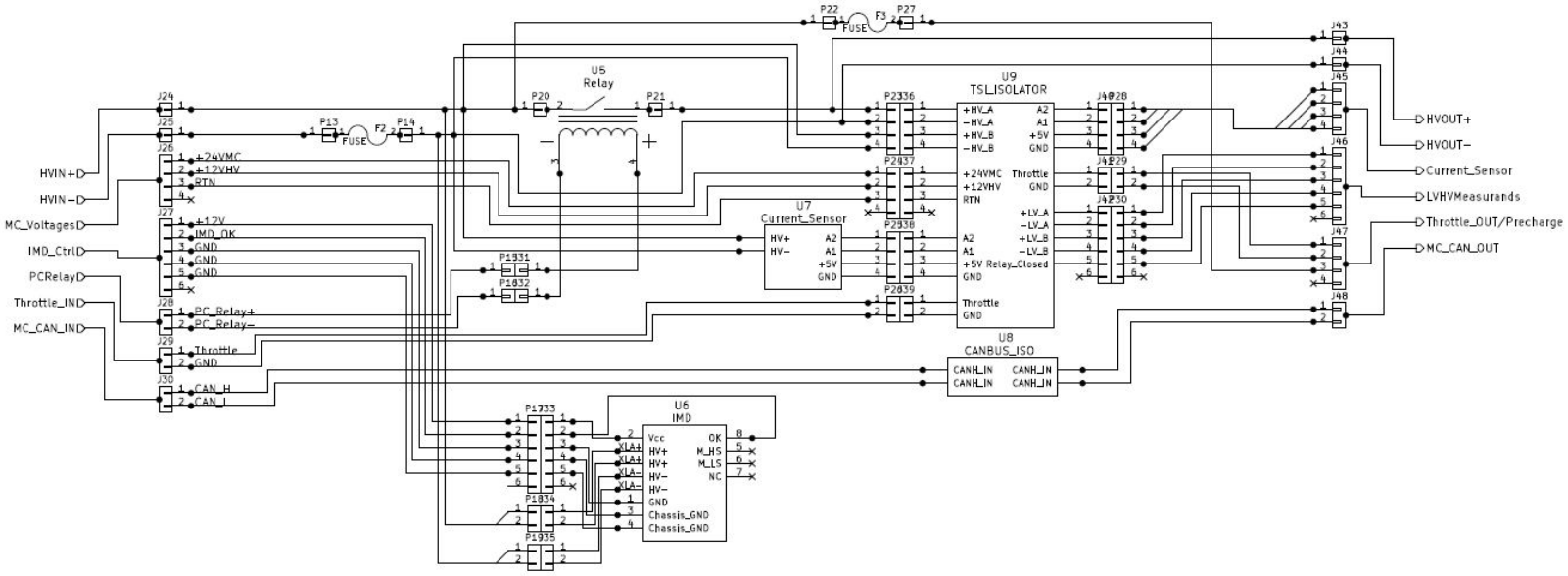
If the Safety Loop is ever opened, the user should find the source of the failure. Sources of failures include:

- IMD Fault
- AMS Fault
- BOTS flipped
- VSCADA error
- Motor Controller
- Safety Loop physically disconnected
- Loss of power from the GLV
- Master Switches

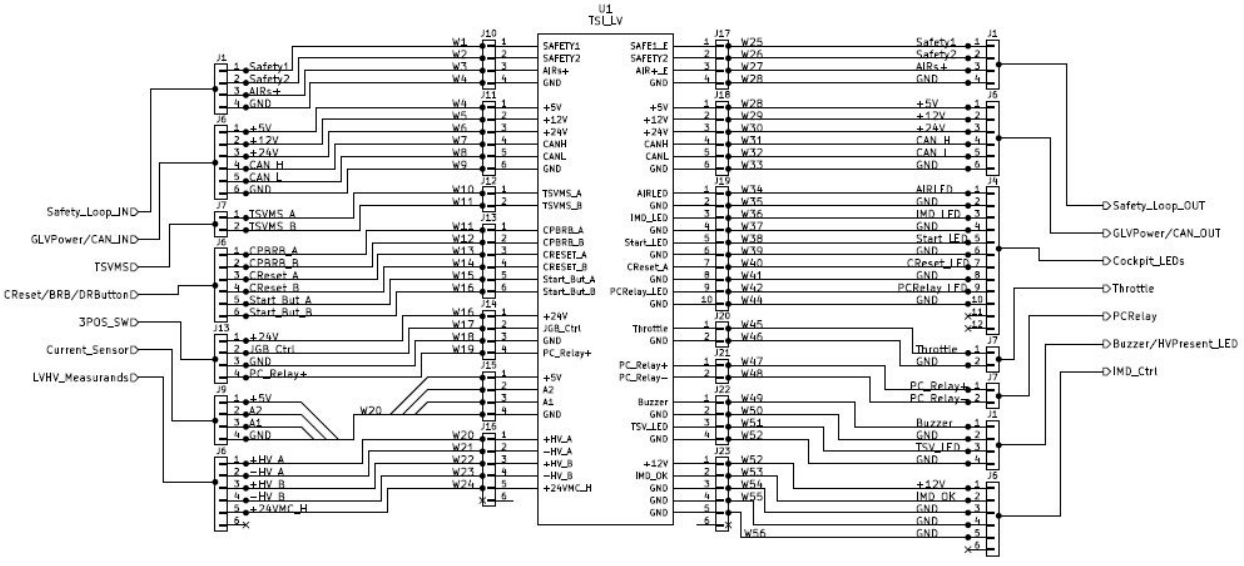
Before resetting the Safety Loop, the user should fully inspect each element to determine the cause of the Safety Loop opening. Do not simply reset the system with the green button without determining the original source of fault.

