

The background features a large, faint watermark of the Lafayette University seal. The seal is circular and contains a portrait of a man, likely a founder or significant figure, surrounded by the Latin motto "VERITAS LIBERABIT VOS" and the year "1826".

# Lafayette Electric Vehicle

# 2015

# LAFAYETTE

ECE 492: Senior Design II

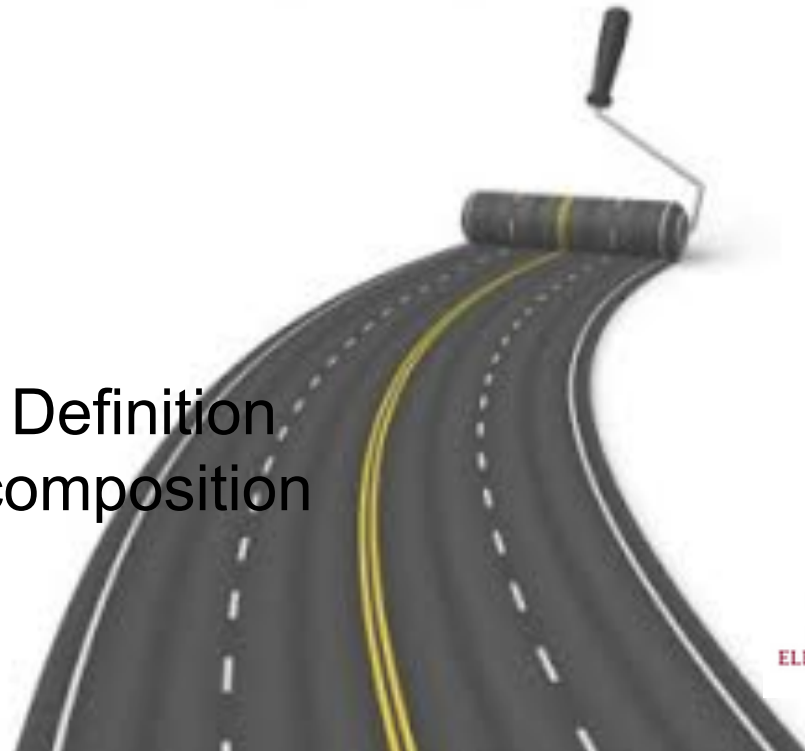
Afternoon Critical Design Review

March 11, 2015

Hugel 100

# Roadmap

10. Meet the Afternoon Teams
11. Interface Control and Assemblies Review
12. Vehicle Supervisory Control and Data Acquisition (VSCADA)
  - a. Daemon
  - b. Interfacing
  - c. User Applications
  - d. Data Storage
13. Dynamometer (DYNO)
  - a. Decomposition and Definition
  - b. Integration and Recomposition



# Roadmap

## 10. Meet the Afternoon Teams

11. Interface Control and Assemblies Review

12. Vehicle Supervisory Control and Data Acquisition (VSCADA)

a. Daemon

b. Interfacing

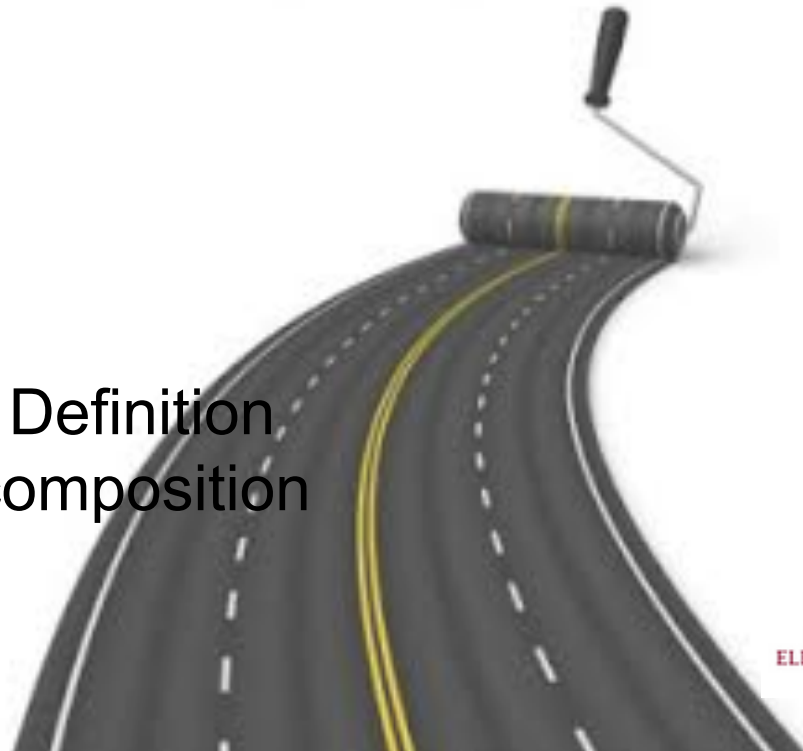
c. User Applications

d. Data Storage

13. Dynamometer (DYNO)

a. Decomposition and Definition

b. Integration and Recomposition



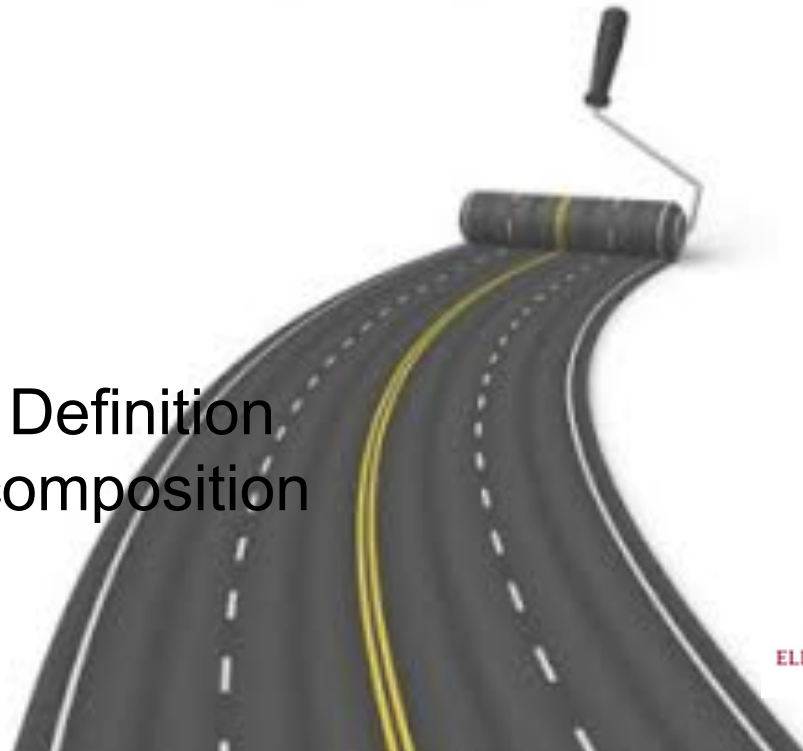
# Meet the Afternoon Teams

- Vehicle Supervisory Control and Data Acquisition (VSCADA)
  1. Yiming Chen
  2. Bikram Shrestha
  3. Rameel Sethi
  4. John Gehrig
  5. Sam Cesario
  6. Adam Cornwell
- Dynamometer (DYNO)
  1. Steve Mazich
  2. Brendan Malone
  3. John Bloore
  4. Nate Hand
  5. Alex Hytha



# Roadmap

10. Meet the Afternoon Teams
- 11. Interface Control and Assemblies Review**
12. Vehicle Supervisory Control and Data Acquisition (VSCADA)
  - a. Daemon
  - b. Interfacing
  - c. User Applications
  - d. Data Storage
13. Dynamometer (DYNO)
  - a. Decomposition and Definition
  - b. Integration and Recomposition



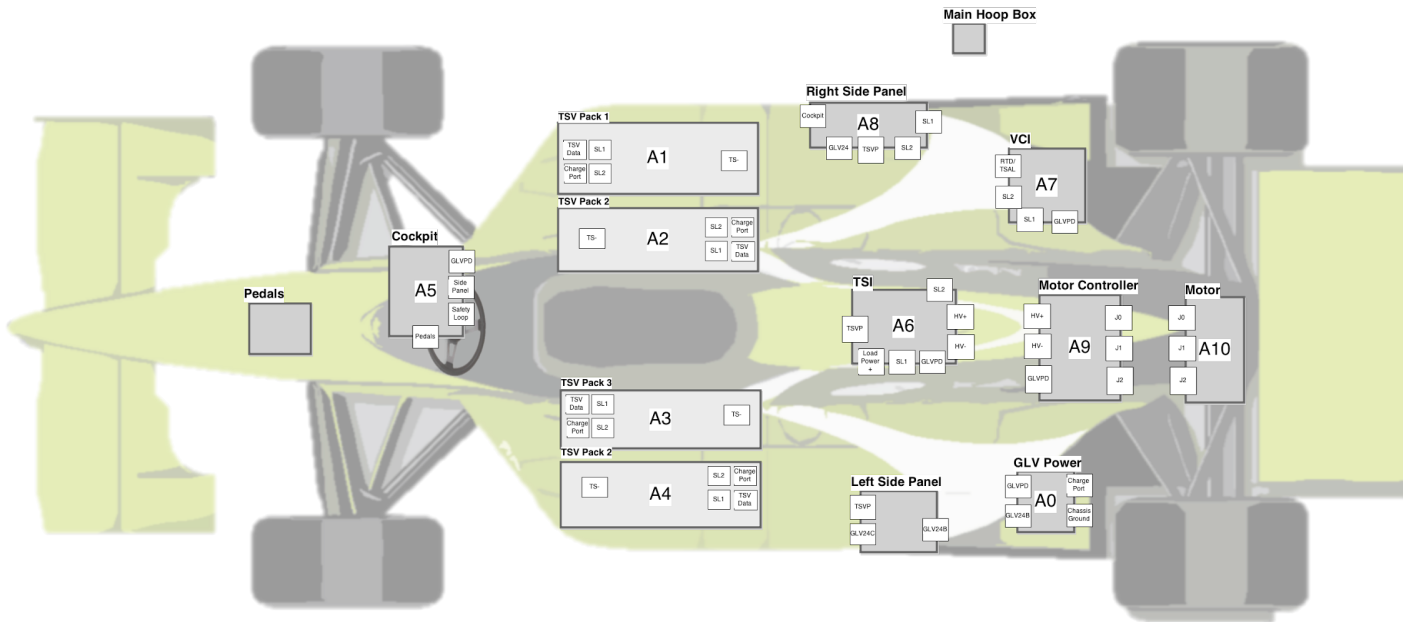
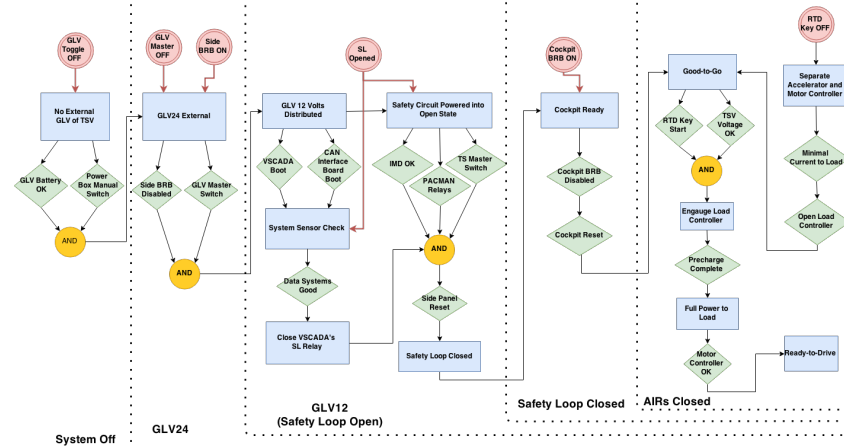
# Interface Control

An Interface Control Document was created to accurately and completely define all (electrical, mechanical, semantic) aspects of top-level interfaces to allow different designers to coordinate with each other successfully.

Next, we will discuss these top-level interfaces.

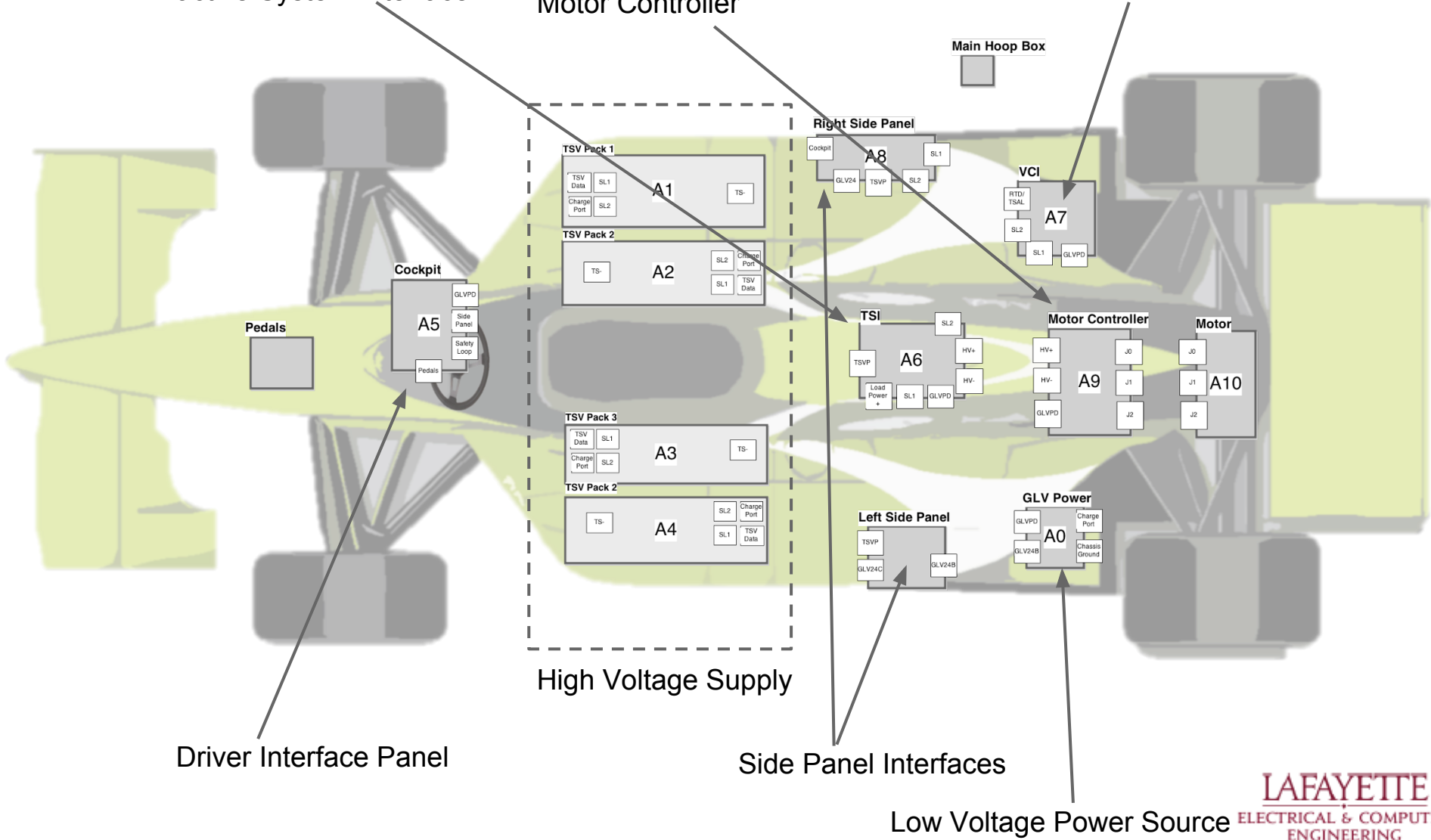
# Introduction: Motivation

- SAE Formula Hybrid Competition Vehicle
  - Electric Car
- Capable of Vehicle Integration
- Four Team Integration



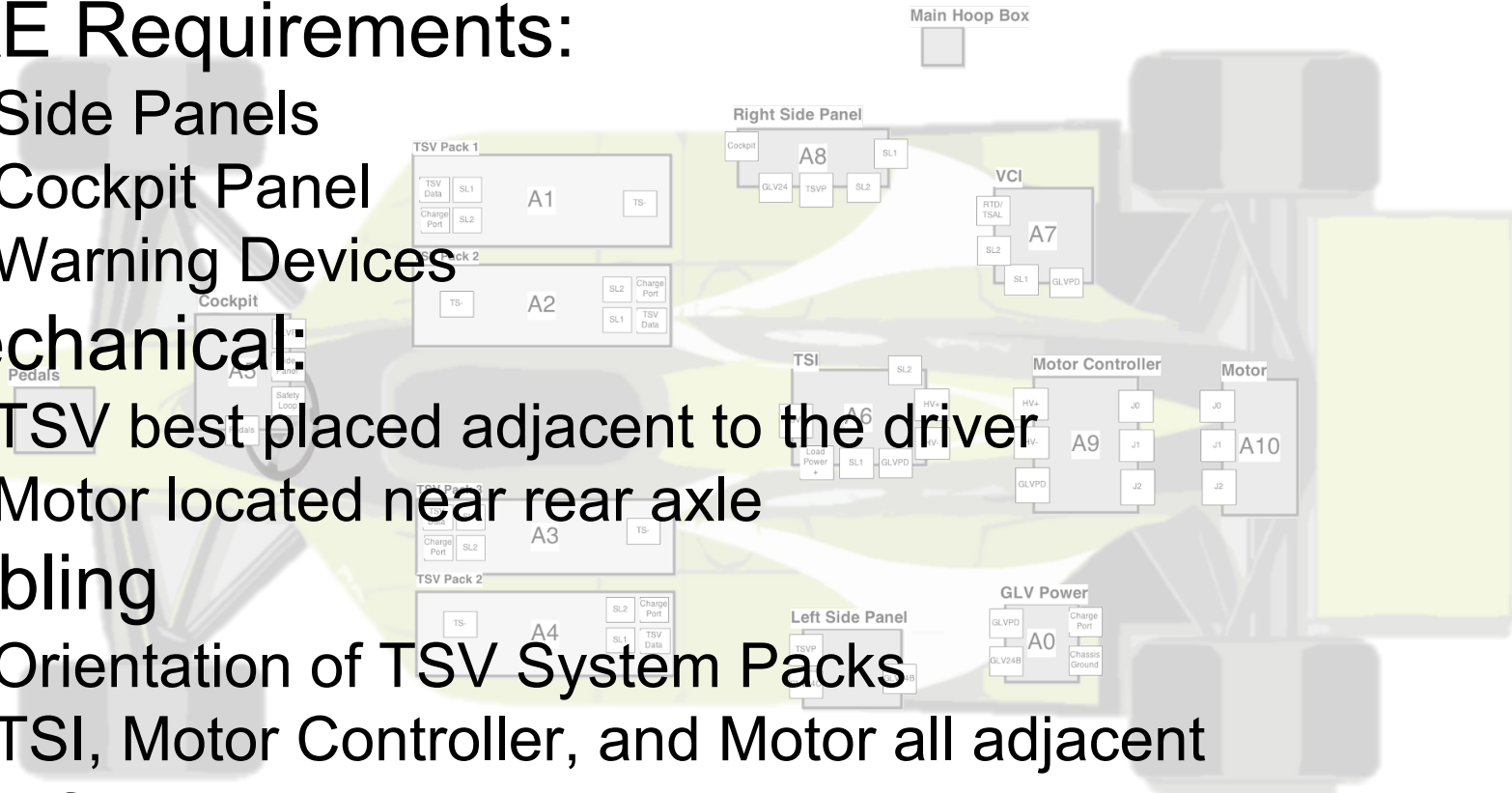
# System Assemblies Layout - Top View

Tractive System Interface      Motor Controller      Vehicle Computer Interface

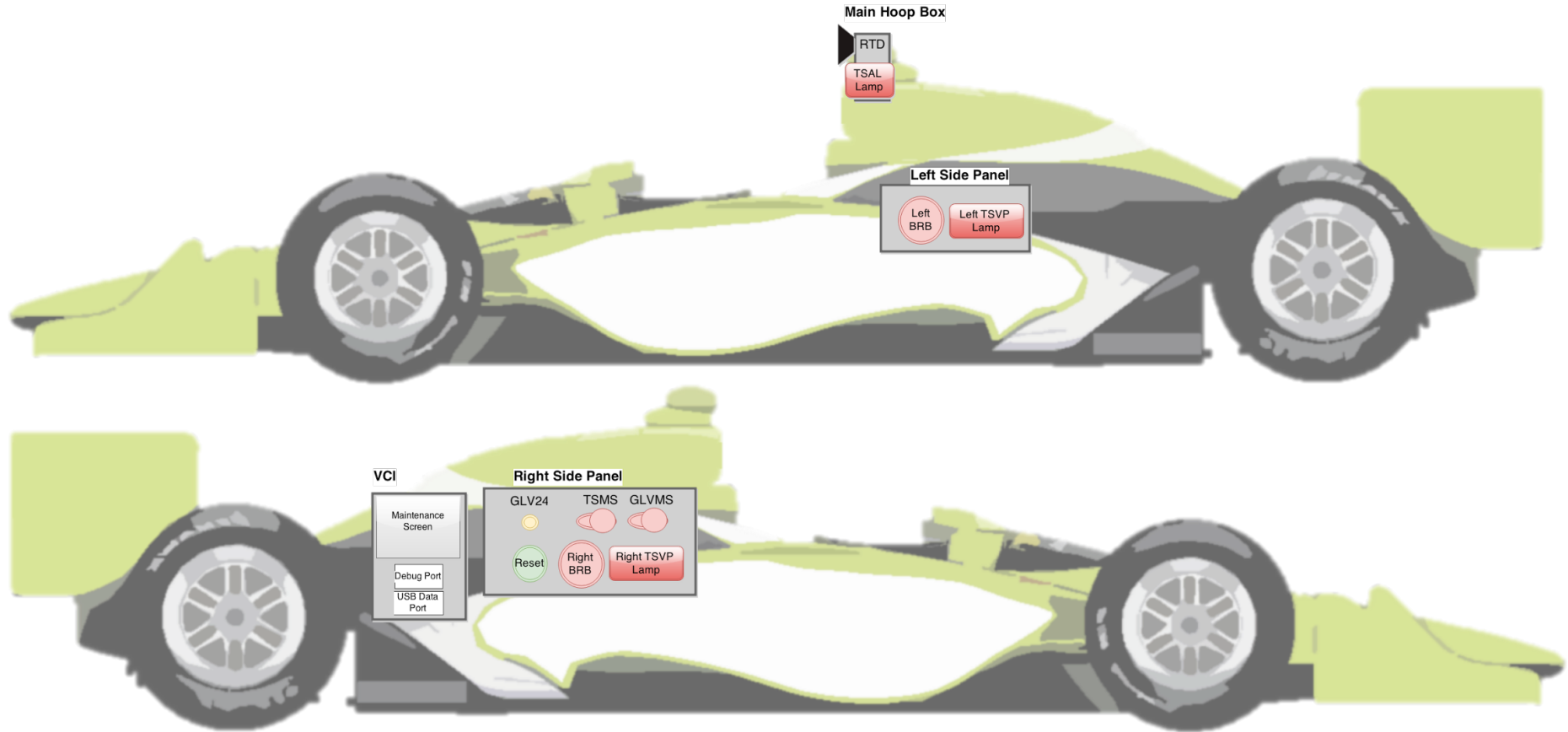


# Layout Selection

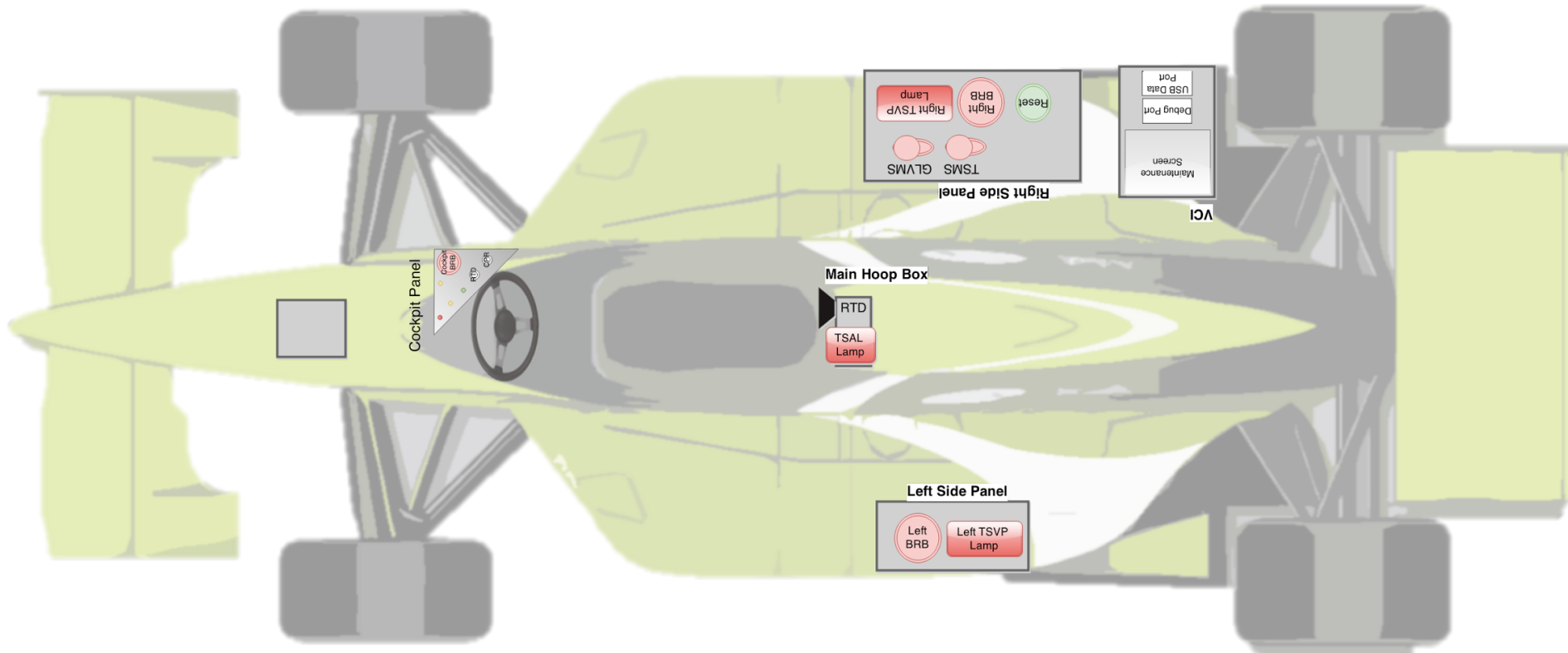
- SAE Requirements:
  - Side Panels
  - Cockpit Panel
  - Warning Devices
- Mechanical:
  - TSV best placed adjacent to the driver
  - Motor located near rear axle
- Cabling
  - Orientation of TSV System Packs
  - TSI, Motor Controller, and Motor all adjacent
- Interfacing
  - VCI accessible by pit station crew