TSI

High Voltage system schematic of the TSI



TSI low voltage system diagram



VSCADA

Hardware

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.

Software

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.

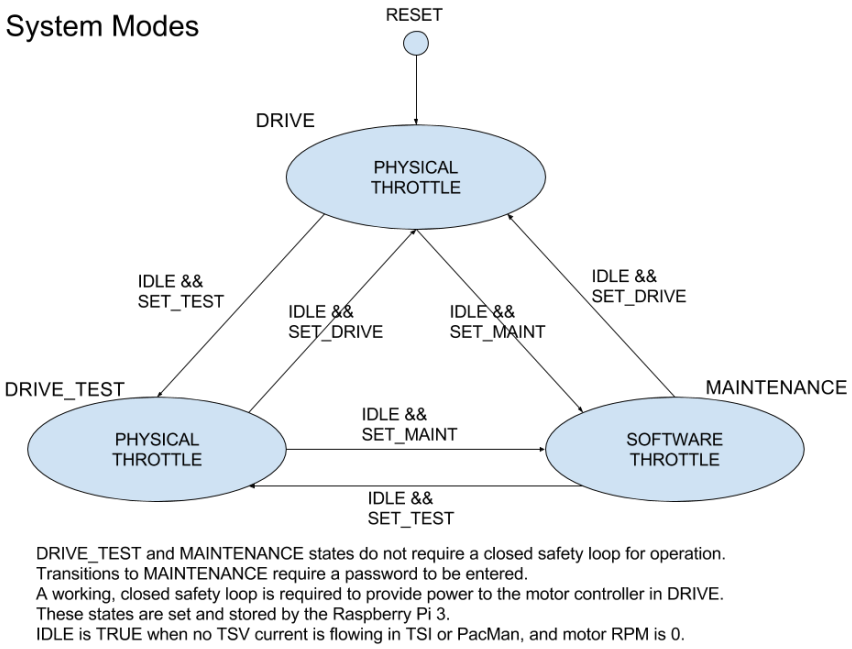
System Model

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. The 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.



Other Information on Maintainability

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 system.