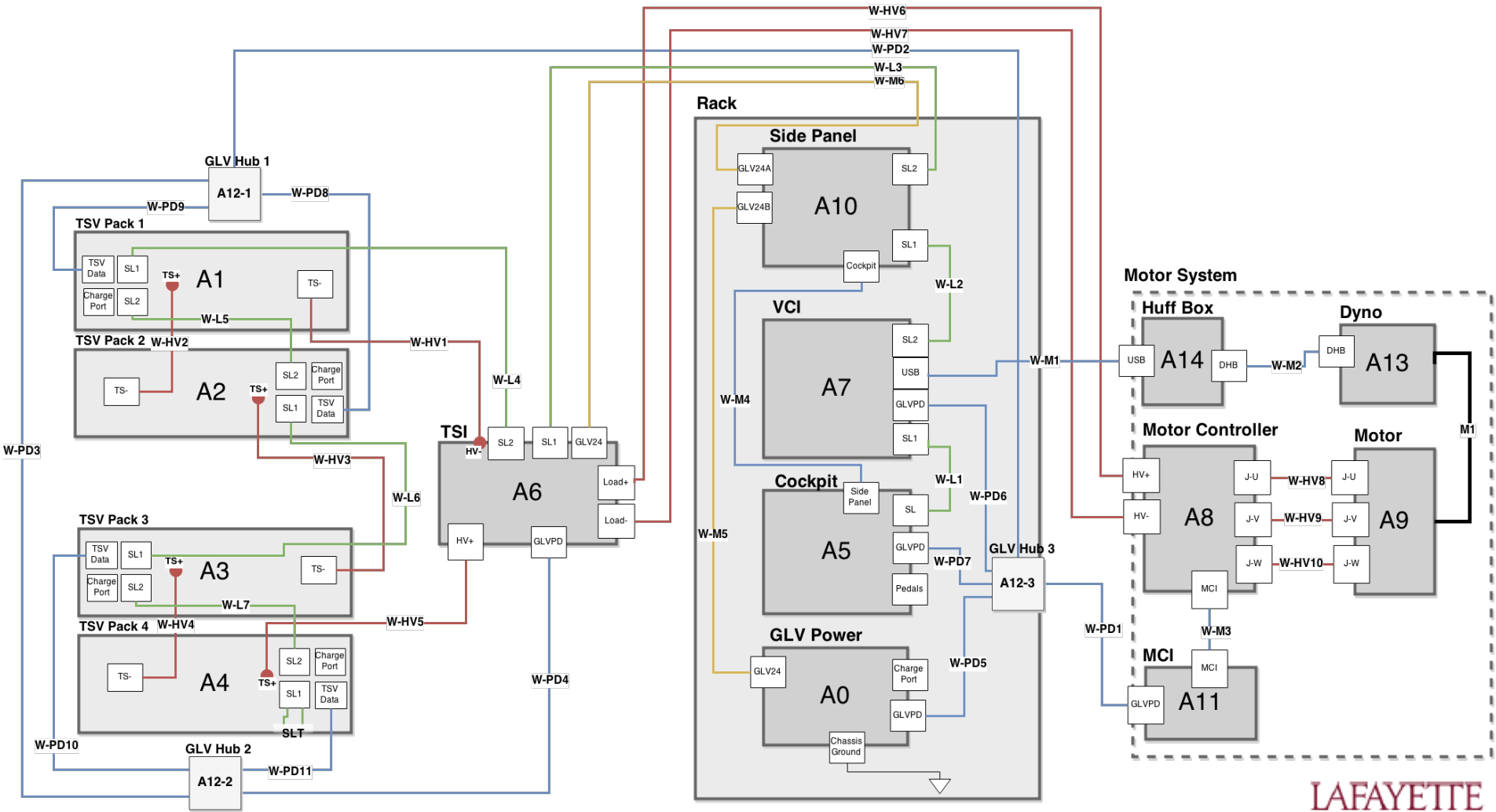
# Physical Interfaces Layout - Side View



# Physical Interfaces Layout - Top View

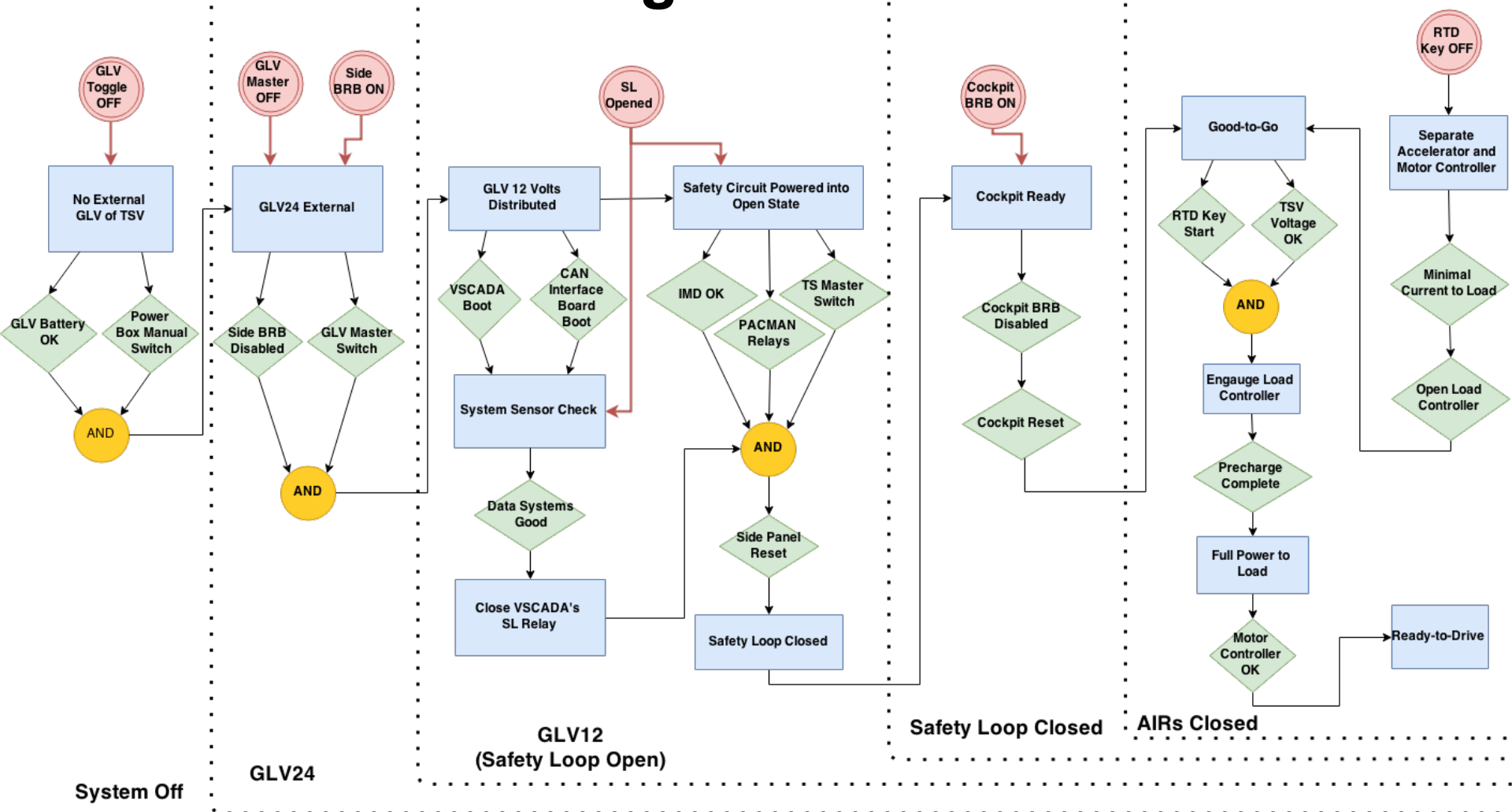


# End-of-Term Integration Layout



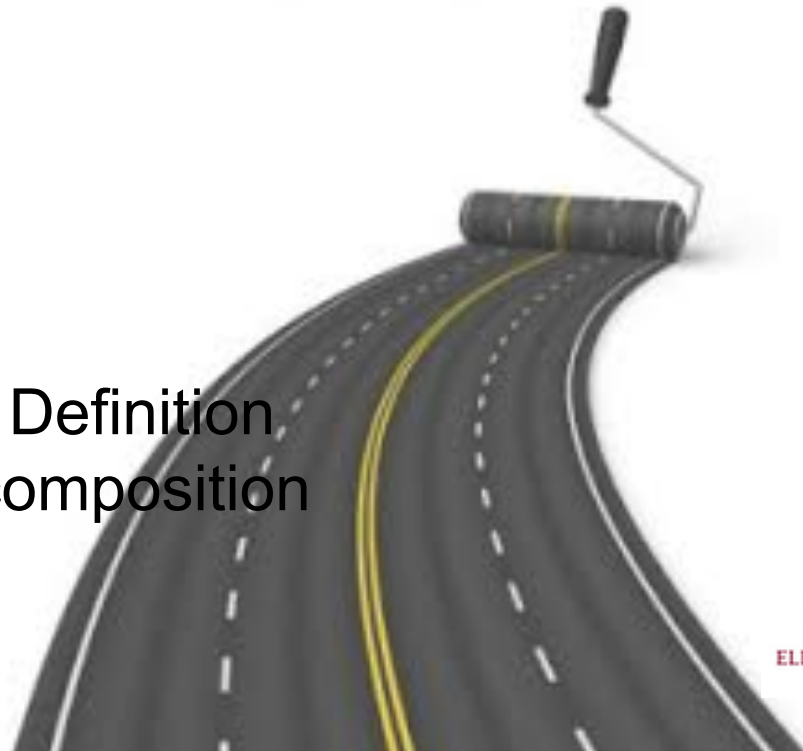


# State Transition Diagram

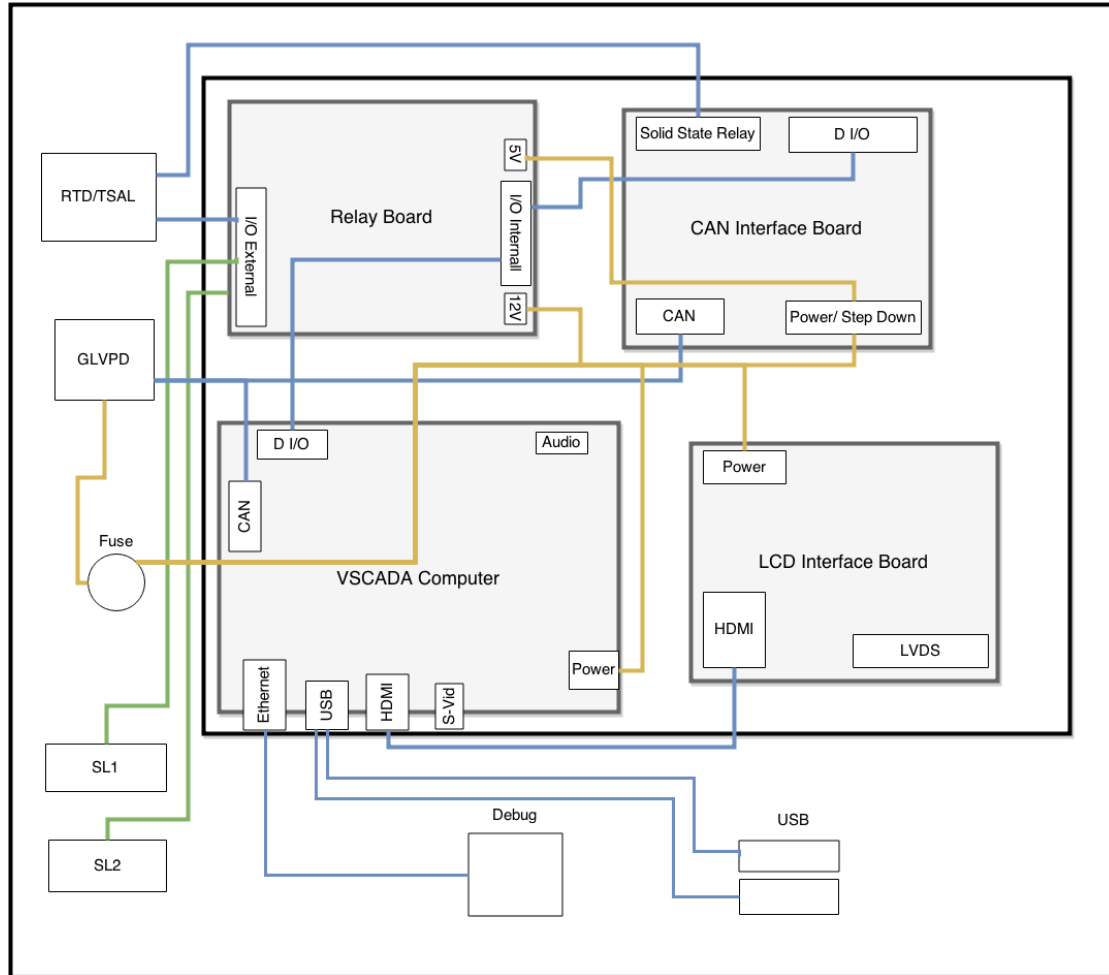


# Roadmap

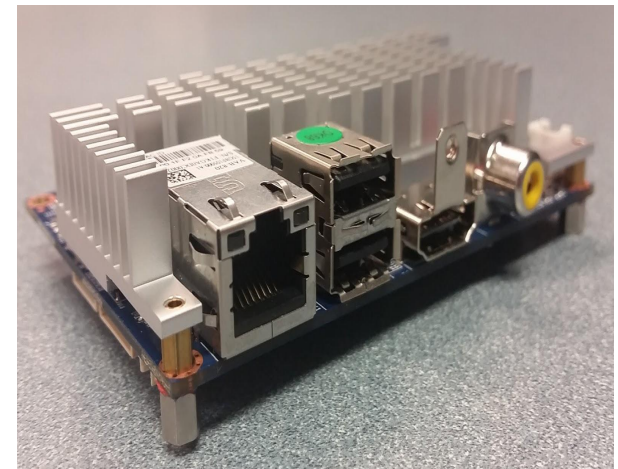
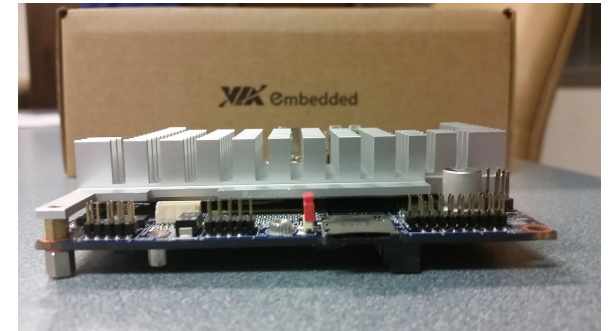
10. Meet the Afternoon Teams
11. Interface Control and Assemblies Review
- 12. Vehicle Supervisory Control and Data Acquisition (VSCADA)**
  - a. Daemon
  - b. Interfacing
  - c. User Applications
  - d. Data Storage
13. Dynamometer (DYNO)
  - a. Decomposition and Definition
  - b. Integration and Recomposition



# VCI

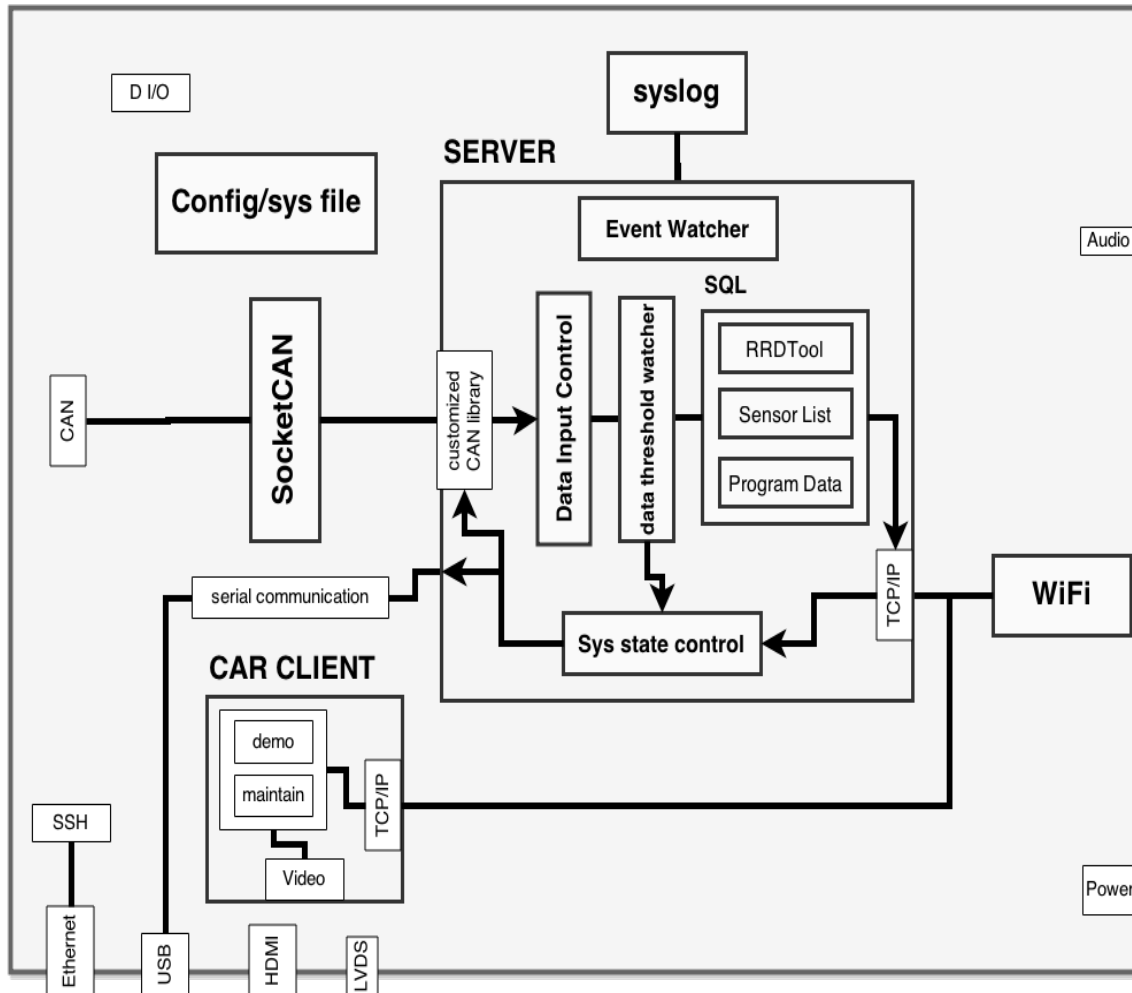


# Embedded Computer: VAB-820

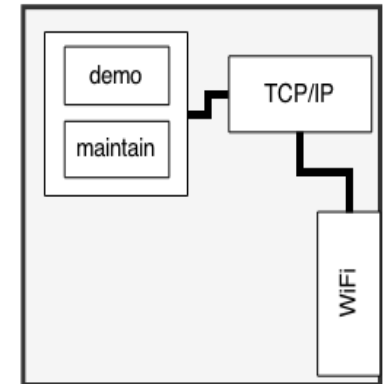


# VSCADA

## VSCADA Computer Linux system

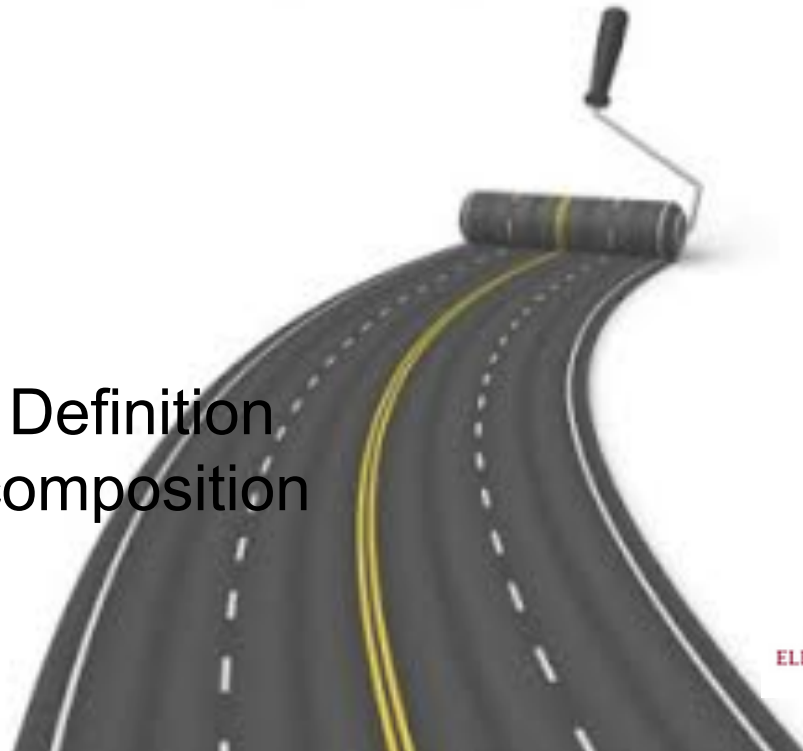


## PIT CLIENT



# Roadmap

10. Meet the Afternoon Teams
11. Interface Control and Assemblies Review
12. Vehicle Supervisory Control and Data Acquisition (VSCADA)
  - a. **Daemon**
  - b. Interfacing
  - c. User Applications
  - d. Data Storage
13. Dynamometer (DYNO)
  - a. Decomposition and Definition
  - b. Integration and Recomposition



# Logging/Daemons - Main Program

- VSCADA uses systemd to initially launch the main program
  - systemd has most major linux distributions support
- The main program will run in the background as the server with PID registered
- The main program will start by doing system startup procedures



# Logging/Daemons - Main Program

```
[Unit]
```

```
Description=vscada main program
```

```
After=default.target
```

```
[Service]
```

```
ExecStart=/usr/bin/python3 /home/vscada/main.py
```

```
Type=simple
```

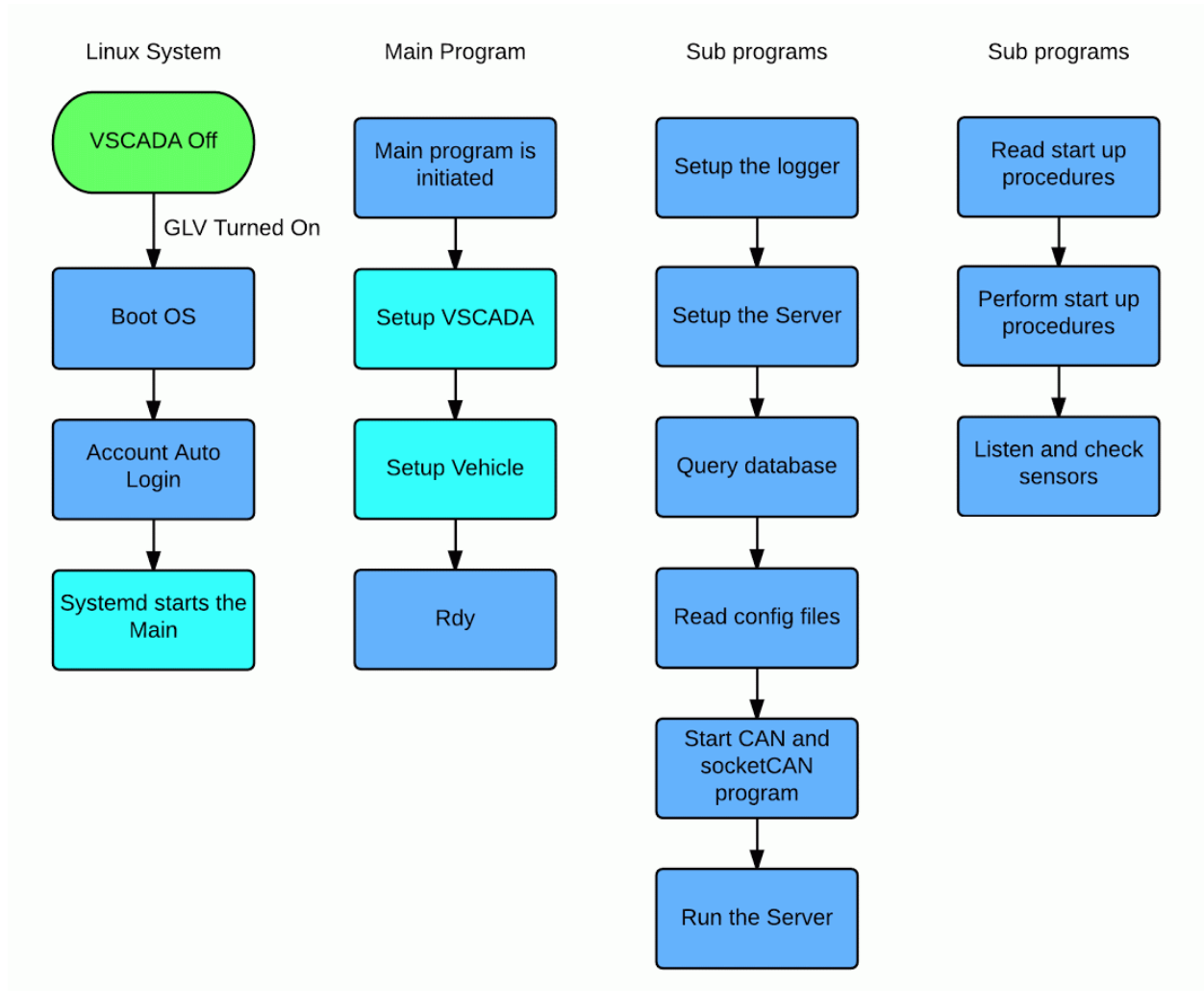
```
WorkingDirectory=/home/vscada
```

```
[Install]
```

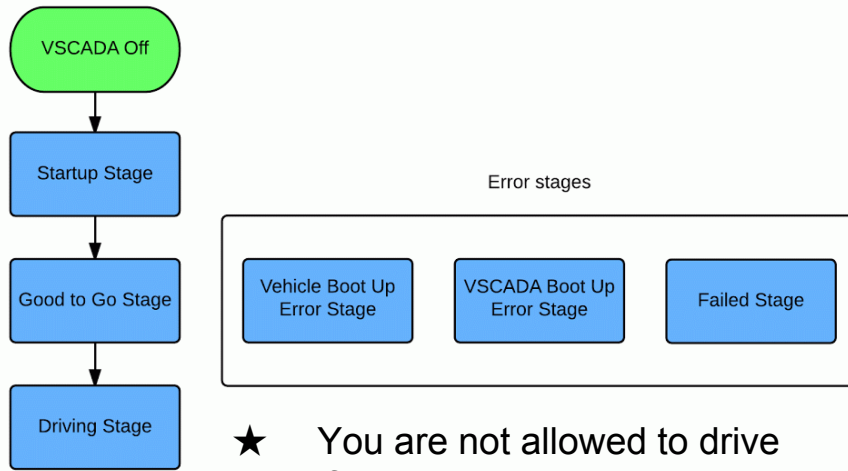
```
WantedBy=multi-user.target
```



# System Startup



# System States and Exceptions



★ You are not allowed to drive if in these error stages

## Startup Stage

- When system boots up and will go to one of the following stages

## Good to Go Stage

- No error or warning and is ready to be driven

## Driving Stage

- The car is driving

## Vehicle Boot Up Error Stage

- VSCADA is functional but other subsystems are not

## VSCADA Boot Up Error Stage

- VSCADA is not functional

## Failed Stage

- VSCADA failed to boot up

# System States and Exceptions

## System Errors:

### Syntax error

→ Failed Stage

### VehicleStartupConfigLoadException

→ Other Boot Up Error Stage

### DatabaseLoadException

→ VSCADA Boot Up Error Stage

### RRDFileNotFoundExcepion

→ VSCADA Boot Up Error Stage

### VehicleStartupTimeoutException

→ Other Boot Up Error Stage

### SensorCheckingTimeoutException

→ Other Boot Up Error Stage

### SystemFailureError

→ Failed Stage

### OtherCommunicationException

→ Vehicle Boot Up Error Stage

### OtherSystemException

→ VSCADA Boot Up Error Stage

## Sensor Errors:

### Logic:

Errors are configurable and specific

If happens before driving, then the car is disabled from driving;

else take actions according to sensor configuration

### Possible Errors:

OverHeating

UnderCharged

### Possible Actions:

Do Nothing

Send warning upward

pull safety relay

```
try{  
  
}catch( Exception ){  
    //Do nothing  
}
```



Exceptions...

Gotta catch 'em all!

# Logging/Daemons - Logging

- Have 5 levels, in their respective order:
  - Debug: detailed information, mainly used for debugging
  - Info: general information, should contain important data
  - Warning: Need user's attention
  - Error: Need user to check the source of the error
  - Critical: Opps.
- Logs are stored in syslog of Linux
  - syslog handles storage, update, filter, etc.
  - Python and other library support for syslog
- Can be viewed by clients over the TCP protocol
- Levels can be set by configuration. Info level by default

# Logging/Daemons - Logging - Updated

- Logs are stored in syslog of linux by default
- They can also be stored in local files, printed to console or remotely to other computer
- The exact method of logging storage is configurable per each logger, as shown on the next slide

# Configuration

```
[loggers]
```

```
keys=root
```

```
[handlers]
```

```
keys=sysLogHandler,consoleHandler
```

```
[formatters]
```

```
keys=simpleFormatter
```

```
[logger_root]
```

```
level=INFO
```

```
handlers=sysLogHandler
```

```
[handler_sysLogHandler]
```

```
class=logging.handlers.SysLogHandler
```

```
level=INFO
```

```
formatter=simpleFormatter
```

```
args=('/dev/log',)
```

```
[handler_consoleHandler]
```

```
class=StreamHandler
```

```
level=DEBUG
```

```
formatter=simpleFormatter
```

```
args=(sys.stdout,)
```

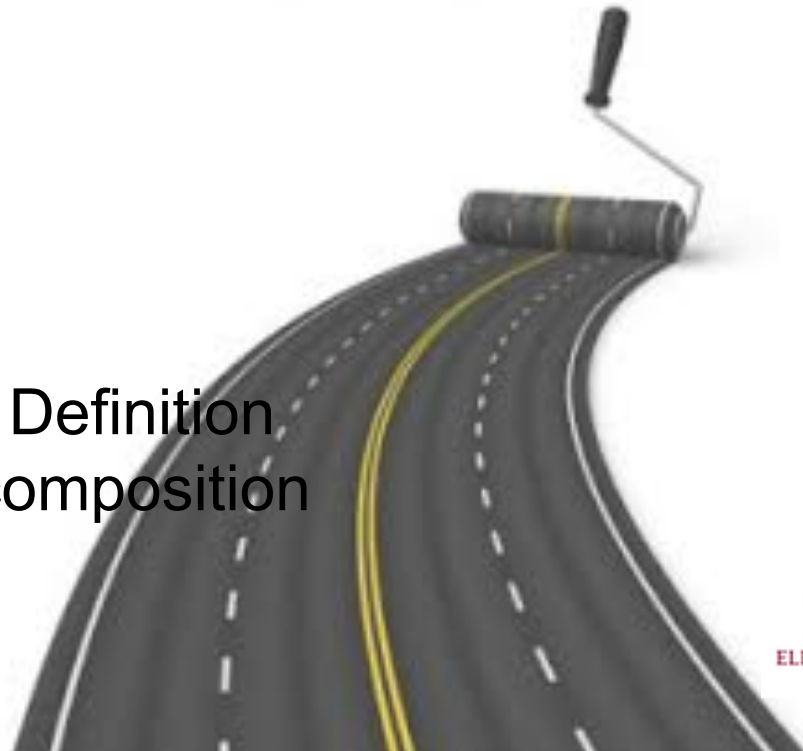
```
[formatter_simpleFormatter]
```

```
format=%(asctime)s - %(name)s - %  
(levelname)s - %(message)s
```

```
datefmt=
```

# Roadmap

10. Meet the Afternoon Teams
11. Interface Control and Assemblies Review
12. Vehicle Supervisory Control and Data Acquisition (VSCADA)
  - a. Daemon
  - b. Interfacing**
  - c. User Applications
  - d. Data Storage
13. Dynamometer (DYNO)
  - a. Decomposition and Definition
  - b. Integration and Recomposition



# Interfaces

- TSV Communication
  - Send/Receive Packets of data from the four PACMAN via CAN
  - will follow Application Layer Protocol
- GLV Communication
  - JGB act as a hub for groups of sensors via CAN
- DYNO Communication
  - Control Throttle via JGB board
  - Motor Controller CAN
  - Dynamometer - USB interface to read RPM and torque, set valve



# Interfaces - Updated

- TSV Communication
  - Send/Receive Packets of data from the four PACMAN via CAN
  - will follow Application Layer Protocol
- GLV Communication
  - JGB act as a hub for groups of sensors via CAN
- DYNO Communication
  - Control Throttle via JGB board
  - Motor Controller CAN
  - Dynamometer - USB interface to read RPM and torque, set valve
- Safety Loop control via CAN
- A virtual CAN driver is used for testing

# CAN Interface

- SocketCAN -Linux Drivers

```
sam@bull3:~/Desktop/vscada/can-lib$ cansend vcan0 123#11.02.fe.fe.ee.ee.95.33
```

```
sam@bull3:~/Desktop/vscada/can-lib$ candump vcan0
vcan0 123 [8] 11 22 33 44 55 66 77 88
vcan0 123 [8] 11 22 33 44 55 66 77 88
vcan0 001 [3] 11 12 13
```

- Python-CAN

```
sam@bull3:~/Desktop/vscada/can-lib$ python3 CANexample.py vcan0
Received: can id=123, can dlc=8, data=b'\x01\x02\xff\xff\xee\xee\x952'
```

# CAN Interface - Updated

- SocketCAN -Linux Drivers

```
sam@bull3:~/Desktop/vscada/can-lib$ cansend vcan0 123#11.02.fe.fe.ee.ee.95.33
```

```
sam@bull3:~/Desktop/vscada/can-lib$ candump vcan0
vcan0 123 [8] 11 22 33 44 55 66 77 88
vcan0 123 [8] 11 22 33 44 55 66 77 88
vcan0 001 [3] 11 12 13
```

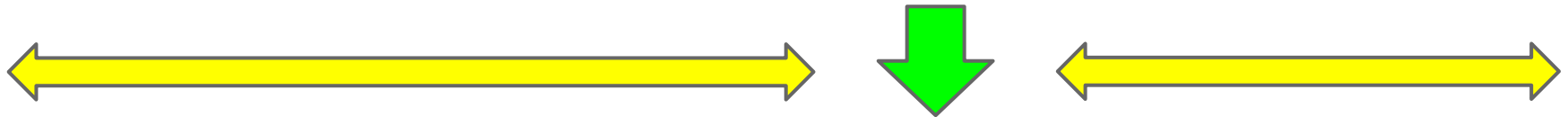
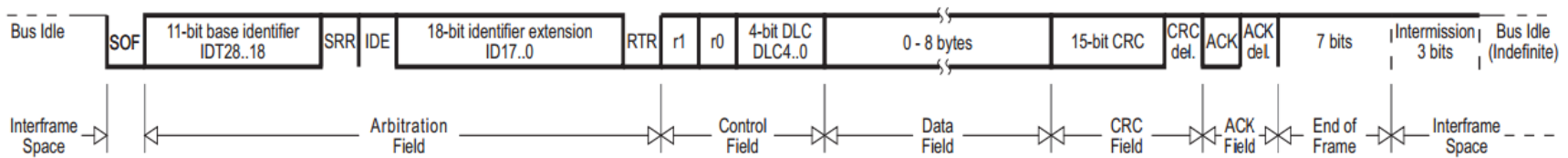
- Python-CAN

```
sam@bull3:~/Desktop/vscada/can-lib$ python3 CANexample.py vcan0
Received: can id=123, can dlc=8, data=b'\x01\x02\xff\xff\xee\xee\x952'
```

Architecture: Server will send a Remote request, can device will respond following the data format in the next slide

# JGB - CAN Frame

## Data frame



BYTE 0	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7
--------	--------	--------	--------	--------	--------	--------	--------

CAN ID: Cockpit, TSI, GLV\_Power  
200 250 275

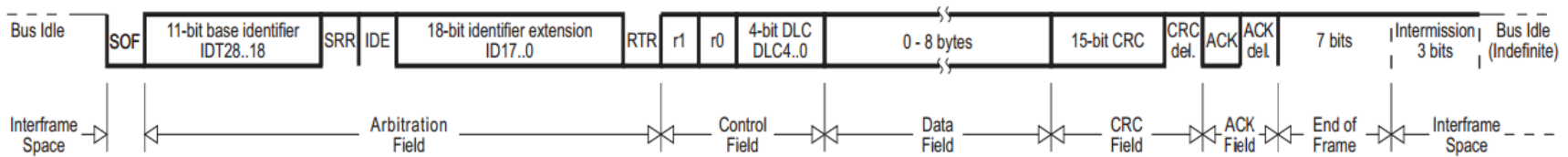
Header: Sensor Id

Byte 0 - Header  
Byte 1 - Measurement (High)  
Byte 2 - Measurement (Low)

Measurements:  
Ambient Temp  
LED status  
Temperature  
Voltage Current  
Temperature SOC

# JGB/TSV CAN Frame (Updated)

## Data frame

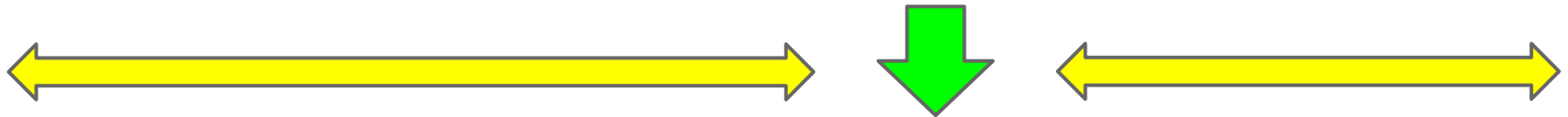
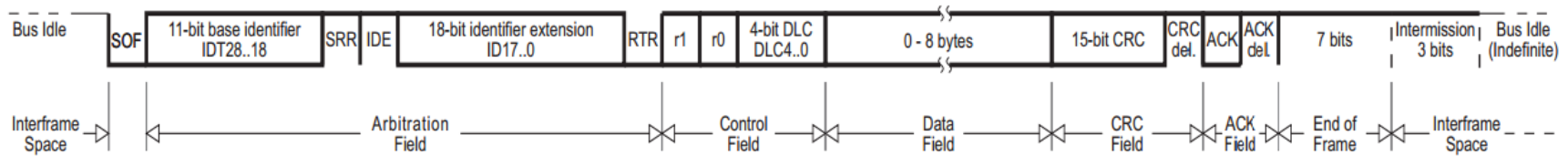


BYTE 0	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7
Mask	Empty	Slot 1		Slot 2		Slot 3	

- Slots will hold measured data
- Mask will define which Slot should be read by server

# PACMAN - CAN Frame - Deprecated

## Data frame



BYTE 0	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7
--------	--------	--------	--------	--------	--------	--------	--------

CAN ID: PACKMAN

1, 2, 3, 4  
110, 120, 130, 140

Header:

AMS # (Can be Zero)

Byte 0 - Header

Byte 1 - Measurement (High)

Byte 2 - Measurement (Low)

Measurement:

Voltage

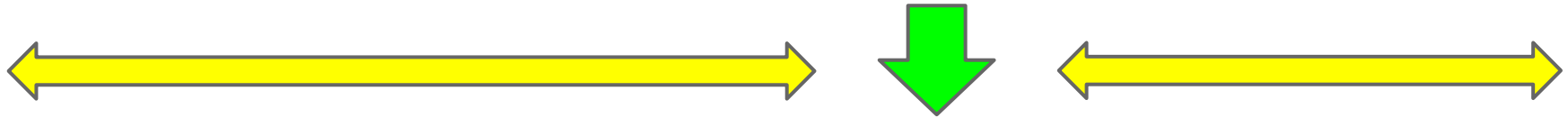
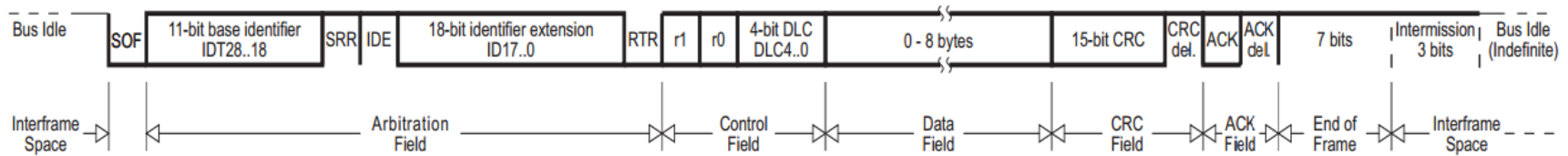
Current SOC

Fuse Temp

Temperature

# Motor Controller CAN Frames (1 / 2)

## Data frame



BYTE 0	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7
--------	--------	--------	--------	--------	--------	--------	--------

Byte 0 - RPM (High)

Byte 1 - RPM (Low)

Byte 2 - Motor Temp

Byte 3 - Controller Temp

Byte 4 - RMS Current (High)

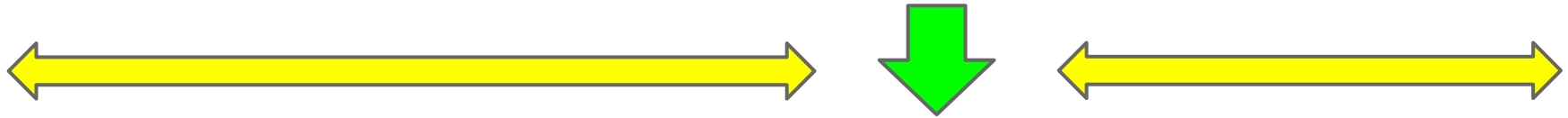
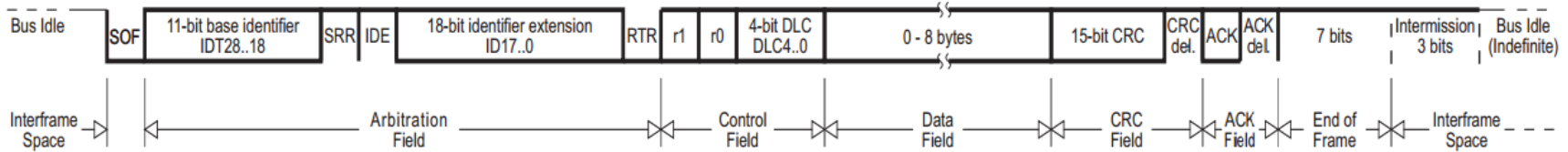
Byte 5 - RMS Current (Low)

Byte 6 - Capacitor V (High)

Byte 7 - Capacitor V (Low)

# Motor Controller CAN Frames (2 / 2)

## Data frame



BYTE 0	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7
--------	--------	--------	--------	--------	--------	--------	--------

Byte 0 - Stator Freq (High)

Byte 1 - Stator Freq (Low)

Byte 2 - Controller Fault P

Byte 3 - Controller Fault S

Byte 4 - Throttle Input

Byte 5 - Brake Input

Byte 6 - System Bits

Byte 7 - (UNUSED)



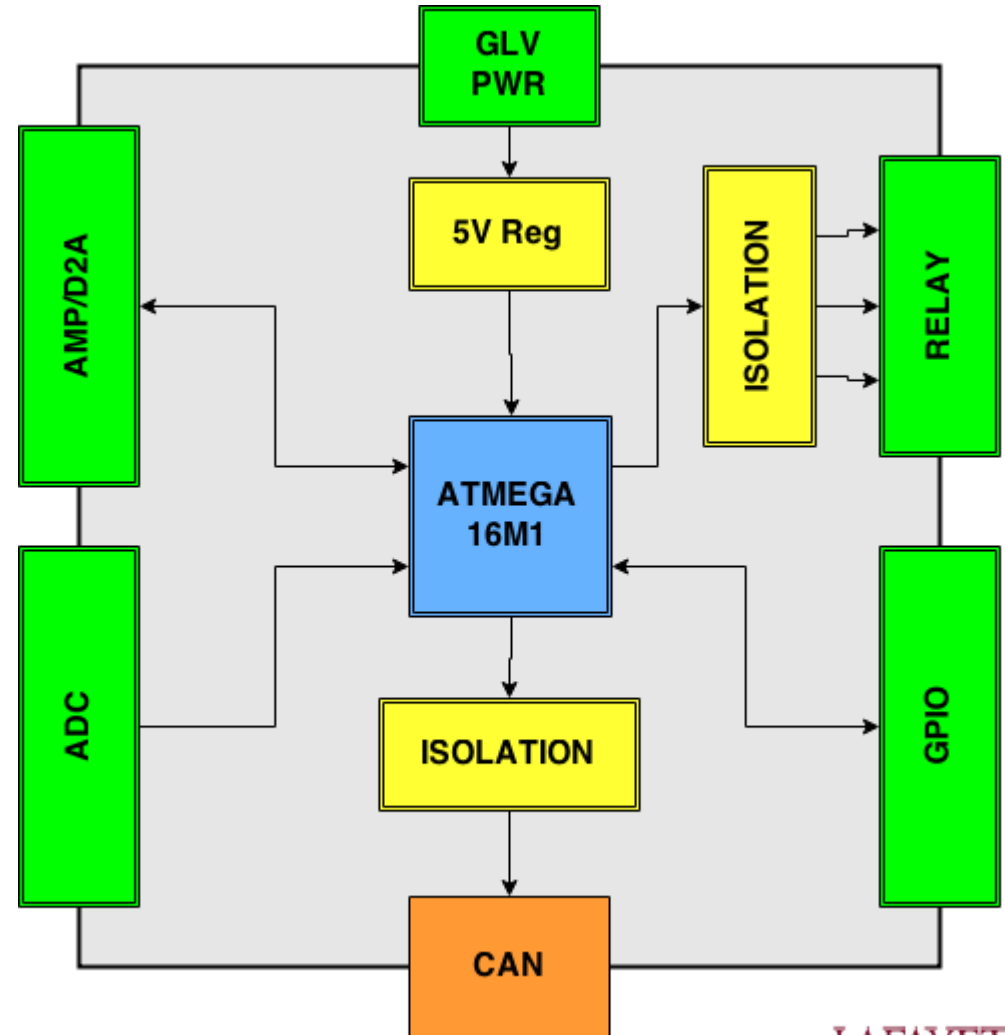
# CAN Microcontroller Board (JGB)

## Automotive AVR

- CAN Bus
- LIN
- UART (RS-232)

## Board Inputs/Outputs

- Internal Temperature
- 5 ADC Channels
- 3 PWM Channels
- 1 DAC Channel
- 6 GPIO/SPI
- 2 Differential ADC
- USB - UART