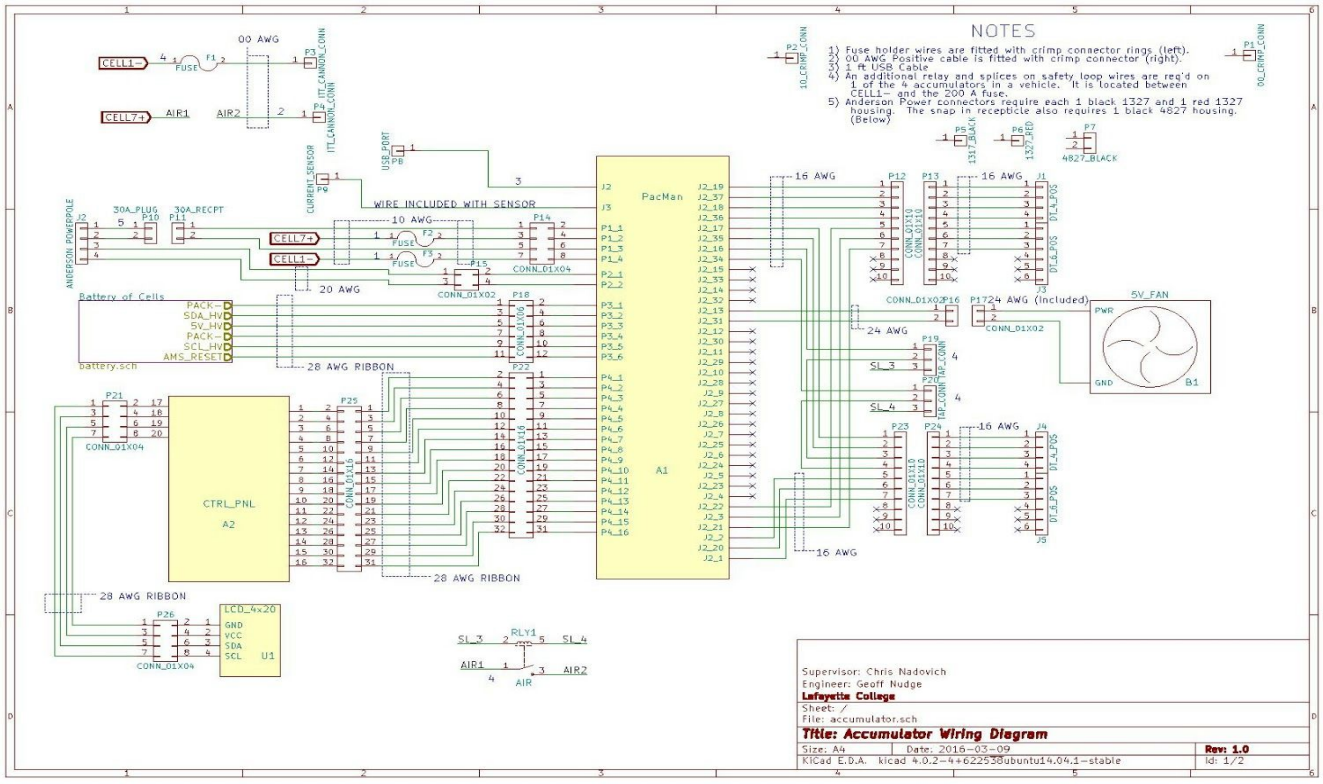
TSV

Data Acquisition

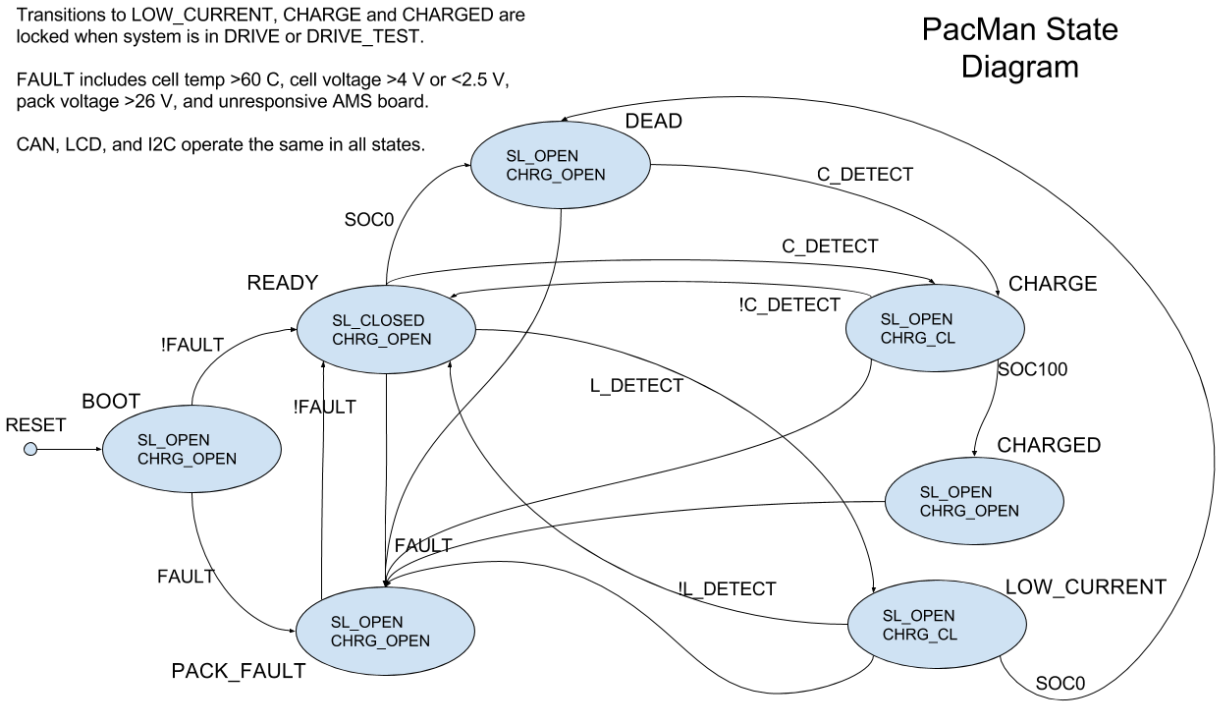
The PacMan computer monitors the overall pack current and voltage directly, and individual cell voltages and temperatures via I2C communication with the AMS boards (see spring 2015 design documents). Calibration factors for these values are stored on PacMan and may be modified with the control panel. 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

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.



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.



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.