# Microcontroller Firmware Design

## UART

Send/Receive  
Test/Debugging Information

## CAN

SCADA Communication

## TIMER

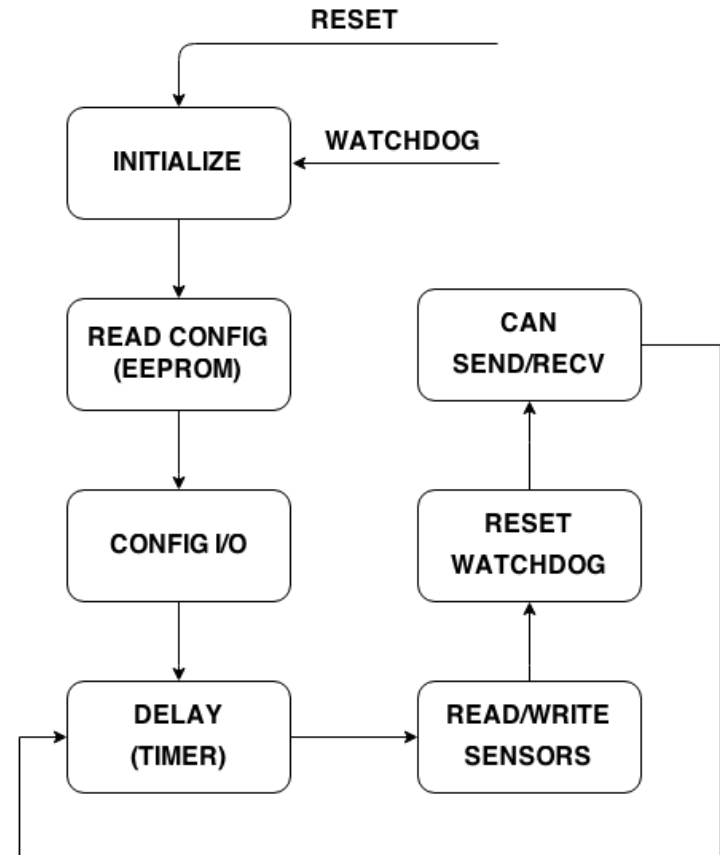
PWM, Sensor Timing

## WATCHDOG TIMER

Crash Prevention

## I/O

System Control Interface



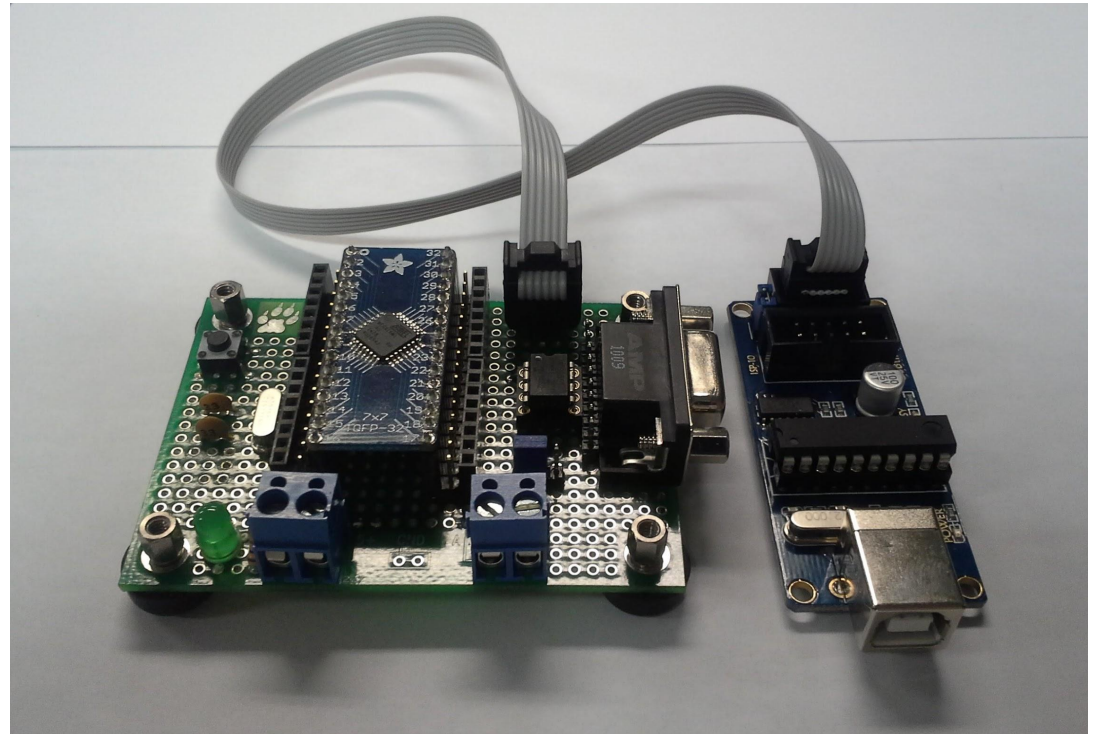
# Microcontroller Prototype Hardware

## WORKING:

- ADC
- D2A
- PWM
- GPIO

## NOT WORKING:

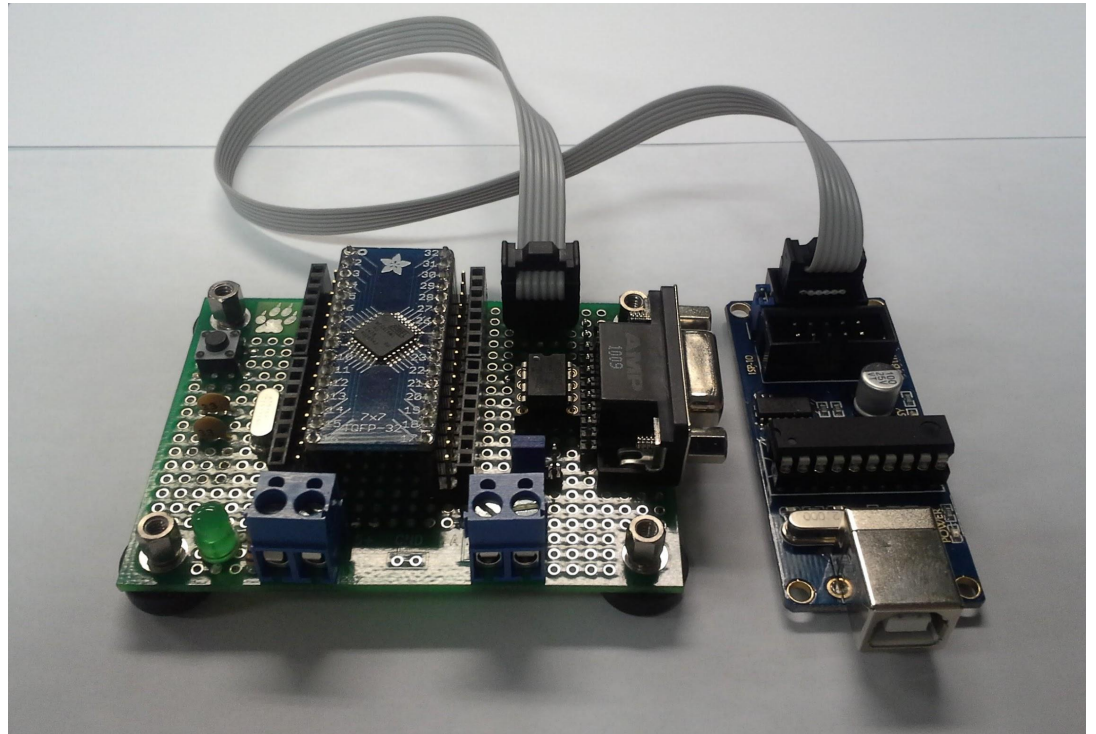
- UART
- CAN



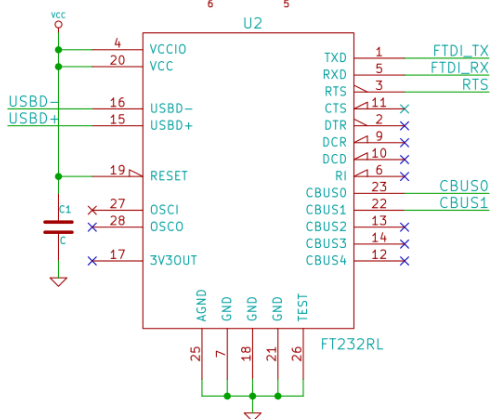
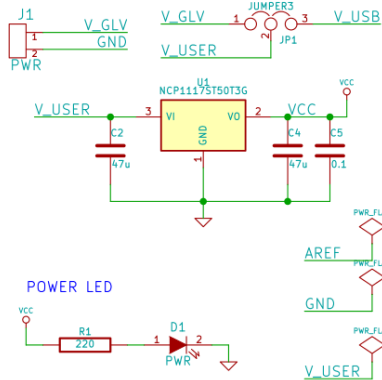
# Microcontroller Prototype Hardware - Update

## WORKING:

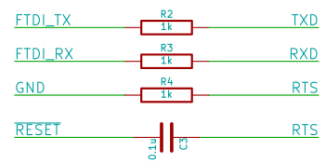
- ADC
- D2A
- PWM
- GPIO
- CAN
- UART



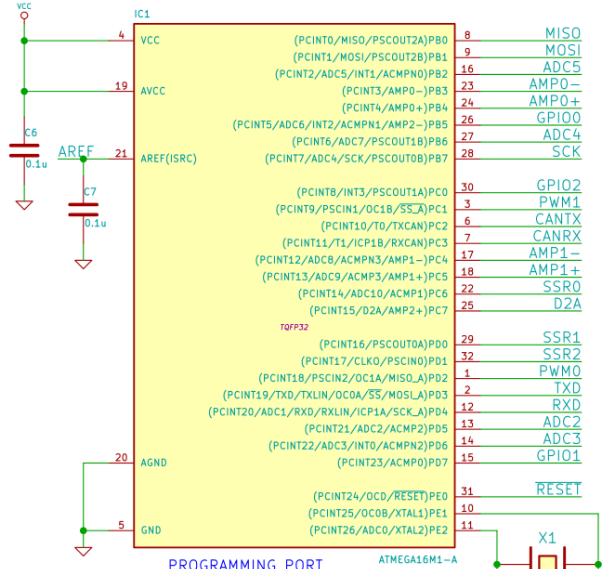
### POWER CONNECTIONS



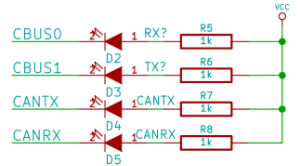
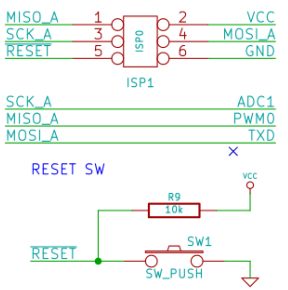
### FTDI <-> AVR



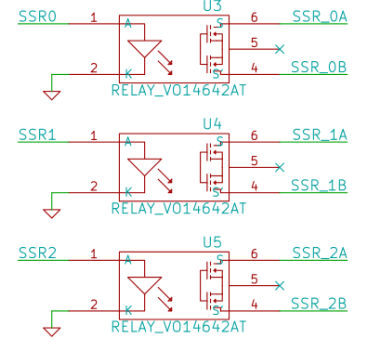
### ATMEGA 16M1 MICROCONTROLLER



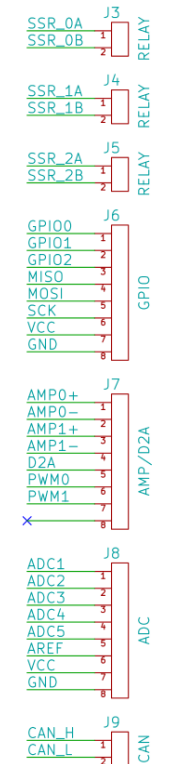
### PROGRAMMING PORT



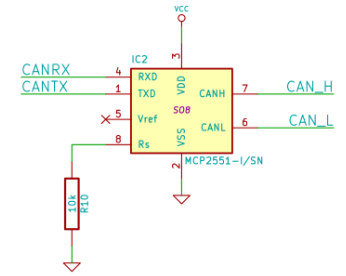
### AC/DC Relay



### I/O HEADERS

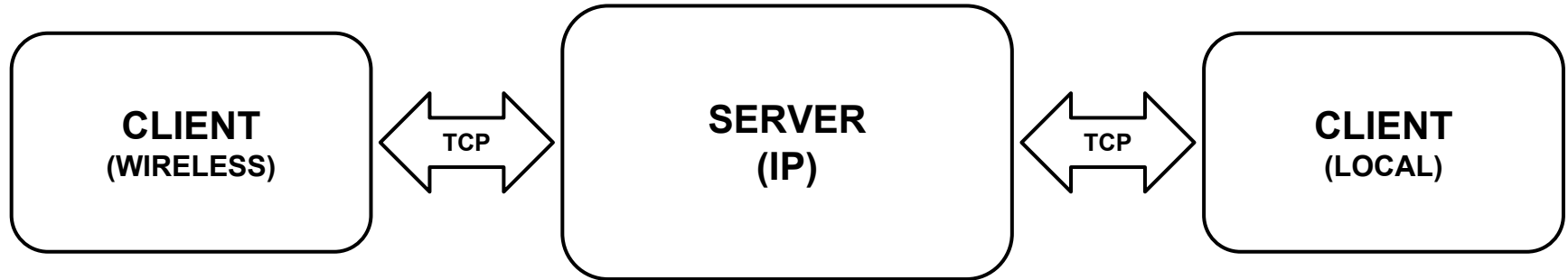


### CAN TRANCEIVER



<b>ATMEGA 16M1 Automotive AVR</b> Engineer: John Gehrig Lafayette College File: ATMEGA16M1.sch Sheet: /		
<b>Title: LFEF CAN Communication Board</b>		
Size: USLetter	Date: 10 mar 2015	Rev: 0
KiCad E.D.A.		Id: 1/1

# Client - Server Architecture



## Request - Response Model

- Client Initiates Request
- Server Responds to Request
- JSON Object Passing

## Unix-Style Commands

- Modular, Flexible, Expandable

# Server Command Architecture

## Server Command Syntax:



### **CMD NAME -**

Unique command name, identifies specific server task to carry out.

### **FLAGS -**

Enables or disables specific command functionality or output.

### **OPTIONS -**

Utilized to pass data from the client to server.

### **ARGUMENTS -**

End objects affected by the server command.

## Syntax Notes:

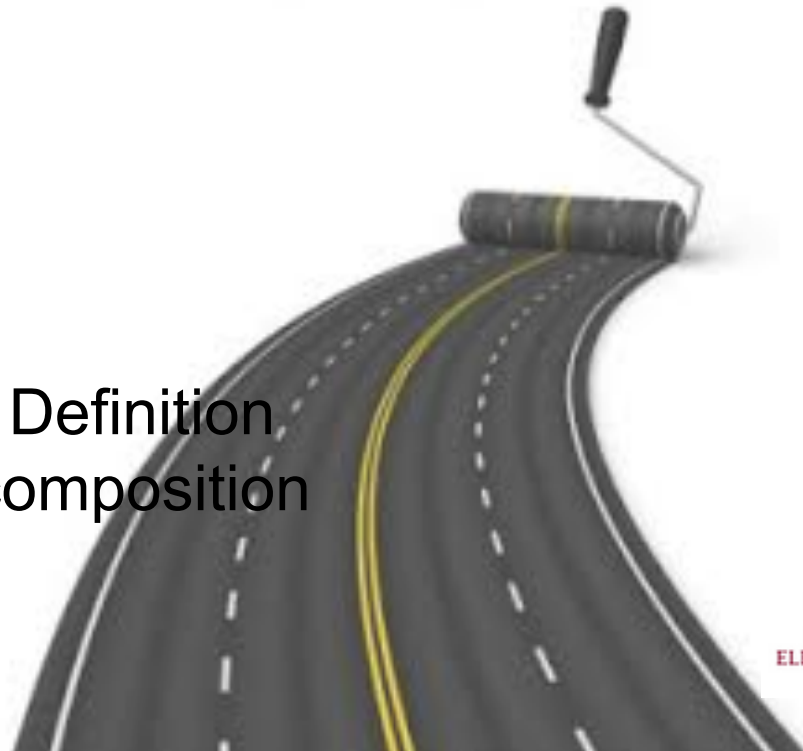
All command Options, Flags, Arguments space separated.

Flags begin with the “-” character.

Options are followed by a string containing no spaces.

# Roadmap

10. Meet the Afternoon Teams
11. Interface Control and Assemblies Review
12. Vehicle Supervisory Control and Data Acquisition (VSCADA)
  - a. Daemon
  - b. Interfacing
  - c. User Applications**
  - d. Data Storage
13. Dynamometer (DYNO)
  - a. Decomposition and Definition
  - b. Integration and Recomposition





# Front-End User Application


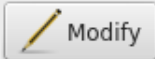

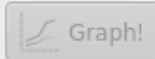
- VSCADA Maintenance Application
  - Contains all required user functionality in one program
  - Runs on remote PC (pit station PC) and vehicle embedded computer with touchscreen or optional USB input devices (mouse, keyboard, etc.)
  - Demo mode can be selected in the maintenance application
  - Password is used to protect maintenance mode from unauthorized access

# Maintenance App - Measurands/Input

VSCADA Maintenance - [Preview]





Measurands/Input Hardware/Output Rules Settings

Measurand	Type	Value	Adj Value	Units
▼ TSV				
▼ Battery Pack 0				
▼ Cell0				
Voltage	Analog In	3.3	3.3	V
Current	Analog In	10	10	A
SOC	Analog In	700	70	%
Temperature	Analog In	70	35	C
▶ Cell1				
▶ Battery Pack 1				
▶ Battery Pack 2				
▶ Battery Pack 3				
▶ GLV				

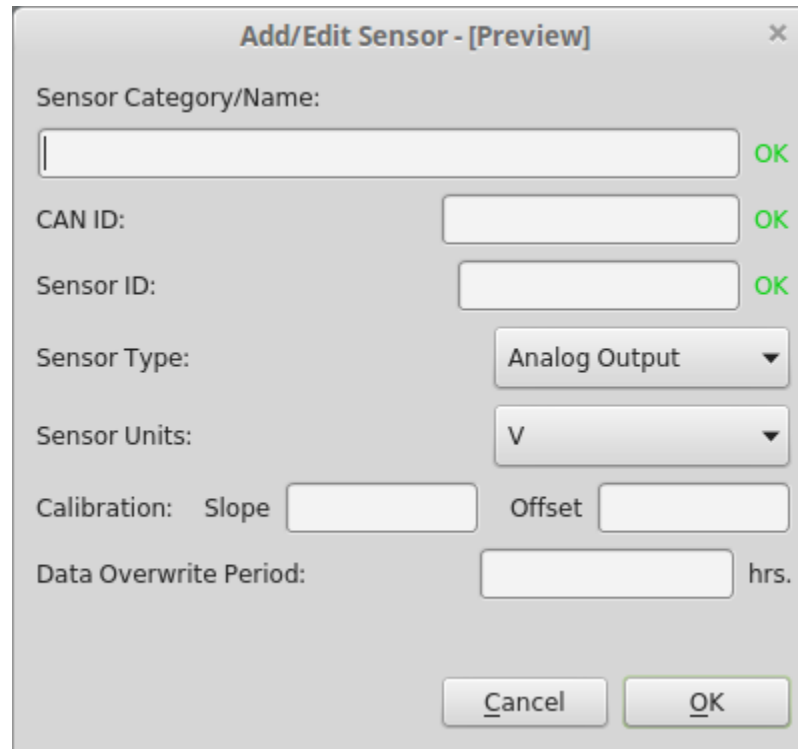
Add or Modify Vehicle Input:  Add  Modify  Delete  Graph!

.....

This is a message.

 Messages  Warnings  Errors  Failures

# Maintenance App - Add/Edit Sensor Window

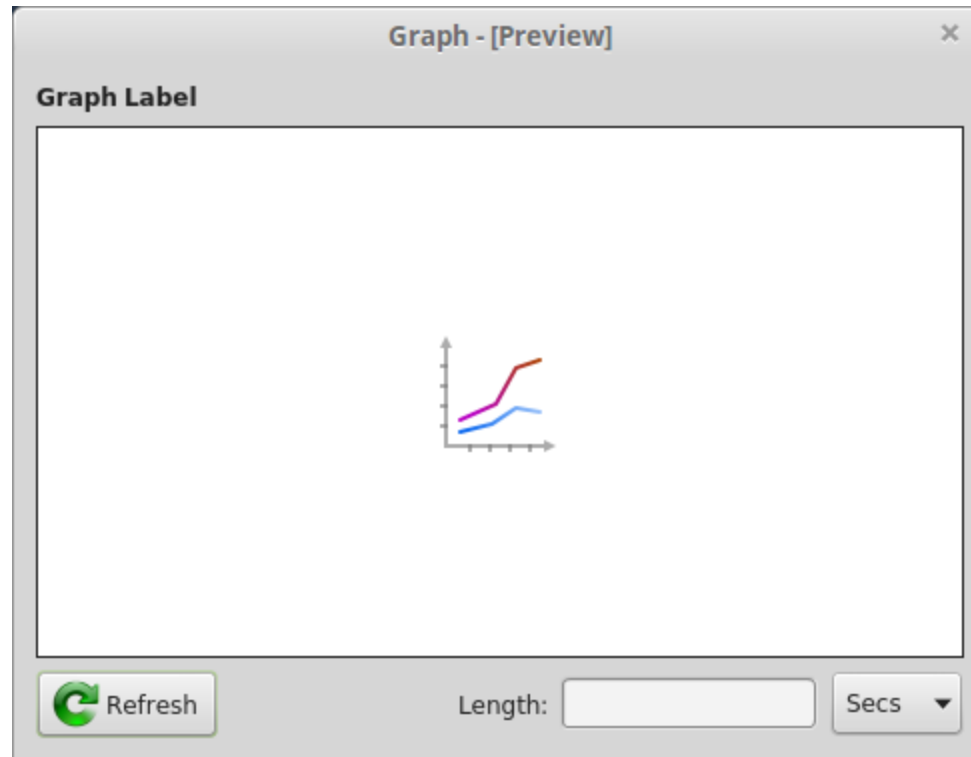


The screenshot shows a dialog box titled "Add/Edit Sensor - [Preview]" with a close button (X) in the top right corner. The dialog contains the following fields and controls:

- Sensor Category/Name:** A text input field with a green "OK" button to its right.
- CAN ID:** A text input field with a green "OK" button to its right.
- Sensor ID:** A text input field with a green "OK" button to its right.
- Sensor Type:** A dropdown menu currently showing "Analog Output".
- Sensor Units:** A dropdown menu currently showing "V".
- Calibration:** Two text input fields labeled "Slope" and "Offset".
- Data Overwrite Period:** A text input field followed by the unit "hrs.".

At the bottom of the dialog are two buttons: "Cancel" and "OK".

# Maintenance App - Measurand Graph Window



# Maintenance App - Hardware/Output

VSCADA Maintenance - [Preview] ×

Measurands/Input Hardware/Output Rules Settings

Name	Type	Value
Throttle	Analog Out	78%
SL VSCADA Relay	Digital Out	On
SL TSI Relay	Digital Out	On

Add or Modify Vehicle Output: + Add ✎ Modify ✖ Delete

.....

This is a message.

Messages ⚠ Warnings ✖ Errors 🐛 Failures

# Maintenance App - Rules

VSCADA Maintenance - [Preview]

Measurands/Input Hardware/Output **Rules** Settings

Measurand	Priority	Upper Bound	Lower Bound
▼ TSV			
▼ Battery Pack 0			
▼ Cell0			
▼ Voltage			
Warning		3.6	3.0
Error		3.8	2.8
Failure		4.0	2.6
▶ Current			

Response Actions

Add or Modify Response Action: + Add ✎ Modify ✖ Delete

This is a message.

Messages ⚠ Warnings ✖ Errors 🐛 Failures

# Maintenance App - Settings

VSCADA Maintenance - [Preview] ×

Measurands/Input Hardware/Output Rules **Settings**

IP Address  .  .  .

Demo Mode





Password

System Logging

Save Settings

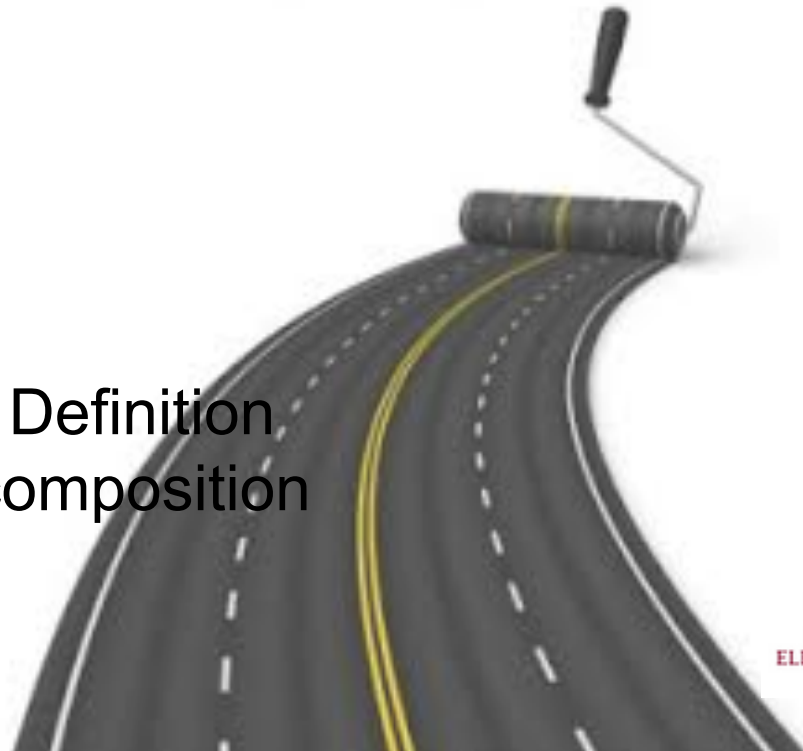
.....

This is a message.

 Messages  Warnings  Errors  Failures

# Roadmap

10. Meet the Afternoon Teams
11. Interface Control and Assemblies Review
12. Vehicle Supervisory Control and Data Acquisition (VSCADA)
  - a. Daemon
  - b. Interfacing
  - c. User Applications
  - d. Data Storage
13. Dynamometer (DYNO)
  - a. Decomposition and Definition
  - b. Integration and Recomposition





# Round Robin Database (RRD)

- High performance data logging and graphing system for time series data
- Uses circular buffer to store data
  - Data size does not expand with time.
  - Overwrites the data once it reach the starting point
- Framework for storing measurement averages, min,max and derivative
- Graphical presentation for both stored and archived data.

# RRD Creation

- Size of the database can be determined at creation time.
- Specify the step time (rate at which the database update the data)
- Specify the step time for archives too. Different time steps can be applied for each archive.

```

data_sources=[ 'DS:speed:GAUGE:10:0:100',|
               'DS:temperature:GAUGE:10:0:100',
               'DS:chargelevel:GAUGE:5:10:100',
               'RRA:AVERAGE:0.5:2:%s' % str(step*datapoints),
               'RRA:MAX:0.5:2:%s' % str(step*datapoints)]

ret = rrdtool.create( rrd_db,
                     '--step', str(step),
                     '--start', str(start_time),
                     data_sources
                     )

```

```

for i in xrange(0, 1000, 0.523) :

    print i
    updated_speed = 10*math.sin(math.radians(i)) +10

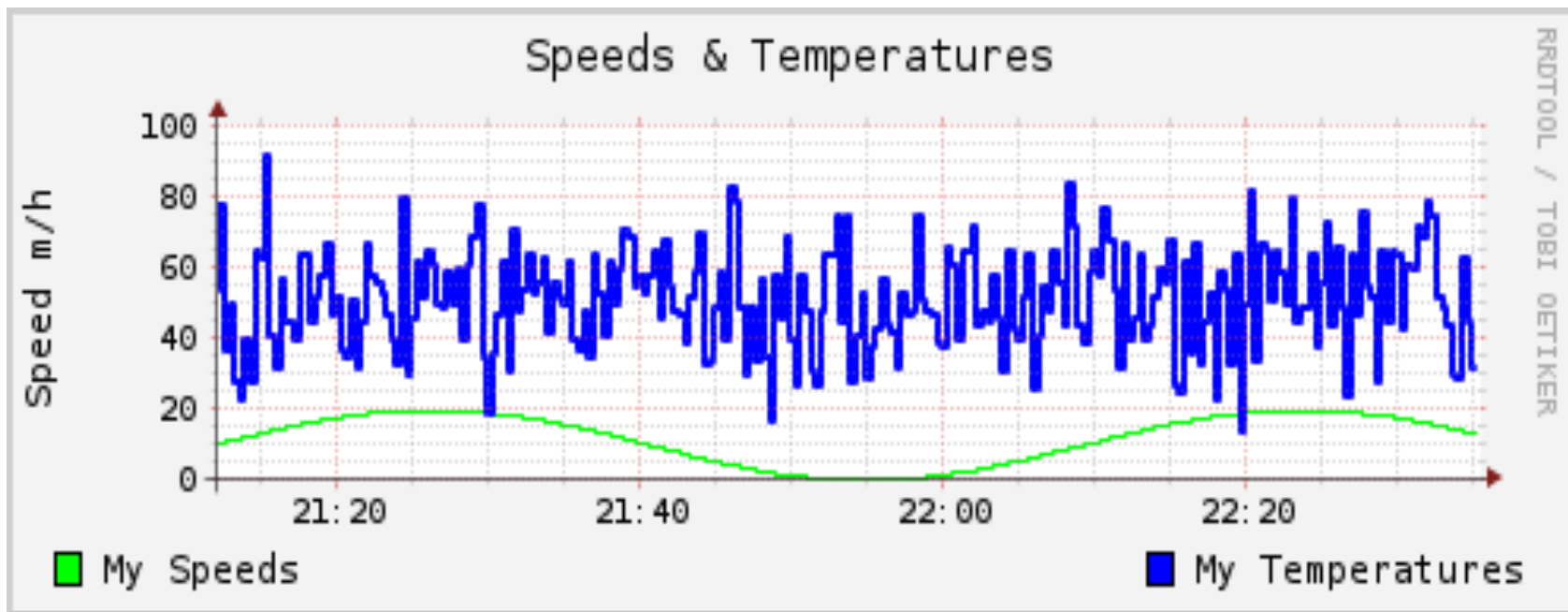
    # updated_speed = random.randrange(0,100)
    updated_temperature = random.randrange(1,100)
    updated_battery = random.randrange(1,100)
    time_stamp += 5;
    print time_stamp, updated_battery, updated_temperature, updated_speed
    ret1 = rrdtool.update(rrd_db, '%d:%d:%d:%d' % (time_stamp,updated_speed, updated_temperature, updated_battery))

```

```

rrdtool.graph( speed_graph,
  '--start' , str(start_time) ,
  '--end' , str(end_time),
  '--vertical-label' , 'Speed m/h' ,
  '--imgformat' , 'PNG' ,
  '--title' , 'Speeds & Temperatures' ,
  '--lower-limit' , '0',
  'DEF:var1=%s:speed:AVERAGE' % rrd_db,
  'DEF:var2=%s:temperature:AVERAGE' % rrd_db,
  'LINE1:var1#00FF00:My Speeds',
  'LINE2:var2#0000FF:My Temperatures'
)

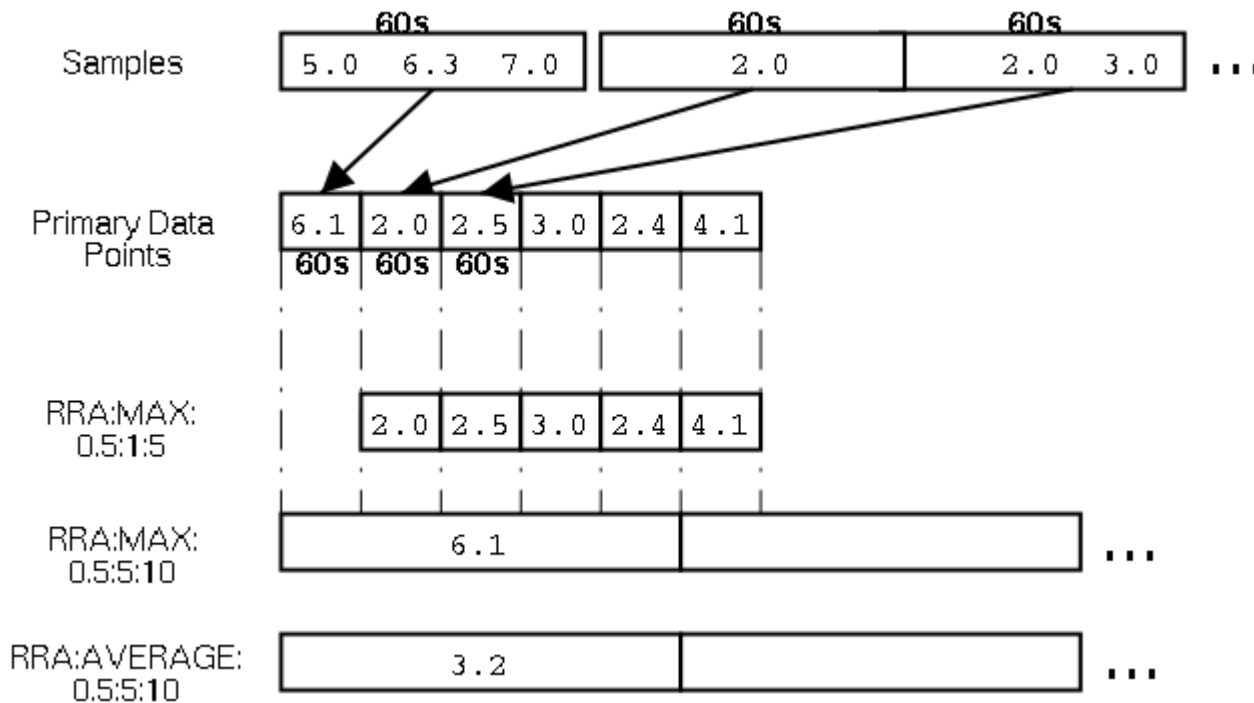
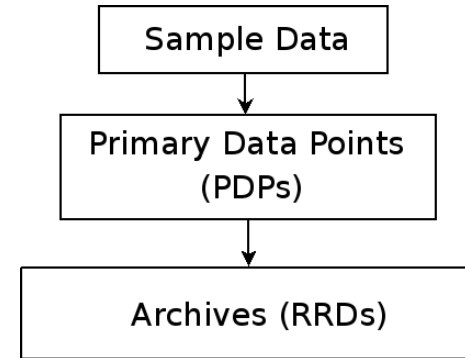
```



# Round Robin Archives (RRA)

- Average
- Minimum
- Maximum
- Last

Data Source



# RRD for this project - Update

- Monitor the time series data.
- Take care of time and space complexity.
- Very simple in structure.
- Manipulate the stored data and archive the data. Only store raw values and calibrate during retrieval and when graphing (saves space and very little performance hit)
- Graphing tools. Images of graphs will be sent over TCP; want to avoid client-side databases

# Database and Configuration

- Database used for sensor list and most of the configurations
- Text files used for setting internal logic (startup procedures, logic switches, etc)

# Database

- SQLite chosen for speed and simplicity
- Database auto generating and tearing down implemented for testing
- Tables used include the following:
  - Table 1: restoring sensor information
    - sensor hierarchy
    - CAN id
    - sampling rate
    - rrd file reference
      - This means, all data is going to be stored in RRD, but a reference is kept in SQL as a cleaner solution
  - Table 2: type of sensor
    - analog in, analog out, digital in, digital out
    - need to know this for sending out data on CAN

Note: the speed of the queries is yet to be tested.



# Database

- Table 3: warning/error threshold
  - High and low values for warnings, errors and failures
  - Reaching these values will trigger some certain actions, which is referred in the next table
- Table 4: warning/error actions
  - Each of the actions here is generic and configurable
- Table 5: calibration
  - have slop and offsets

# SQL DB: 'Sensor\_Table'

Name	id_CAN	id_Sensor	Type	Sample_Rate
Overwrite_Period	Units	Factor	Offset	RRD_DB

**id**

Unique numerical identifier for each sensor.

**id\_CAN**

Non-Unique 11-bit CAN bus id

**id\_Sensor**

Non-Unique sensor id

**Type**

Enumerated value from 'Sensor\_Type'

**Sample\_Rate**

How many often the sensor is read/written from over a 60-second period of time.

**Overwrite\_Period**

Persistence of data in hours.

**Units**

Enumerated unit from 'Unit\_Table'

**Factor**

Sensor scaling factor (double).

**Offset**

Sensor offset factor (double)

**RRD\_DB**

Sensor RRD database location pointer.

# SQL DB: 'Sensor\_Type'

id	Description	Type	Direction
----	-------------	------	-----------

**id**

Unique numerical identifier link to 'Sensor\_Table' for each sensor.

**Description**

A human readable description of the sensor/output.

**Type**

Sensor type description. Boolean value {0 - Analog, 1 - Digital}

**Direction**

Directionality of sensor. Boolean value {0 - Output, 1 - Input}

# SQL DB: 'Sensor\_Levels'

id	Warning_Low	Warning_High	Error_Low	Error_High	Fail_Low	Fail_High
----	-------------	--------------	-----------	------------	----------	-----------

id

Unique numerical identifier link to 'Sensor\_Table' for each sensor.

Warning\_Low

A value below this threshold triggers a 'WARNING' state.

Warning\_High

A value above this threshold triggers a 'WARNING' state.

Error\_Low

A value below this threshold triggers an 'ERROR' state.

Error\_High

A value above this threshold triggers an 'ERROR' state.

Fail\_Low

A value below this threshold triggers a 'FAILURE' state.

Fail\_High

A value above this threshold triggers a 'FAILURE' state.

# SQL DB: 'Sensor\_Actions'

id	Action_Name	Priority_Level	Effector_Name	Effector_State
----	-------------	----------------	---------------	----------------

**id**

Non-Unique numerical identifier link to 'Sensor\_Table' for each sensor.

**Action\_Name**

A human readable name for the rule.

**Priority\_Level**

A number between 0-5 describing the priority trigger for the action.

**Effector\_Name**

An identifier link to 'Sensor\_Table' for the effector sensor.

**Effector\_State**

A numerical value which the Effector should be set to on occurrence of the event.

# SQL DB: 'Unit\_Table'

id	Unit Description	Unit_Abrv
----	------------------	-----------

**id**

Unique identifier for each system unit.

**Unit\_Description**

A human readable description of the unit.

**Unit\_Abrv**

An abbreviation to use when displaying the unit.

# SQL DB: 'Sensor\_Actions'

id	Name	id_CAN	id_Sensor	Type	Sample_Rate	Overwrite_Period	Units	Factor	Offset	RRD_DB
1	TSV/Pack1/Voltage	0	1	Analog	20	24	V	1	0	/data/TSV/pack1/voltage.rrd
2	TSV/Pack1/Current	1	2	Analog	21	25	A	1	0	/data/TSV/pack1/current.rrd
3	TSV/Pack1/SOC	2	3	Analog	22	26	%	1	0	/data/TSV/pack1/SOC.rrd
4	TSV/Pack1/Fuse_Temperature	3	4	Analog	23	27	°C	1	0	/data/TSV/pack1/fuse_temperature.rrd
5	TSV/Pack1/AMS1/Temperature	4	5	Analog	24	28	°C	1	0	/data/TSV/pack1/AMS1/temperature.rrd
6	TSV/Pack1/AMS1/Voltage	5	6	Analog	25	29	V	1	0	/data/TSV/pack1/AMS1/voltage.rrd
7	TSV/Pack1/AMS1/Current	6	7	Analog	26	30	A	1	0	/data/TSV/pack1/AMS1/current.rrd
8	TSV/Pack1/AMS2/Temperature	7	8	Analog	27	31	°C	1	0	/data/TSV/pack1/AMS2/temperature.rrd
9	TSV/Pack1/AMS2/Voltage	8	9	Analog	28	32	V	1	0	/data/TSV/pack1/AMS2/voltage.rrd
10	TSV/Pack1/AMS2/Current	9	10	Analog	29	33	A	1	0	/data/TSV/pack1/AMS2/current.rrd
11	TSV/Pack1/AMS3/Temperature	10	11	Analog	30	34	°C	1	0	/data/TSV/pack1/AMS3/temperature.rrd
12	TSV/Pack1/AMS3/Voltage	11	12	Analog	31	35	V	1	0	/data/TSV/pack1/AMS3/voltage.rrd
13	TSV/Pack1/AMS3/Current	12	13	Analog	32	36	A	1	0	/data/TSV/pack1/AMS3/current.rrd

# Configuration

- Bash style
- Read during startup, and bad syntax will raise exceptions and the car will be disabled from driving
- Switches can be updated and modified by maintenance app
- Will be stored under same directory and database in a separate folder
- Is accessible from debug port



# Configuration

```
[loggers]
```

```
keys=root
```

```
[handlers]
```

```
keys=sysLogHandler,consoleHandler
```

```
[formatters]
```

```
keys=simpleFormatter
```

```
[logger_root]
```

```
level=INFO
```

```
handlers=sysLogHandler
```

```
[handler_sysLogHandler]
```

```
class=logging.handlers.SysLogHandler
```

```
level=INFO
```

```
formatter=simpleFormatter
```

```
args=('/dev/log',)
```

```
[handler_consoleHandler]
```

```
class=StreamHandler
```

```
level=DEBUG
```

```
formatter=simpleFormatter
```

```
args=(sys.stdout,)
```

```
[formatter_simpleFormatter]
```

```
format=%(asctime)s - %(name)s - %  
(levelname)s - %(message)s
```

```
datefmt=
```



# Acceptance Testing

- Show that VSCADA meets all requirements as both:
  - part of integrated LFEV system
  - standalone software system
- Strive for minimum amount of test configs/avoid recompiling software
- Main criteria:
  - Exception handling
  - Automated hardware detection/configuration
  - Logging, plotting and storing of measurands
  - Controlling system state

# Acceptance Testing (cont.)



Test configurations:

- Config A: VSCADA powered by 12 V power source
- Config B: VSCADA interfaced with GLV
- Config C: VSCADA interfaced with GLV and TSV
- Config D: VSCADA interfaced with GLV, TSV and DYNO

# Acceptance Testing (cont.)



## T000 - System Startup/Shutdown and GLV Data Logging

- Config B
- Tests:
  - Automatic startup without user interaction once GLV power is provided
  - Logging of GLV measurands
  - Keeping of backup in case of unexpected shutdown

# Acceptance Testing (cont.)



## T001 - Safety Checking/Exception Handling

- Config D
- Tests:
  - Lighting of Ready LED on cockpit if all subsystems are in a safe state when Ready-to-Drive button pressed
  - Lighting of warning LEDs to warn user and prevent drive mode being activated by Ready-to-Drive button if unsafe condition occurs (exception handling)
  - Examples are open safety loop, voltage threshold exceeded, temperature threshold exceeded, missing config file for sensors

# Acceptance Testing (cont.)



## T002 - Maintenance App Operation

- Config D
- Tests:
  - Requirement of proper user credentials to login to maintenance mode
  - Logging and storing of all subsystem measurands (TSV pack/cell voltages, currents, temperatures, GLV voltage, current, Dyno torque, RPM)
  - Allowing user to control all aspects of VSCADA such as disabling safety checks, disabling data logging, and programming individual shutdown rules



# Acceptance Testing (cont.)

## T003 - Drive Mode Operation

- Config D, then repeat with Config A (simulated throttle)
- Tests:
  - Accurate reporting of measurands while driving
  - Logging of exceptions should unsafe condition occur while driving
  - Demo operation of vehicle through software throttle if other subsystems not available

# Acceptance Testing (cont.)



## T004 - Pack Charging/Discharging

- Config C
- Tests:
  - Displaying that accumulator is charging
  - Displaying that accumulator is discharging



# Acceptance Testing (cont.)



## T005 - Reliability Test

- Config D
- Tests:
  - System can run through series of drive modes/simulations and maintenance configuration changes over period of 24 hours without failure

# Acceptance Testing (cont.)



## T006 - Maintainability Test

- Config D
- Tests:
  - Novice user can solve frequently occurring problem
  - Expert maintenance individual can solve unexpected problem
  - New sensors can be added to system without software recompilation
  - VSCADA software can be installed easily using “make/install” on different computer

# Schedule

## Week 9

### **Demonstration System Integration & Debugging**

System parts designed in the past six weeks will be integrated into a cohesive system demonstration for CDR, and for displaying system capabilities to other groups.

### **CAN Communication PCB Fabrication**

The General Sensor CAN Communication PCB GERBER files will be ready for fabrication and sent out for production.

## Week 10

### **Preliminary Demonstration System**

A primitive scada system will be functioning, and ready for demonstration to other groups. This system should be capable of allowing groups to test communications between themselves and the SCADA system in the future.

## Week 11

### **SCADA Server Maintenance Mode**

The main system server will be capable of performing all 'Maintenance Mode' tasks, and interfacing with all 'Maintenance Mode' client interfaces.

### **QA Report Submitted**

Deliverable **D006** (QA Report) will be submitted.

# Schedule cont'd

## Week 12

### **System Integration & Debugging**

Any remaining components not added to the SCADA system will be added at this time. Debugging and integration into other vehicle sub-systems.

### **SCADA Server Demo Mode**

The main system server will be capable of performing 'Demo Mode' tasks.

## Week 13

### **Final ATR Report Submitted**

Deliverable **D005** (ATR Report) will be submitted.

### **System Integration & Debugging**

Any remaining components not added to the SCADA system will be added at this time. Debugging and integration into other vehicle sub-systems.

### **Dynamometer Communication Library**

The main system is capable of sending messages to the Huff Box over serial ports.

# Schedule cont'd

## Week 14

### **System Integration & Debugging**

Any remaining components not added to the SCADA system will be added at this time. Debugging and integration into other vehicle sub-systems.

### **System Documentation**

All project documentation will be finalized and completed.

### **Completed Maintenance Manual Submitted**

A VSCADA Maintenance Manual Working Draft will be submitted.

## Week 15

### **Final Report & Maintenance Manual Submitted**

Deliverable **D003** (Final Report) will be submitted.

### **System Errata Documentation**

Any known bugs, and system errata will be documented for use by future students.

# Budget

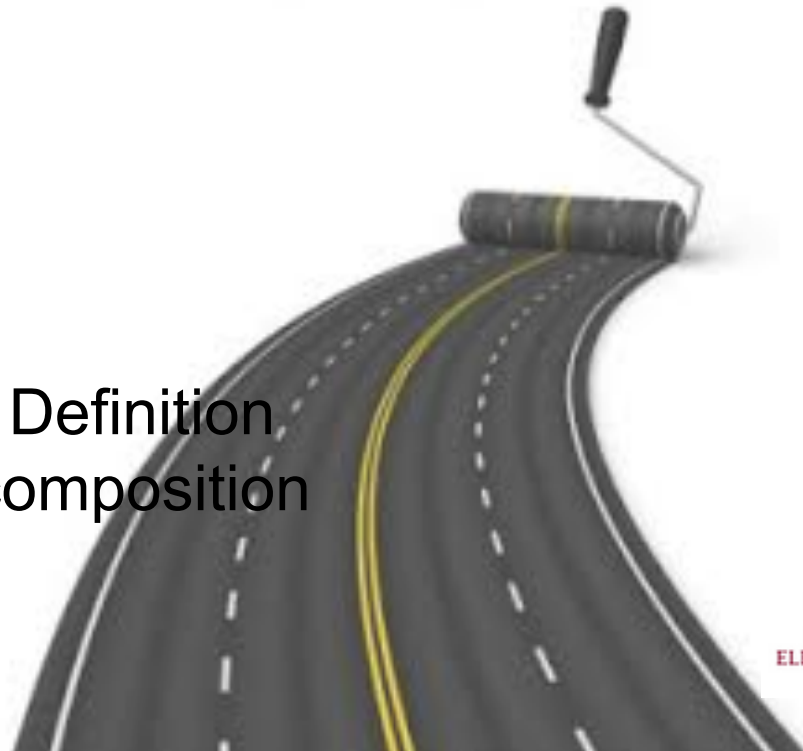
Item/Group	Total
<b>SCADA</b>	
Embedded Computer System	270
Dashboard LCD display and controller	80
Wireless Radio	50
Slave Sensor Micro Controller Hardware	100
Debugger	80
Programmer	10
Total	590
Budget:	715

# Other Items Which Will Be Addressed - Update

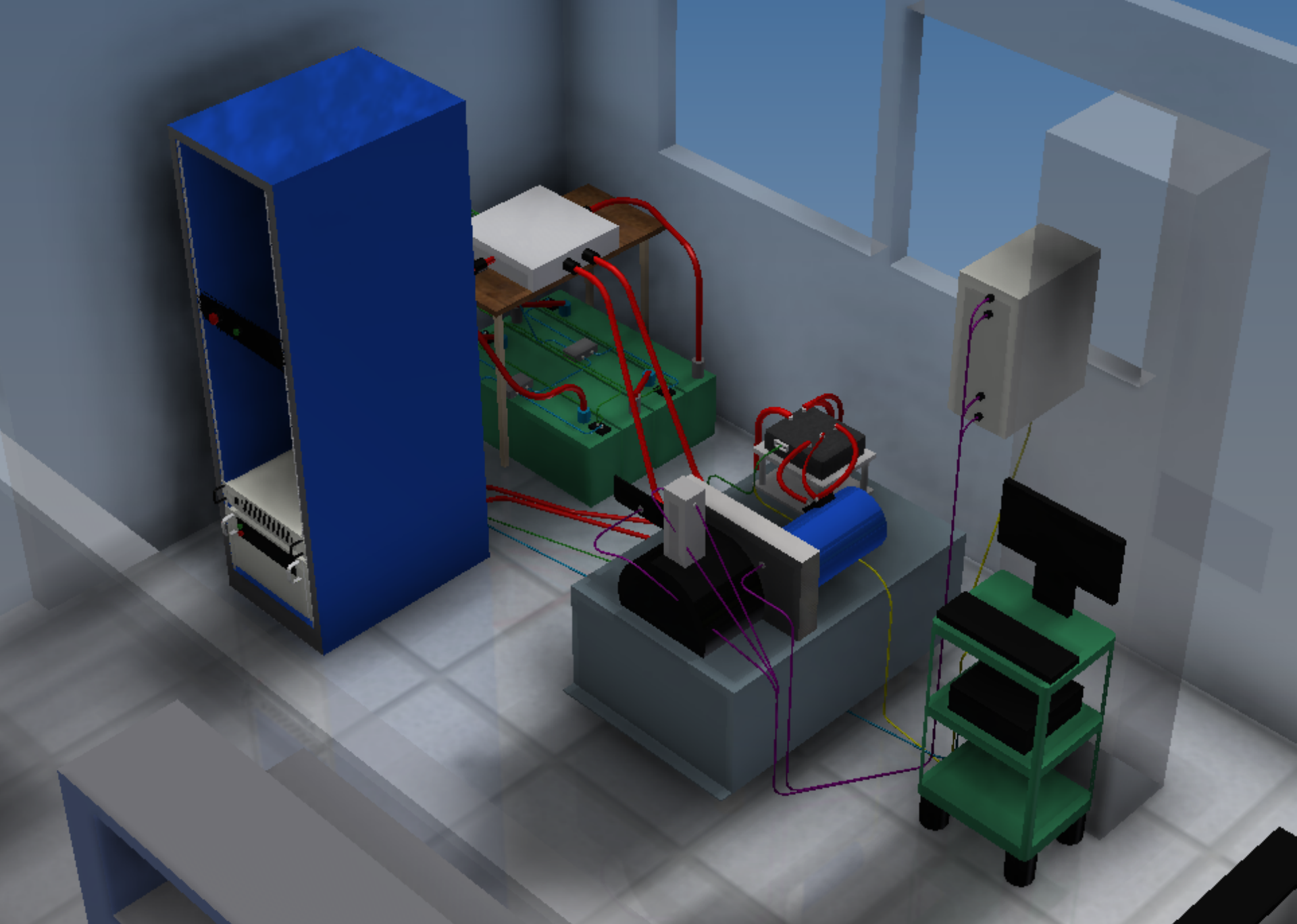
- Brownouts/System Power Loss Handler needs to be designed
- To go along with this, crash handling system needs to be designed
- Watchdog timer exists and has been tested on main board, will be integrated into code
- Timing diagrams will be made and inspected a little later on in the database process
- UML diagrams will be auto-generated using code and comments with Doxygen; any missing or incorrect diagrams will be created or modified manually
- Other documentation such as API docs, code comments following language specific specifications, etc. will either be created by hand or generated with Doxygen