Dyno

Calibration

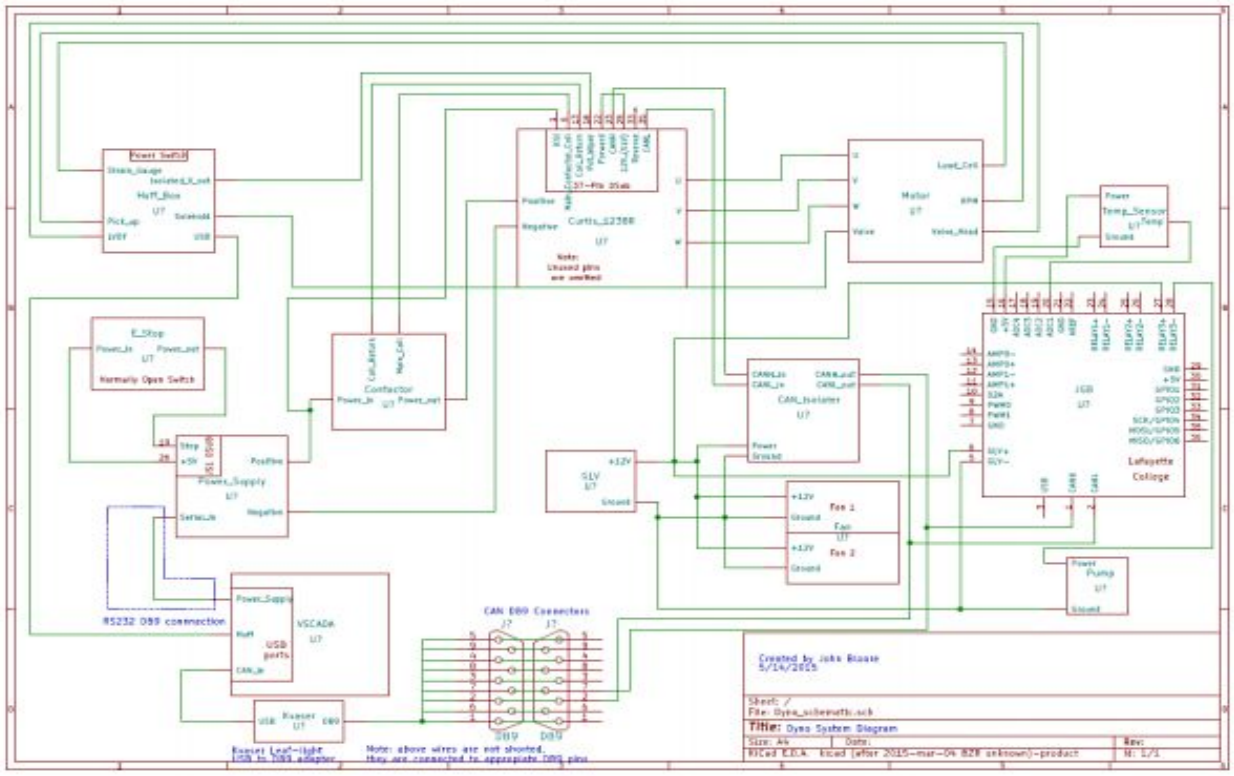
Refer to the torque calibration document. This lists the entire procedure involved with calibrating the torque sensor.

Maintenance

Water Cooling System Maintenance:
In its current state the water cooling system is not maintainable. The water reservoir can not be emptied without removing the entire motor controller stand from the dynamometer. This issue must be addressed next year. A possible solution to consider is to move the main water basin off the dynamometer. This will let the water be changed more easily as well as air would be able to escape the system.

Dynamometer Oil Maintenance:
The oil in the dynamometer does not need to be changed or replaced unless there is a spill. In the case of a spill there is oil absorbent located in AEC 402. After consulting an advisor proceed to apply the adsorbent followed by a clean up of the room when the oil has been absorbed. Once the new oil has arrived to replace the oil you must get the oil pump from the mechanical engineering workshop. They will instruct you on how to use this to refill the dynamometer with oil.

The Dynamometer system schematic is provided below to better understand the interfacing between subsystems in AEC 401. This was all completed previously by the class of 2015 whose schematic is provided below.


Power Switch
Hall_Box
Curtis_1238B
Motor
Contactor
E_Stop
Normally Open Switch
Power_Supply
CAN_Isolater
Fan 1
Fan 2
Pump
Temp_Sensor
Lafayette College
VSCADA
USB ports
RS232 DB9 connection
CAN DB9 Connectors
Note: above wires are not shorted. they are connected to appropriate DB9 pins
Created by John Blouse 5/14/2015
Sheet: /
File: Dyno_schematic.sch
Title: Dyno System Diagram
Size: A4
Date:
Rev:
KiCad E.D.A. kicad (after 2015-mar-04 BZR unknown)-product
Id: 1/1