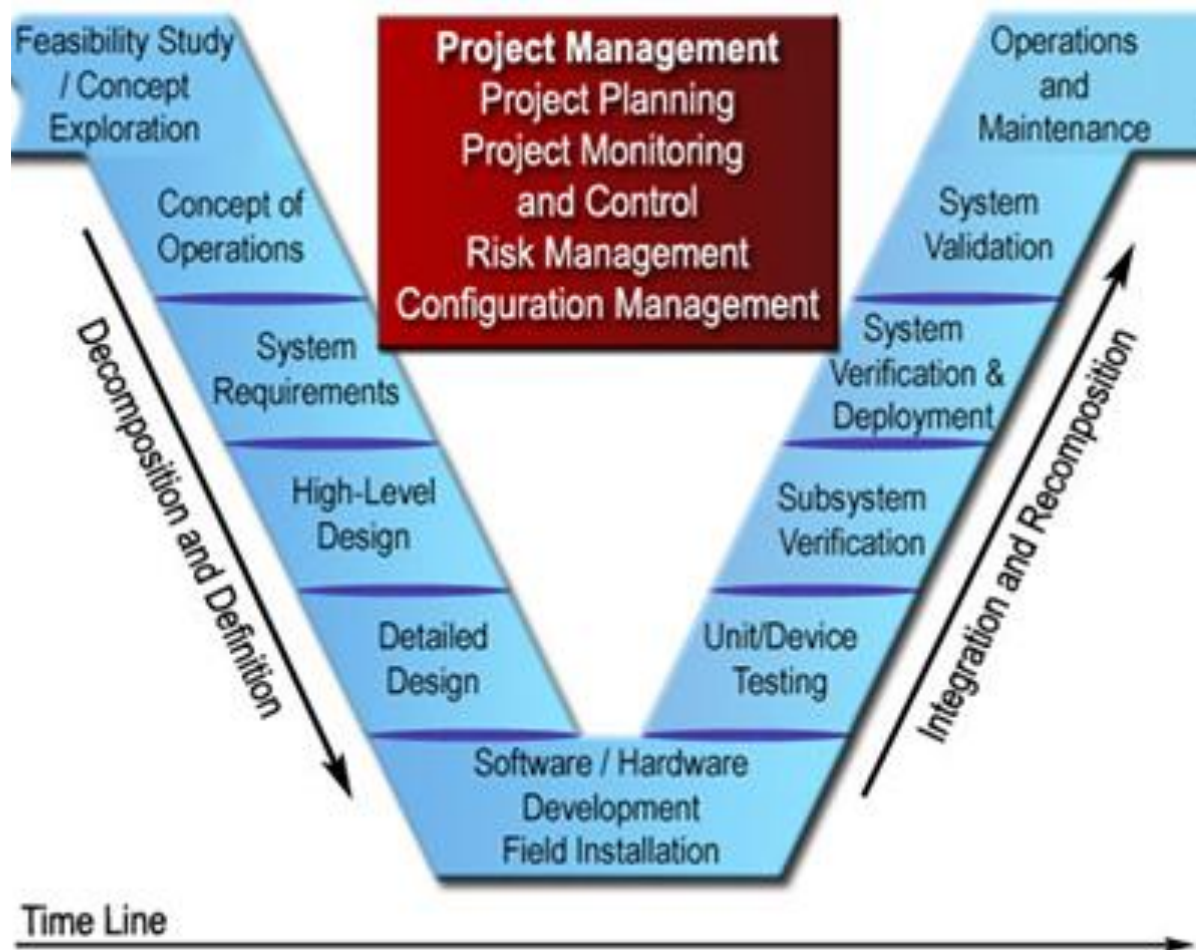
# Roadmap

10. Meet the Afternoon Teams
11. Interface Control and Assemblies Review
12. Vehicle Supervisory Control and Data Acquisition (VSCADA)
  - a. Daemon
  - b. Interfacing
  - c. User Applications
  - d. Data Storage
13. **Dynamometer (DYNO)**
  - a. Decomposition and Definition
  - b. Integration and Recomposition





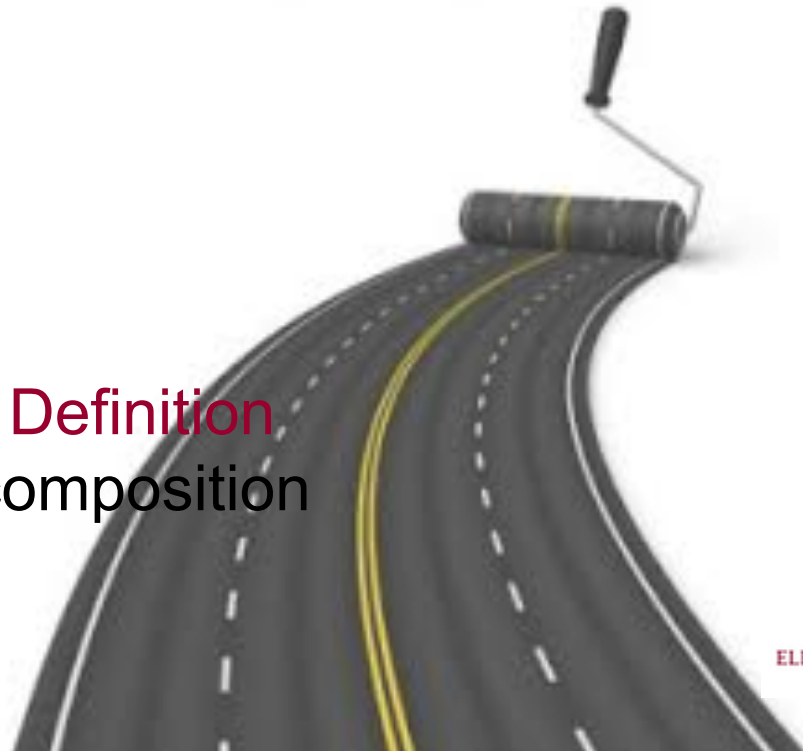




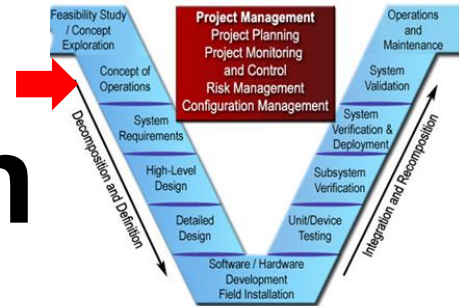
<http://ops.fhwa.dot.gov/publications/seitsguide/images/image068.jpg>

# Roadmap

10. Meet the Afternoon Teams
11. Interface Control and Assemblies Review
12. Vehicle Supervisory Control and Data Acquisition (VSCADA)
  - a. Daemon
  - b. Interfacing
  - c. User Applications
  - d. Data Storage
13. Dynamometer (DYNO)
  - a. Decomposition and Definition
  - b. Integration and Recomposition

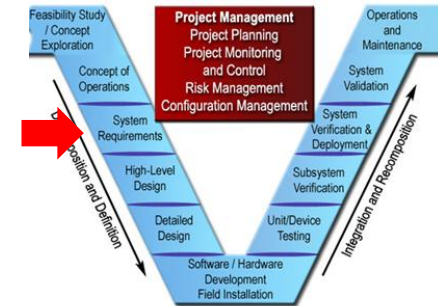


# Concepts of Operation



- Generate a torque curve
- Develop a software simulation of the car
- Develop a hardware simulation of the car
- Determine the car gear ratio

# System Requirements



- Motor/Dyno Selection
- Motor Controller
- Software
  - Data Acquisition
  - Throttle Control
- Interfaces
  - VSCADA
  - GLV
  - TSV
- Safety

# Motor Selection

- HPEVS AC 50-27.28

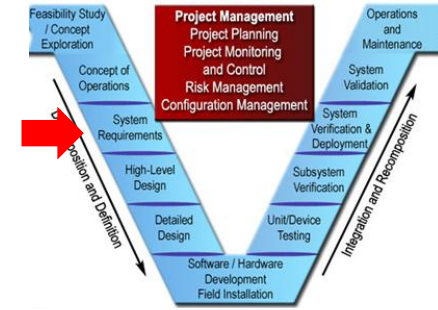
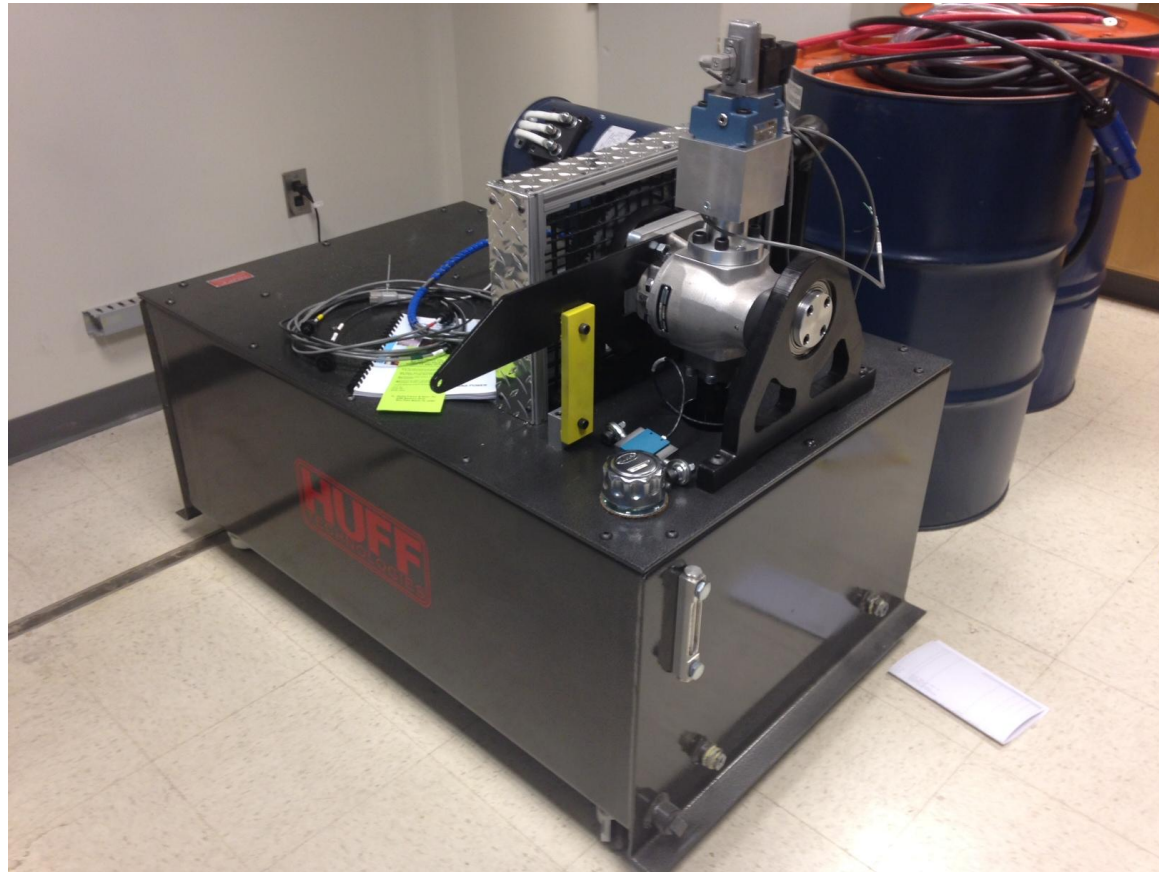
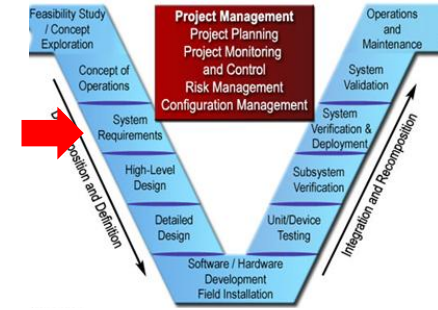


Photo: HPEVS



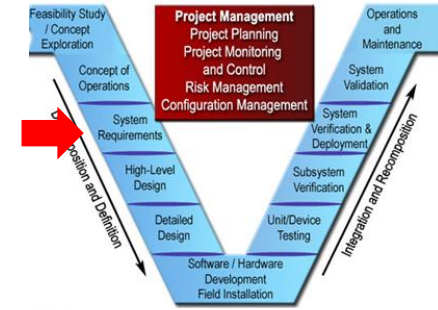
# Dynamometer Selection

- Huff HTH-100



# Motor Controller

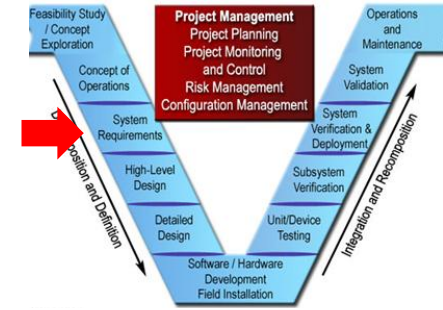
- Curtis 1238R-7601





# Interfaces

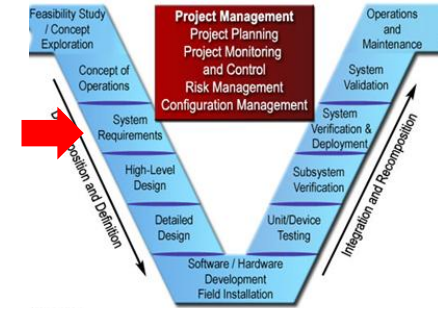
- VSCADA - Interface for data acquisition and throttle control
- GLV - Interface for power and data transmission
- TSV - Interface to power supply



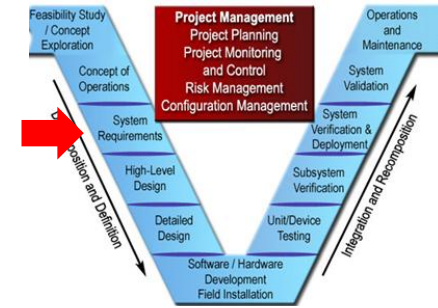
# Software

## VSCADA - Dyno

- Data Acquisition
  - RPM
  - Torque
  - Temp - Motor and Controller
  - RMS Current
  - Voltage
- Throttle Control



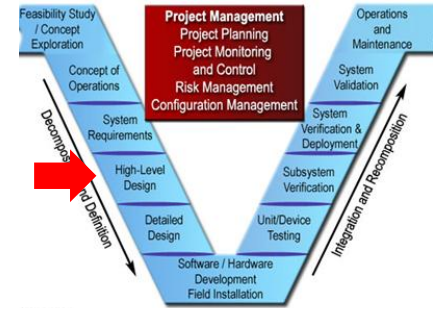
# Safety



- Emergency Shutoff
  - Must be have an emergency stop
  - Must be shut down when GLV is down
- Oil Temperature Shutoff
  - Must shut down when temperature limit is exceeded
- Galvanic Isolation
  - Must separate high and low voltage subsystems
- Motor Controller Contact Shield
  - Prevents accidental contact with terminals

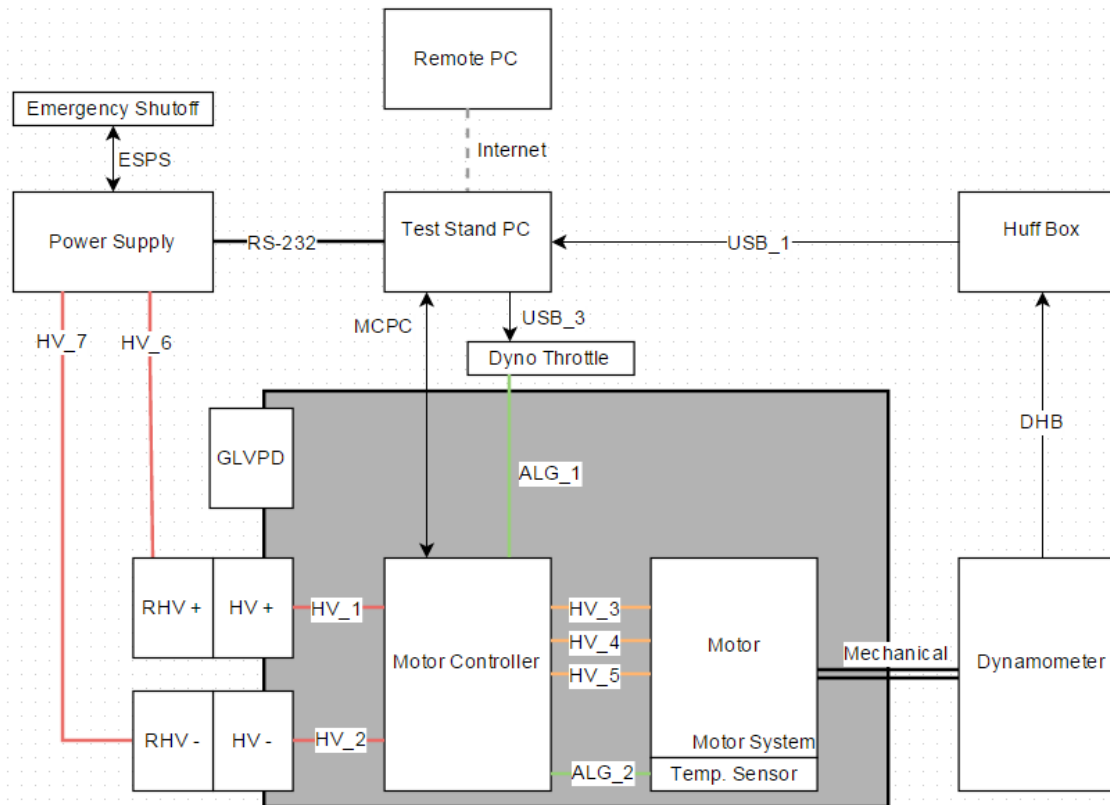
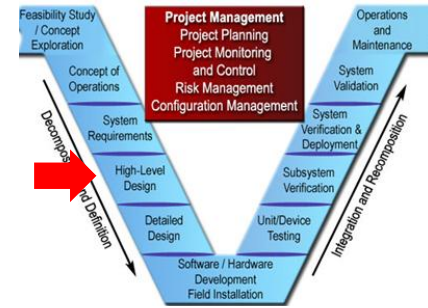
# High-Level Design

- ICD Layouts
- HUFF - VSCADA Interface
- Motor Controller
  - Cooling
  - Safety
- Safety Shutoff
- Throttle
- Galvanic Isolation

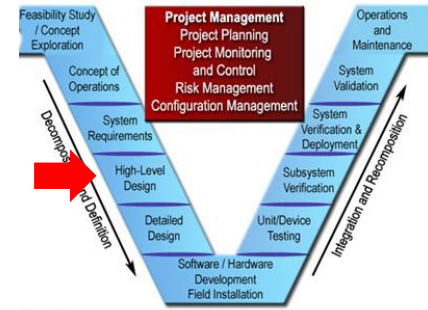


# ICD Layout

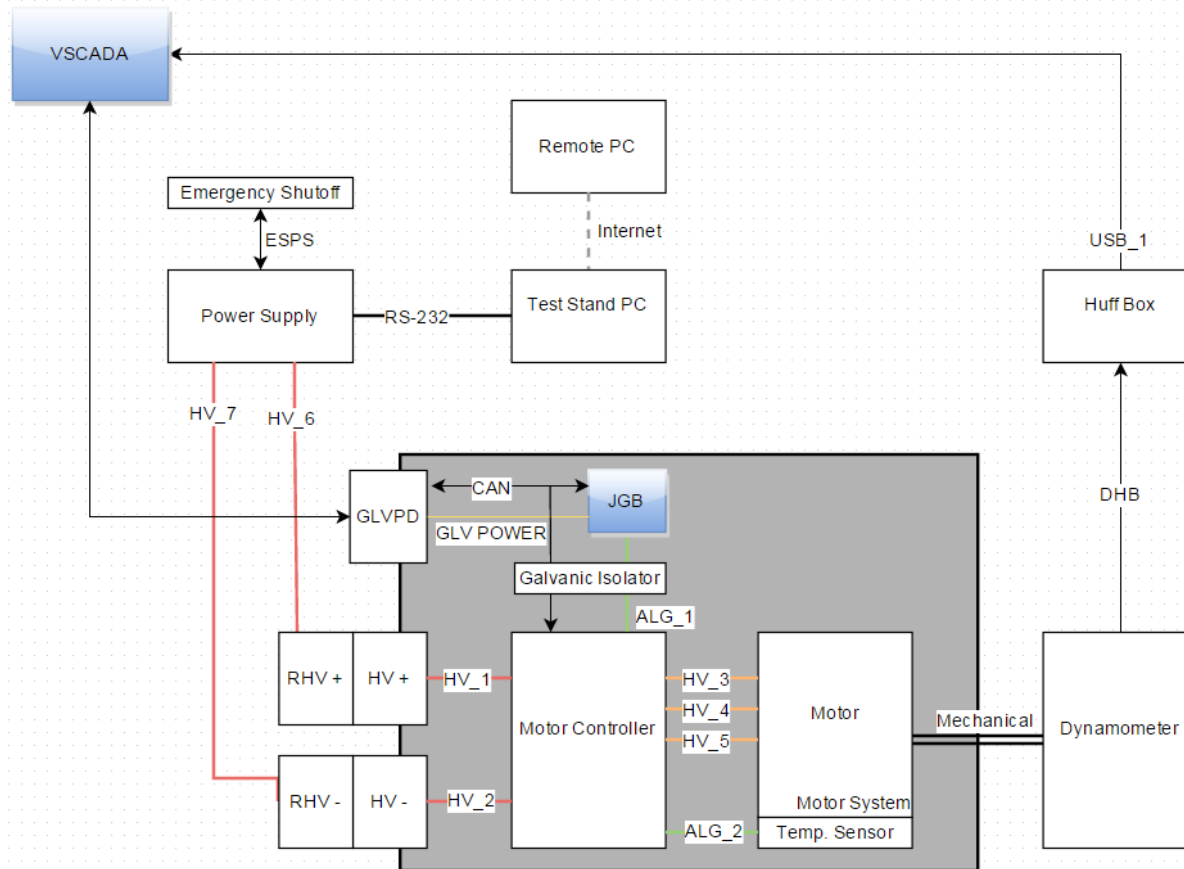
- Two configurations
  - Dyno Testing Configuration:



# ICD Layout

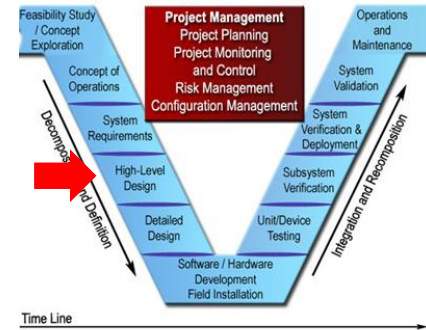


- Two configurations
  - Integrated Design Configuration:

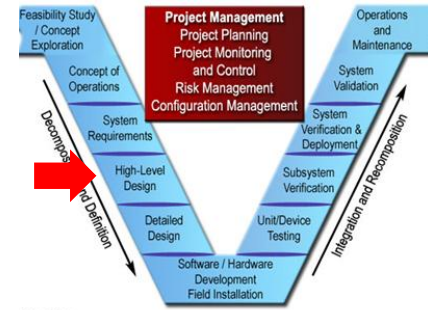


# Huff - VSCADA interface

- USB interface
- Utilizes serial communication
- Based on a call and response system
- Used to acquire data and set values
- Protocol is defined by the chip on data acquisition board



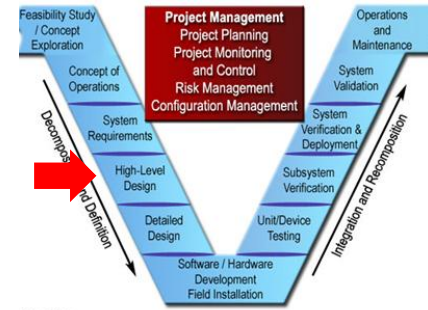
# Motor Controller Cooling



- Must regulate MC temperature
  - Storage ambient temperature range:-40°C to 95°C
  - Operating ambient temperature range:-40°C to 50°C
  - Internal heatsink operating temperature range:-40°C to 95°C
- Utilize a Water Cooling system
  - Pump→ MC→ Radiator→ Pump
  - Mounted Cooling Housing
  - Effectiveness to be determined upon delivery of parts



# Motor Controller Safety

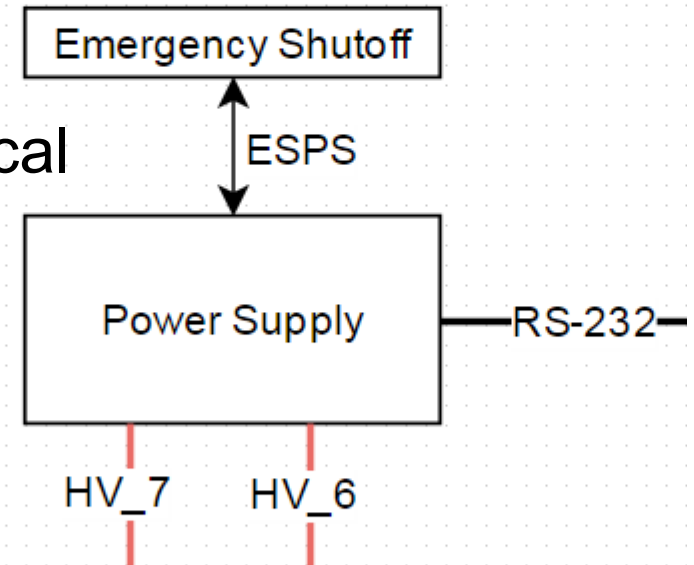


- Must prevent conductive injury from MC ports
  - High Voltage
- Cover all electrical hazards to prevent accidental contact
  - Use non-conductive plastic cover

# Safety Shutoff



- Requirements -
  - Must include emergency stop
  - Must include temperature shutoff
- Design -
  - Use the power supply control inputs. These control mechanical contactors.

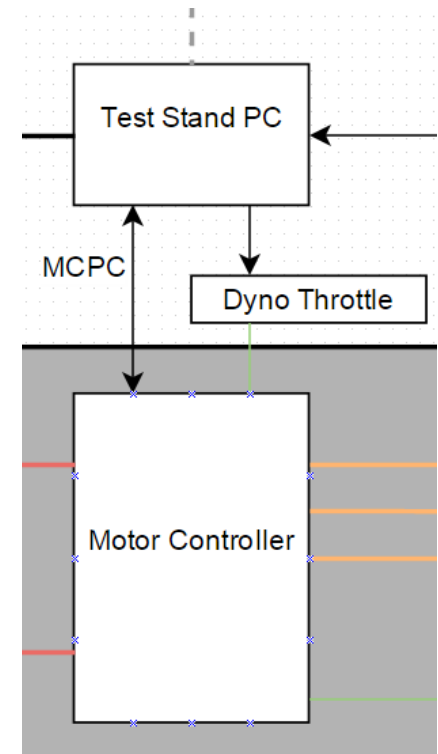
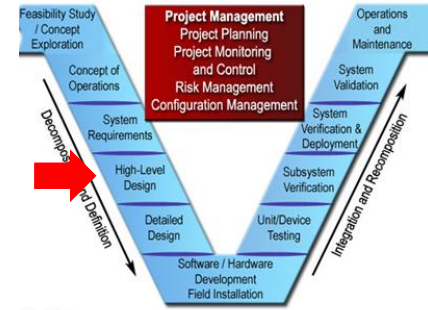


# Conditions to trip the Safety Loop

- Overcurrent
- Current Sensor Fault
- Precharge Fail
- Severe Undertemp
- Severe Overtemp
- Severe Overvoltage
- Main Open/Short
- Main Contactor Welded
- Main Contactor Did Not Close
- EEPROM Failure
- Parameter Change Fault
- Motor Open
  - U, V, or W not connected
- VCL Run Time Error
- OS General Fault
- Motor Characterization Fault
- Motor Type Fault
- VCL/OS Mismatch
- Illegal Model Number

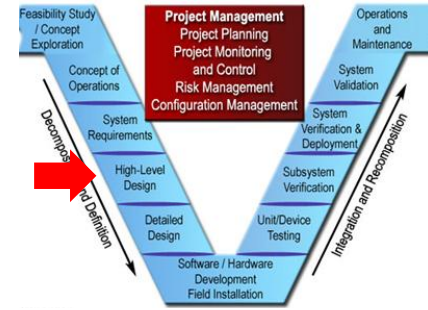
# Throttle

- Must control the throttle input of the MC
  - Throttle input is 0 to +5 volts
  - VSCADA must be connected
  - Must be scriptable for testing
- Use two systems:
  - Use a VSCADA CANbus node with an analog output for the integrated system
  - Use an Arduino connected over USB to control an analog output

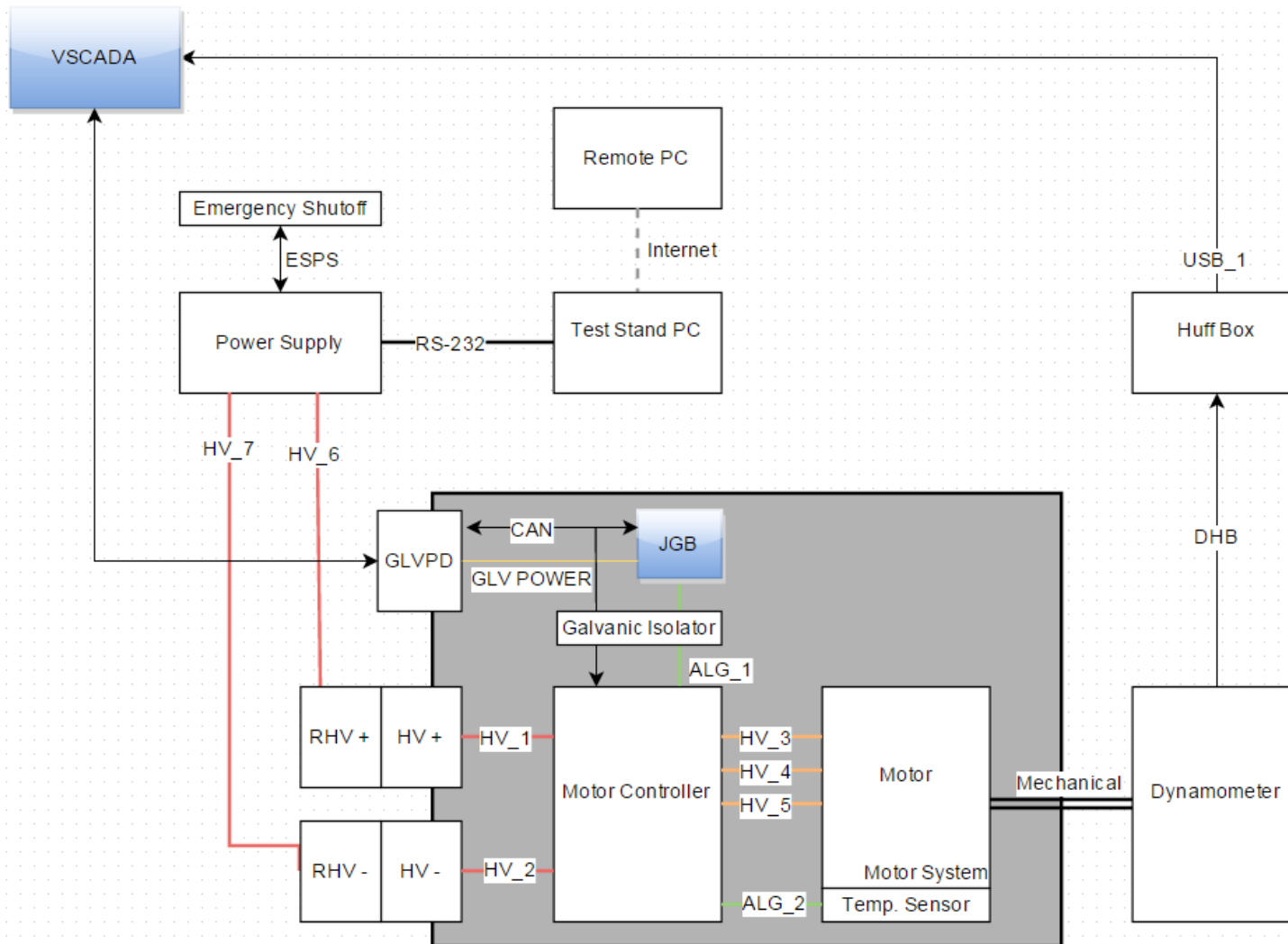
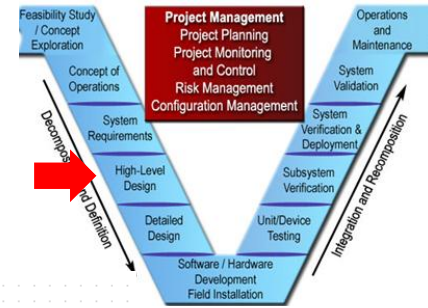


# Galvanic Isolation

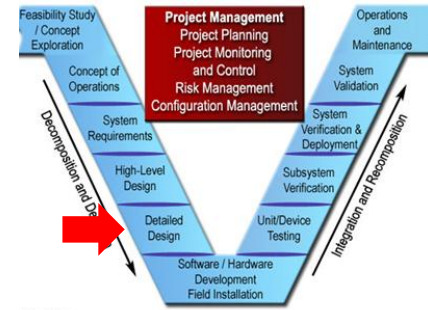
- High/Low voltage CAN must be separated
- High/Low Voltage Throttle must be separated



# Layout Review:

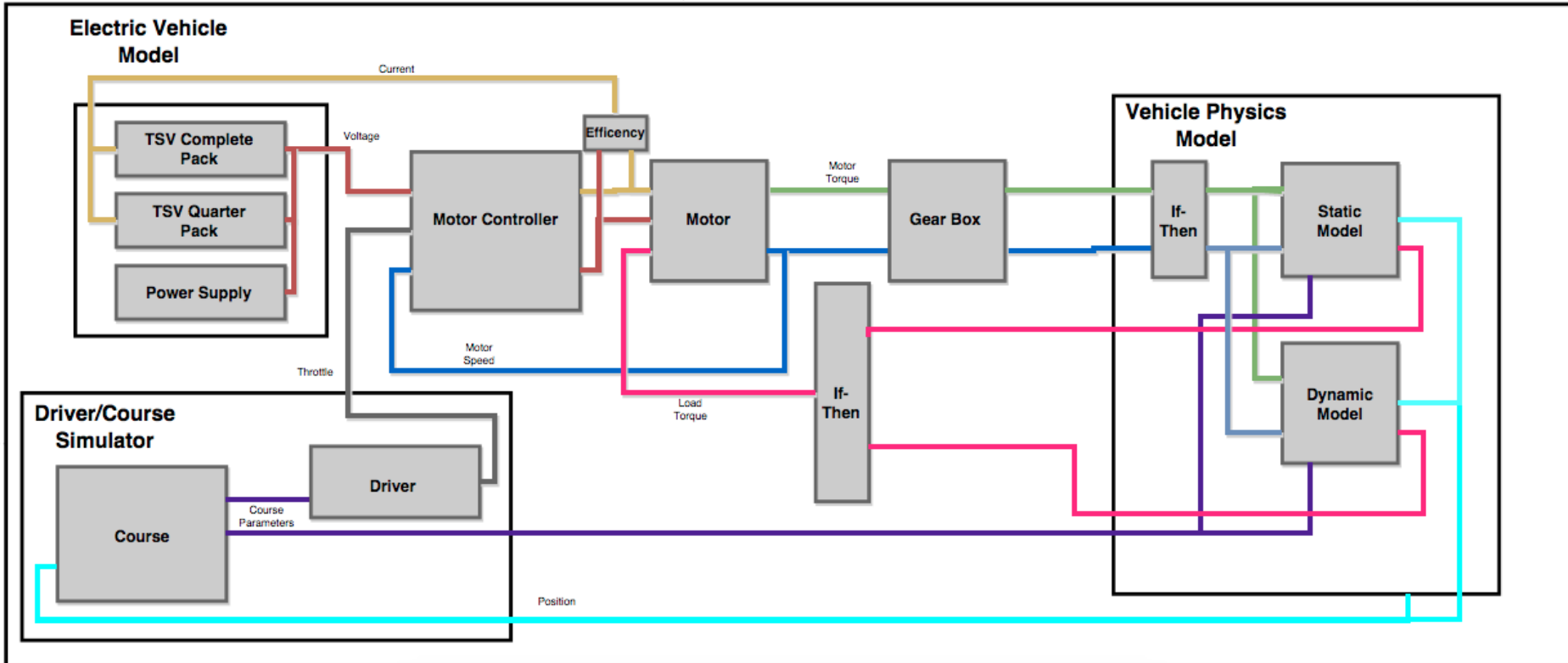


# Detailed Design



- Simulations
  - Motor
  - Car
  - Track
- Safety
  - Independent shutoff
  - Insulating covers
- Throttle
  - Independant solution
  - VSCADA solution
- Motor Controller
  - Isolation
  - Parameters
  - Wiring Diagram
  - Cooling
  - Safety
- Room Wiring
  - Testing config
  - Integrated config

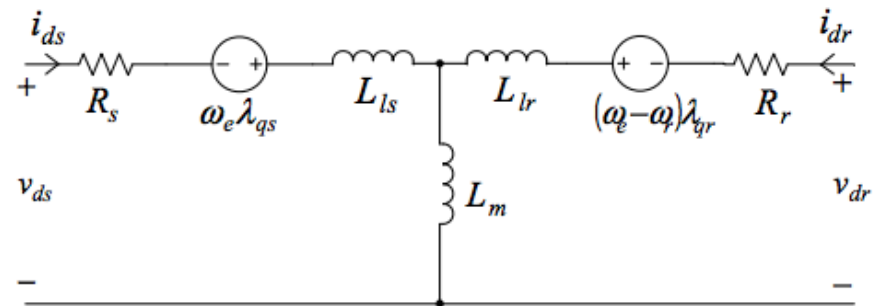
# Simulation Block Diagram



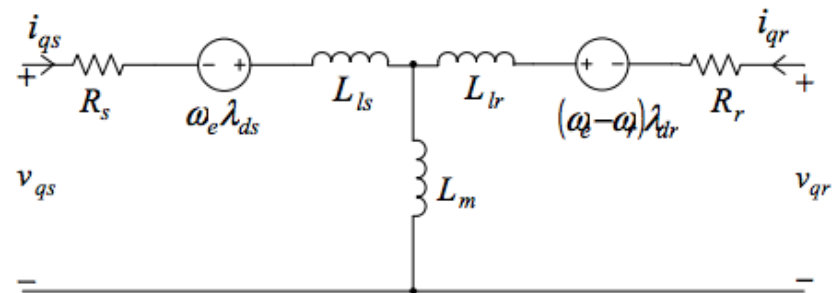


# Simulations - Motor

- dq0 Circuit Equivalent Circuit
- Transform 3-phase circuit into 2 phases for easier analysis
- MATLAB
- Simulink

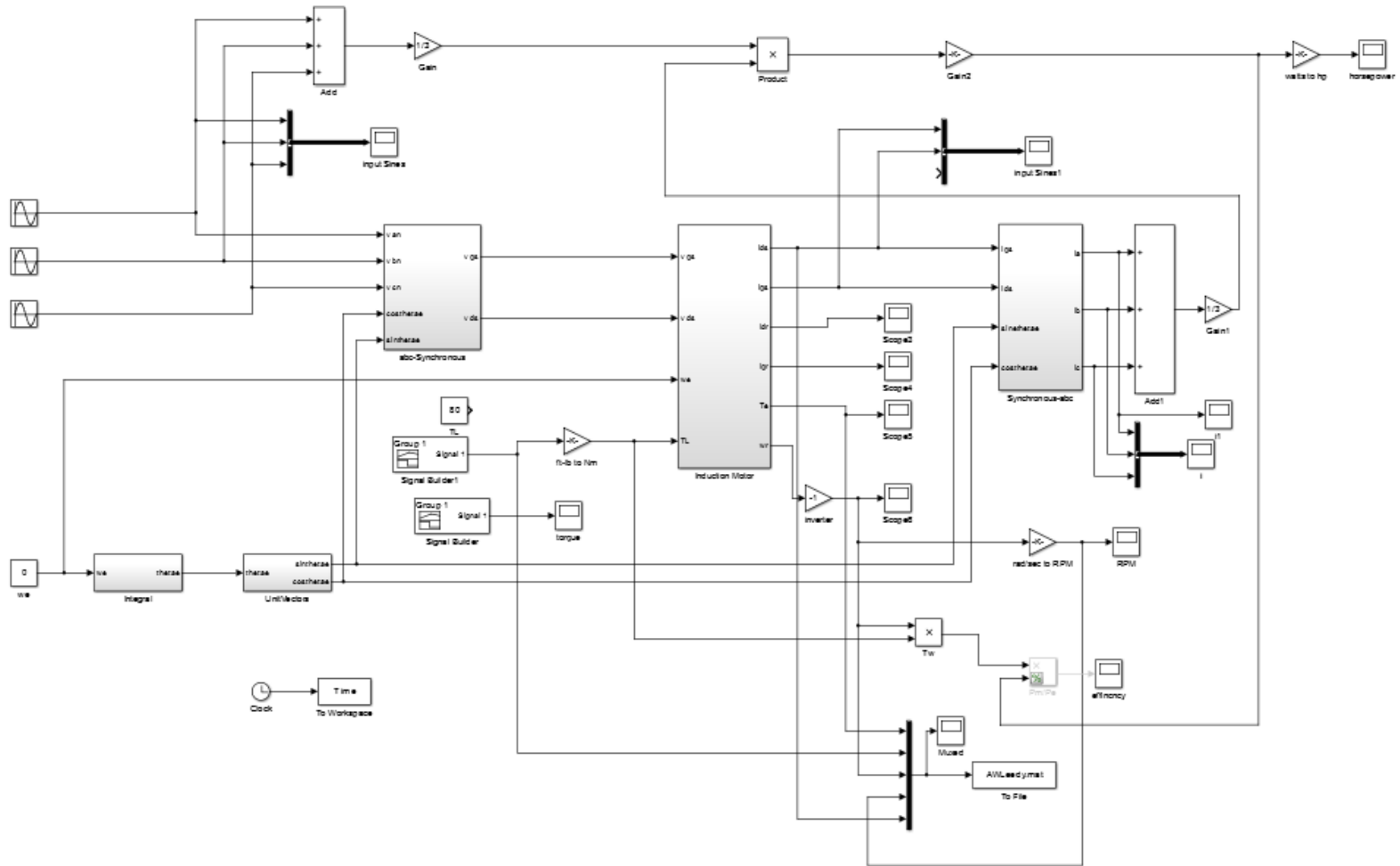


d-axis equivalent circuit

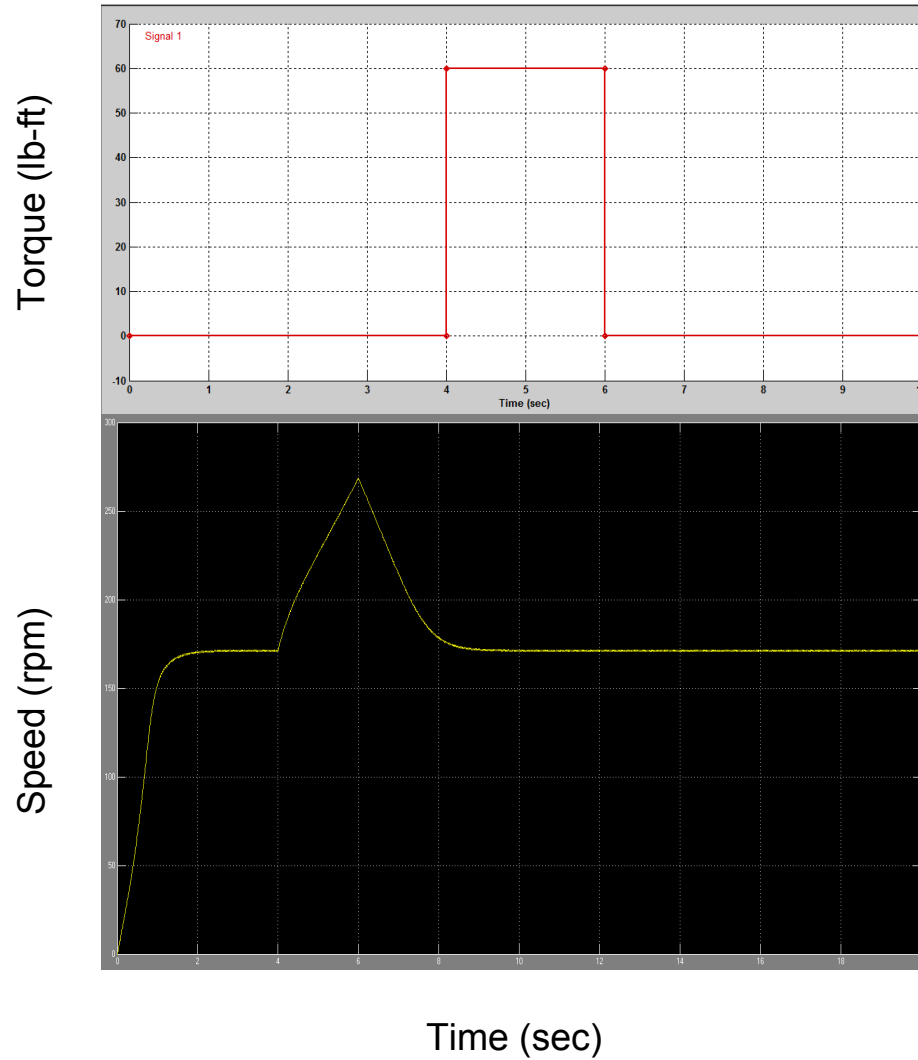


q-axis equivalent circuit

# AC Induction Motor Model: dq0 Transformation



# Simulation - Torque Response



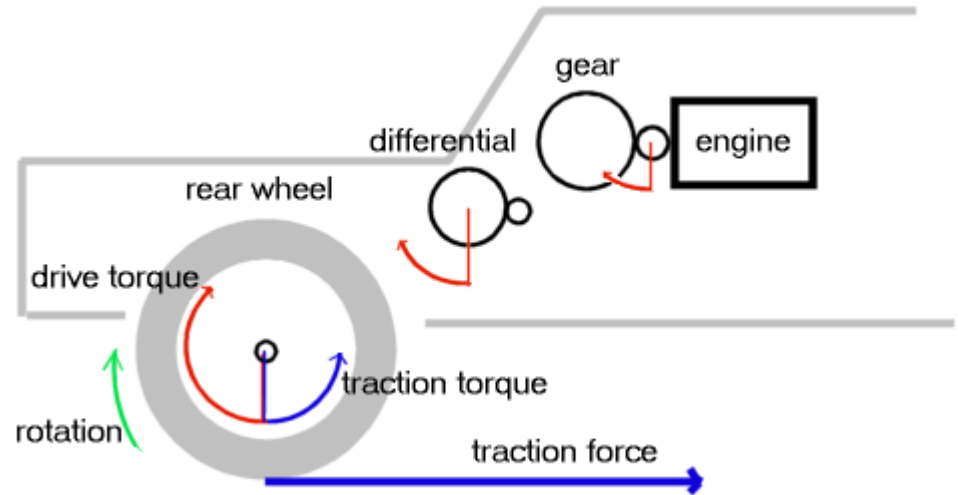
# Simulations - Car

## Static

- Static Friction

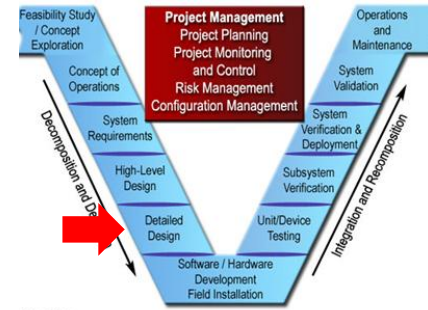
## Dynamic

- Air Drag
- Dynamic Friction
- Turn Radius
- Momentum



<http://www.asawicki.info/Mirror/>

# Simulations - Track



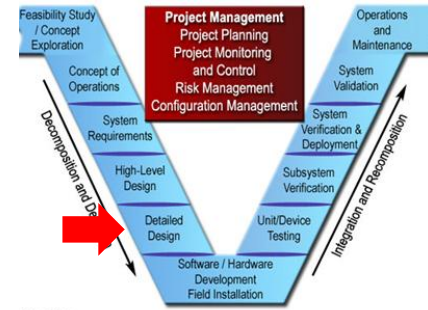
Course Layout using arrays to match specifications of courses in SAE EV rules

- Position on the track
- Angle of incline and decline
- Angle of curves
- Embankments
- Throttle input

Course Parameters

- Coefficient of friction (dry, damp, wet)

# Safety - Independent Shutoff



- Use the input control lines
  - Directly linked to mechanical contactors
- Interface is a 37-pin D-Sub connector
  - Need only the start and stop inputs

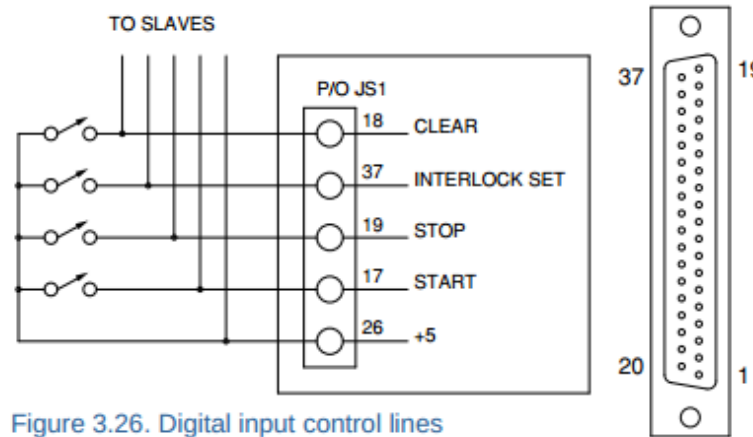
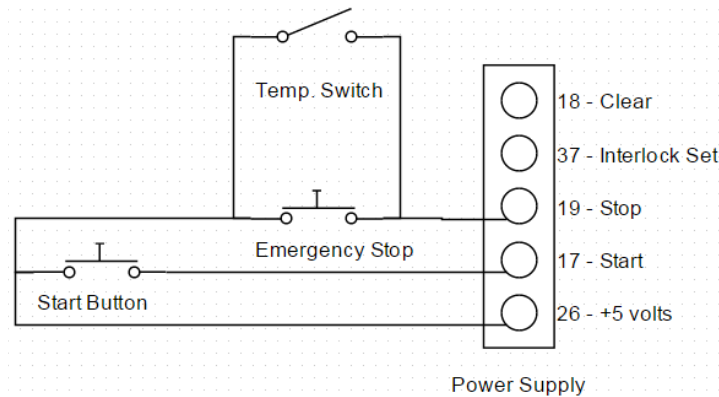
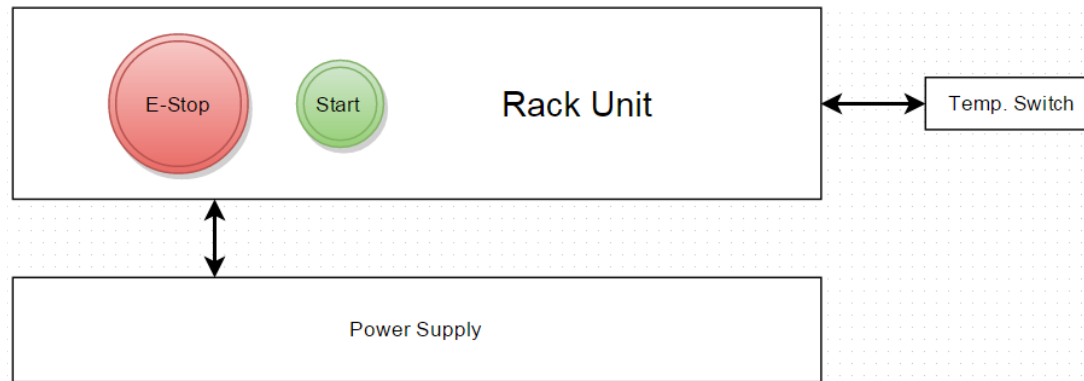
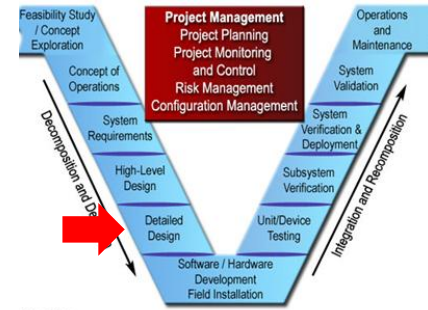


Figure 3.26. Digital input control lines

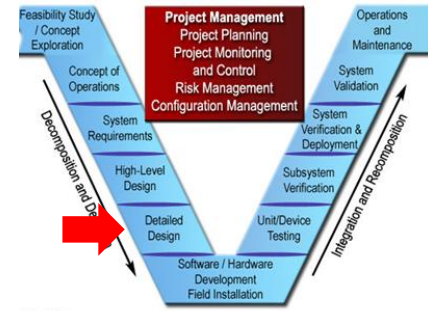
# Safety - Independent Shutoff

- Solution: simple rack mounted unit

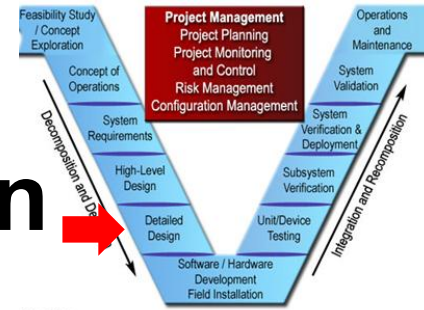


# Safety - UPDATE

- The extra panel will be removed in integrated system
  - The GLV emergency stop will be integrated into the power supply

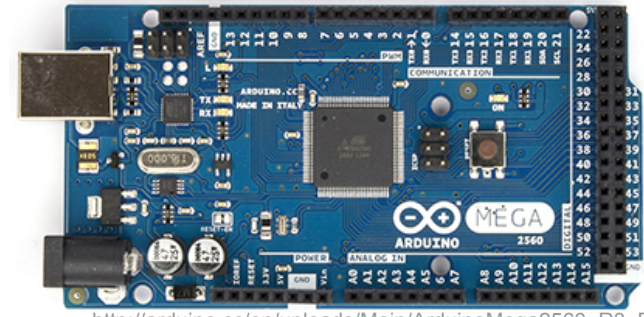




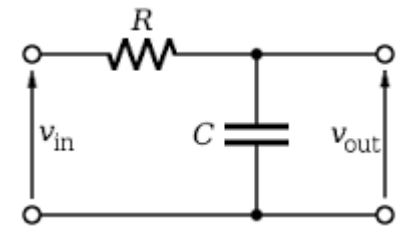


# Throttle - Independent Solution

- Need to control throttle from a computer
  - Arduino with USB connection
- No analog outputs
  - Low pass filter on a PWM
- Scripting
  - Write values in a python script

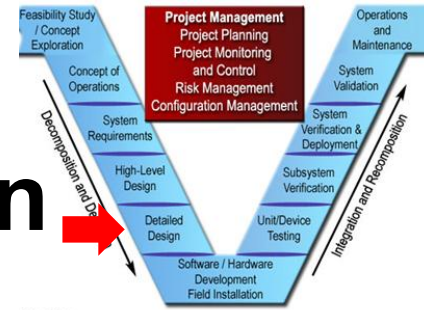


[http://arduino.cc/en/uploads/Main/ArduinoMega2560\\_R3\\_Fronte.jpg](http://arduino.cc/en/uploads/Main/ArduinoMega2560_R3_Fronte.jpg)

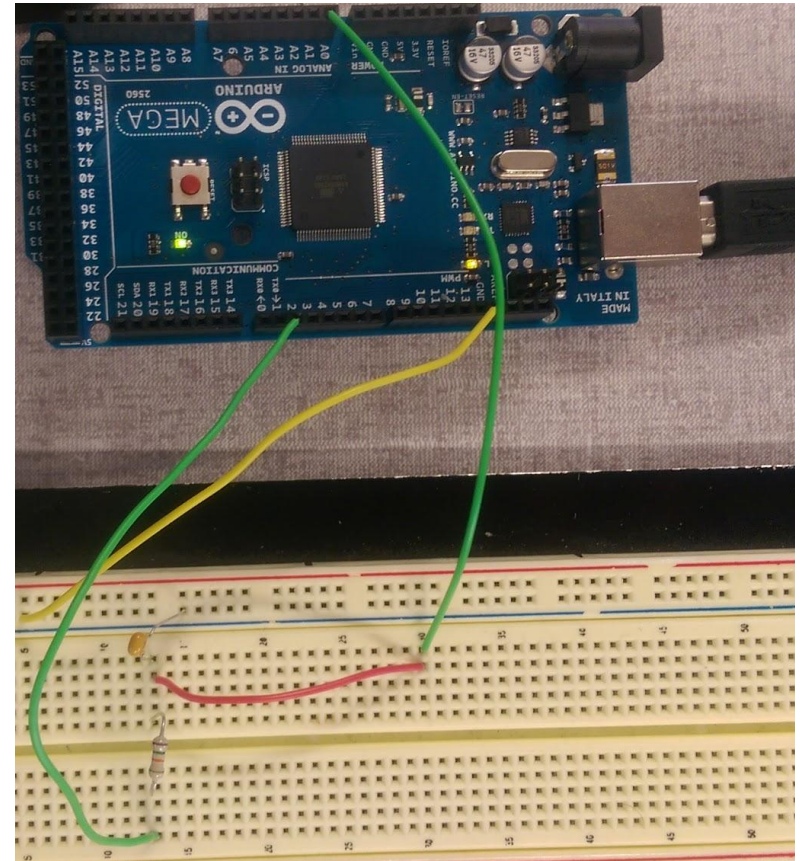
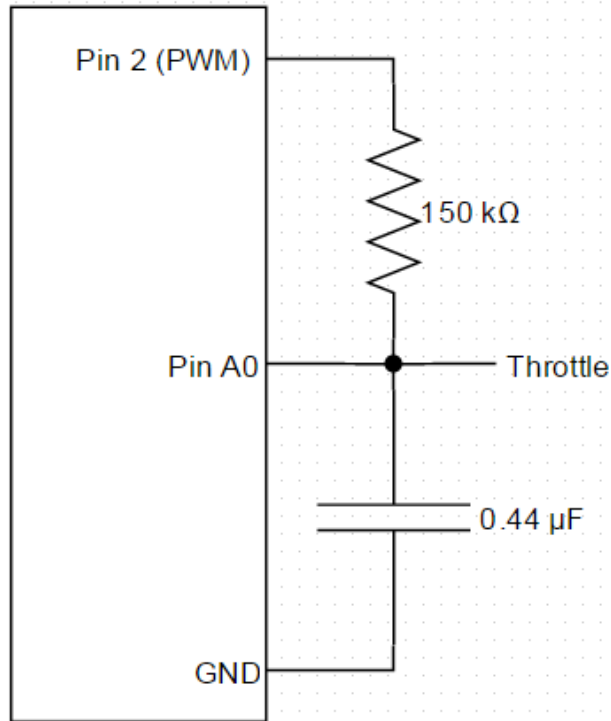


[https://www.python.org/static/community\\_logos/python-logo.png](https://www.python.org/static/community_logos/python-logo.png)

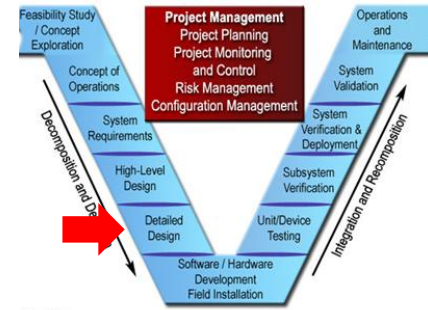
# Throttle - Independent Solution



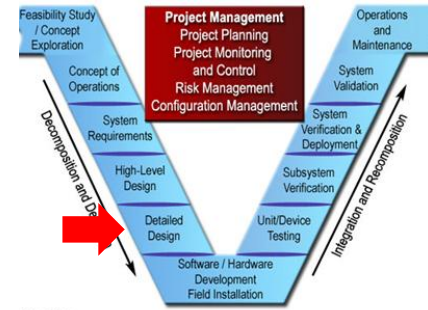
Python script writes PWM values, which are filtered to analog voltages



# Throttle - VSCADA Solution

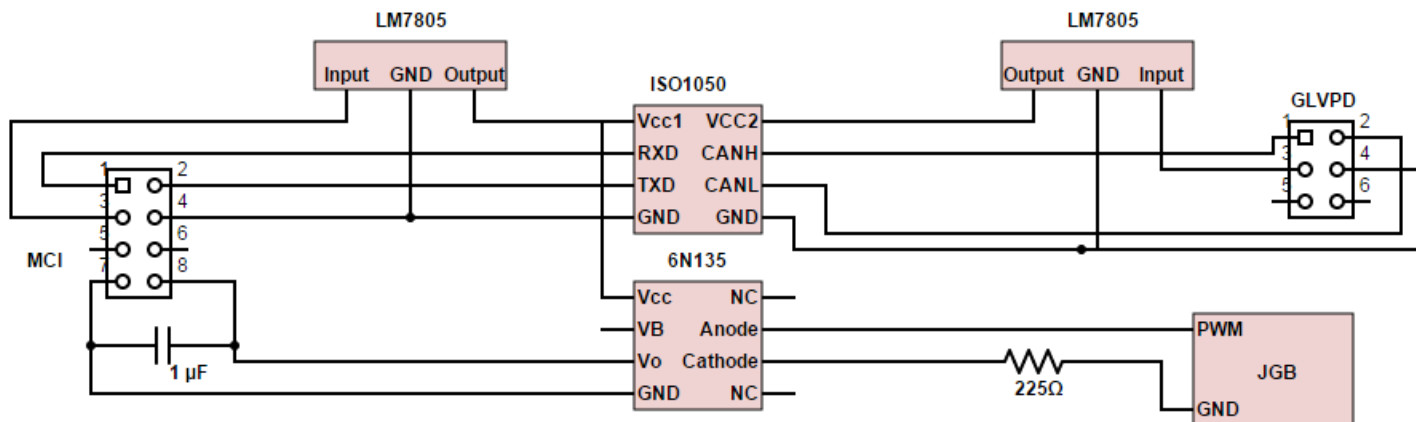


- USB connection from VSCADA to Huff Box
- Serial communication
- Call and response
- Protocol dictated by DAQ chip
- Generate PWM signal
  - Relates RPM to voltage
  - PWM is low pass filtered

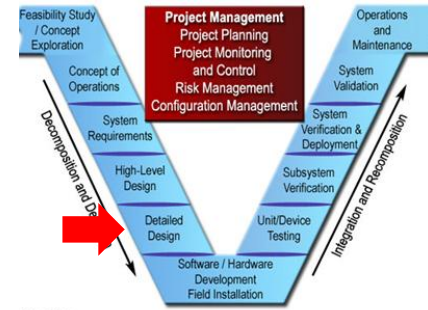


# Motor controller - Isolation

- Isolates motor controller from GLV systems
  - Isolate CANbus
    - Using TI ISO1050DUBR
    - Voltage step down using LM7805
  - Isolate throttle
    - Using 6N135 optocoupler
    - low pass filter PWM signal



# Motor Controller - Parameters

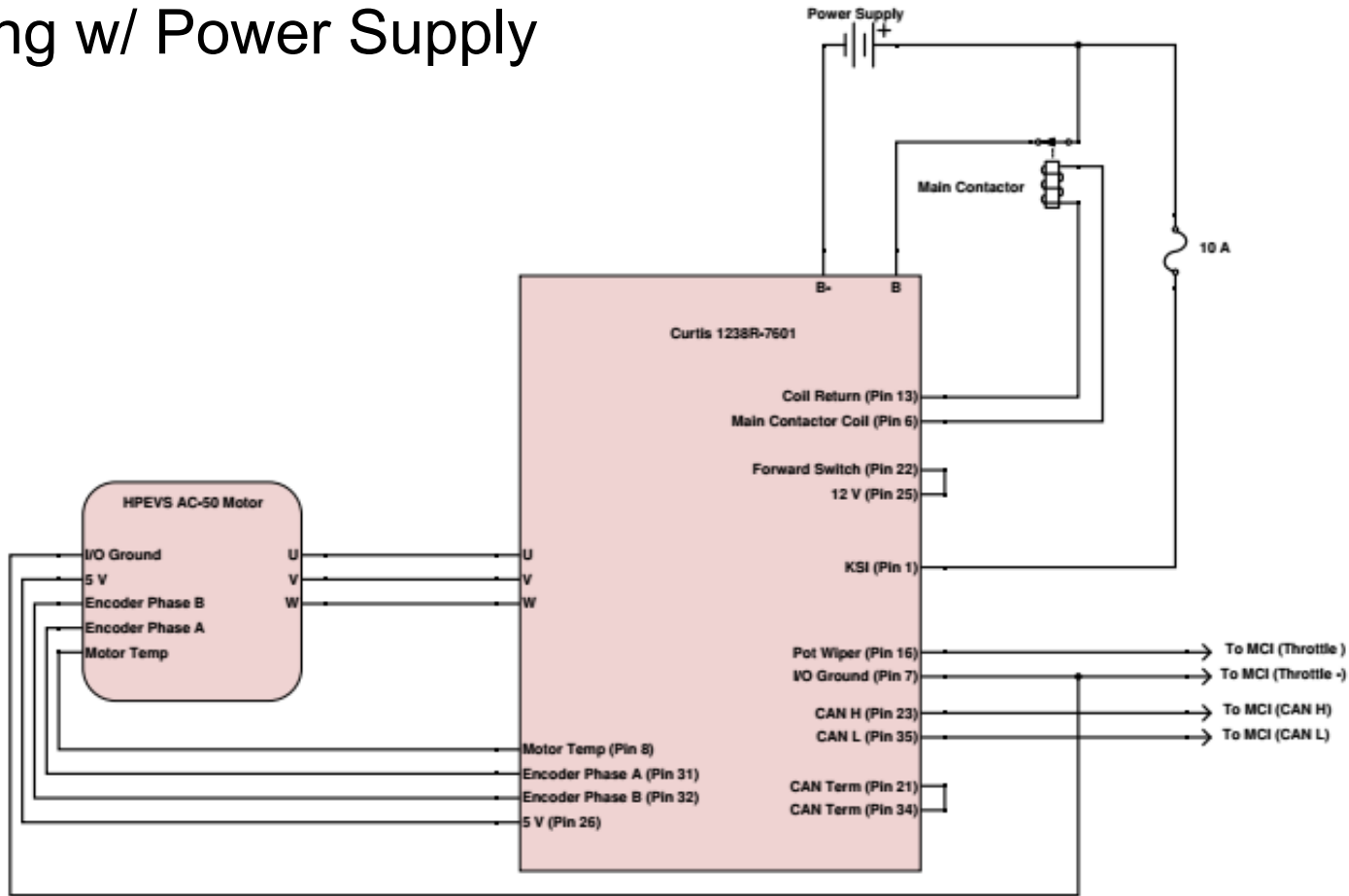
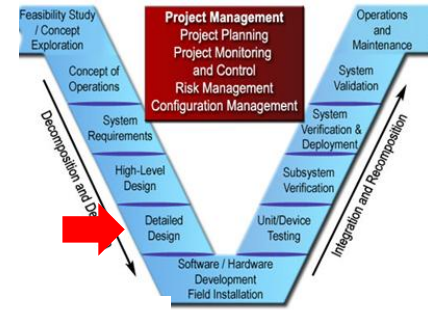


Program		
User Settings		
Speed Settings		
Forward Speed	6500	rpm
Reverse Speed	6500	rpm
Econo Speed	6500	rpm
Accel Rates		
Normal Accel Rate	0.4	Seconds
Econo Accel Rate	1.0	Seconds
Throttle Settings		
Throttle Type	2	
Deadband	0	Volt
Throttle Max	5	Volt
Mapped Throttle	50	%
Brake Pedal Settings		
Brake Type	0	
Brake Deadband	0.30	Volt
Brake Max	3.50	Volt
Regen Brake Light Th...	50	Ampere
Current Limits		
Normal Neutral Braking	15	%
Econo Neutral Braking	25	%
Shift Neutral Braking	7	%
Normal Drive Current...	100	%
Econo Drive Current ...	60	%
Brake Current Limit	10	%
Idle Setup		
Idle Enable	Off	
Clutch Start Enable	Off	
Idle Speed	600	rpm
Idle Torque	50	%
Creep Torque	0	%

Motor Tuning		
Motor Type	50	
Base Speed	3000	rpm
Field Weakening	50	%
Econo Field Weakening	20	%
Weakening Rate	60	%
Main Contactor		
Main Contactor Voltage	48	Volt
Main Holding %	80	%
Display Menu Items		
Auto Scroll	On	
Scroll Delay Time	10	Seconds
Display SOC	Off	
Display Motor RPM	On	
Display Battery Amps	On	
Display Voltage	On	
Display Motor Temp	On	
Display Controller Temp	On	
Display Minimum Volt...	On	
Display Maximum Cur...	On	
BMS		
BMS Installed	Off	
BMS Address	768	
User Undervoltage	80	%
Low Cell Begin Cutback	2.800	Volt
Low Cell Full Cutback	2.300	Volt
Max Current at Full C..	50	%
Maximum Cell Voltage	3.700	Volt
Low SOC Cutback	20	%
Max Current at Low ...	30	%
Dual Drive		
Dual Drive Mode	Off	
Response Timeout	200	ms

# Motor Controller - Wiring

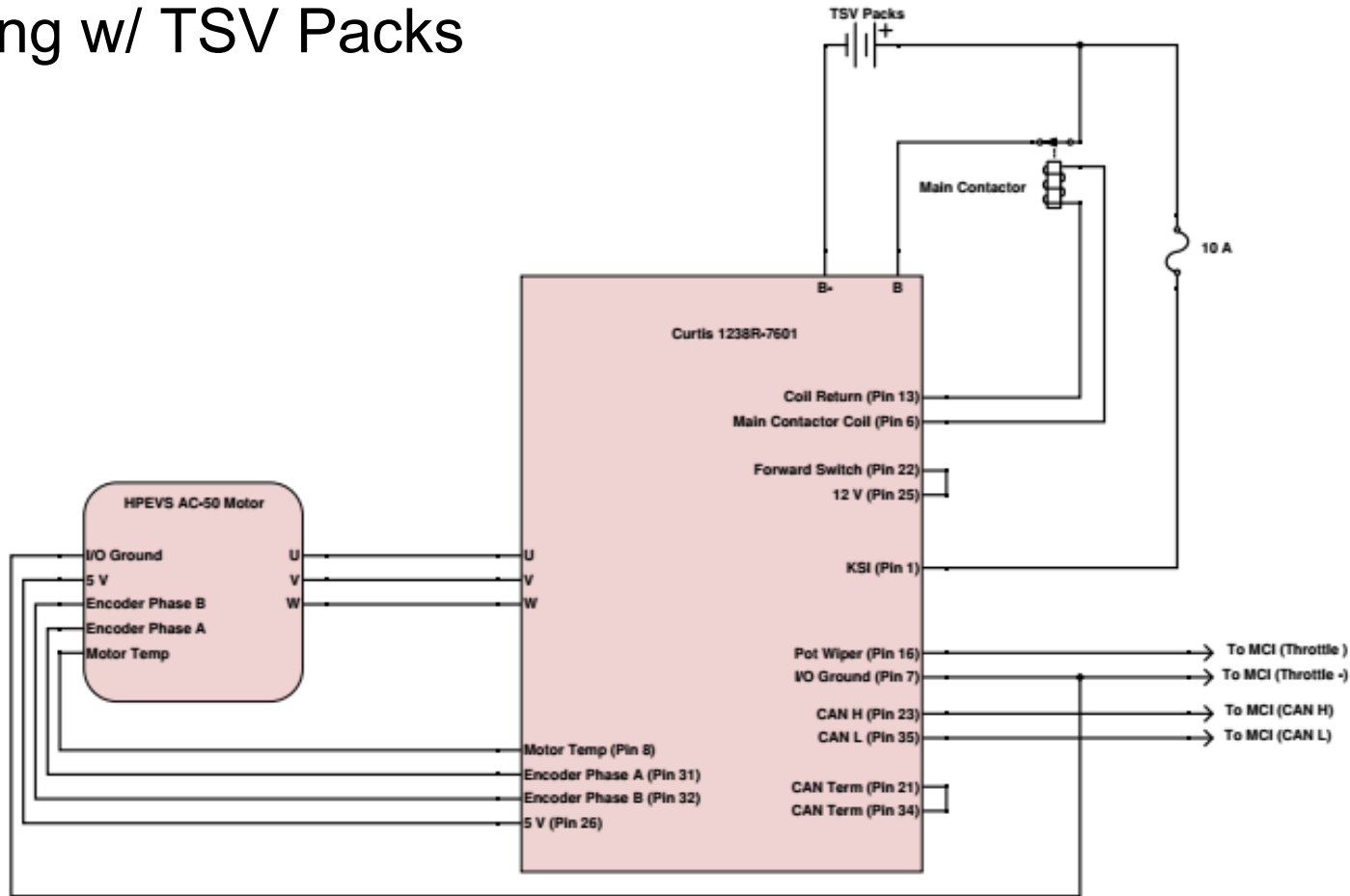
## Testing w/ Power Supply





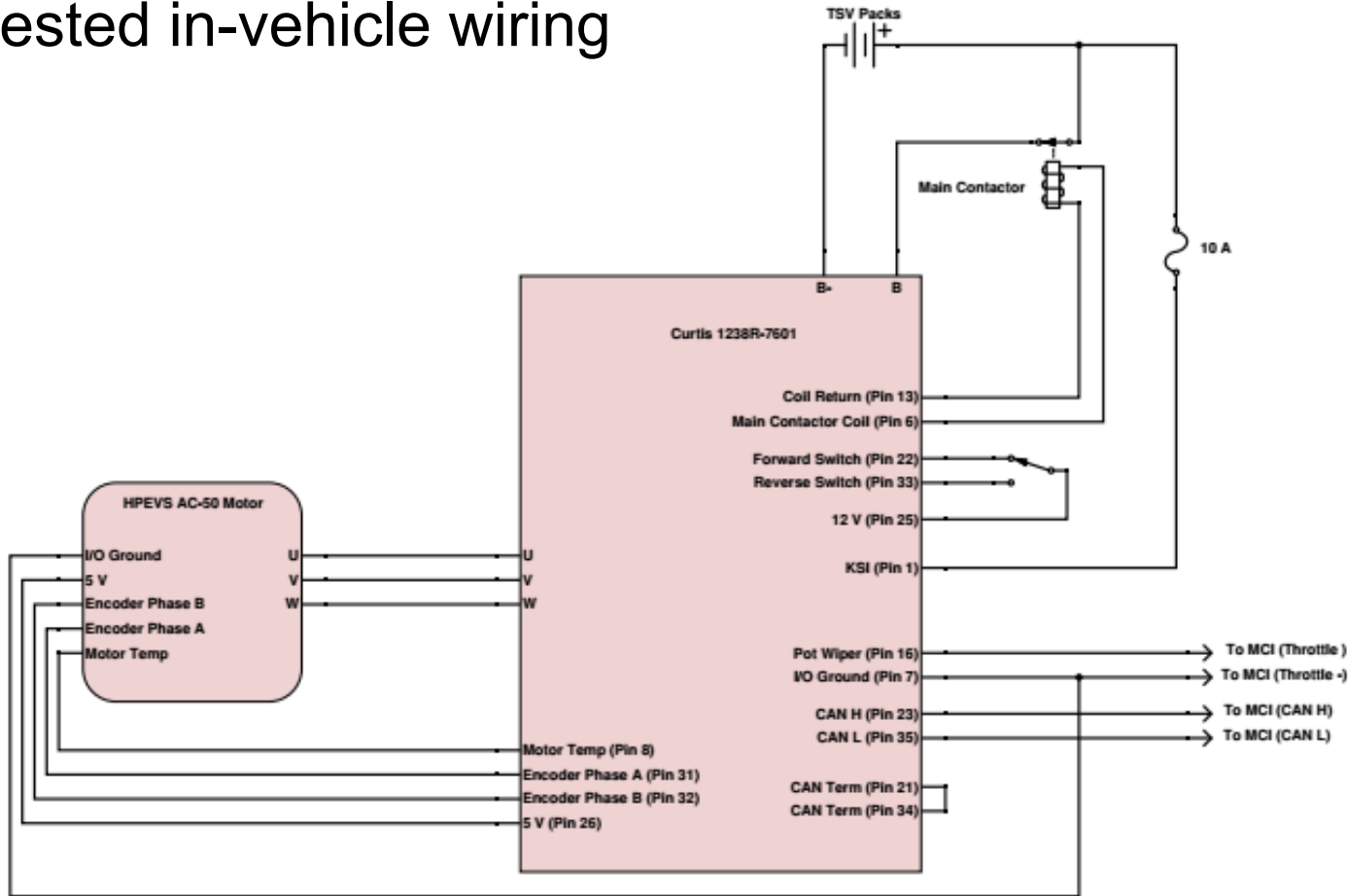
# Motor Controller - Wiring

## Testing w/ TSV Packs



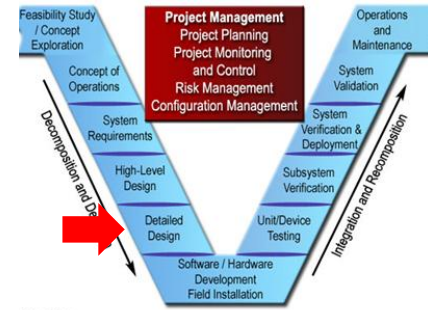
# Motor Controller - Wiring

Suggested in-vehicle wiring

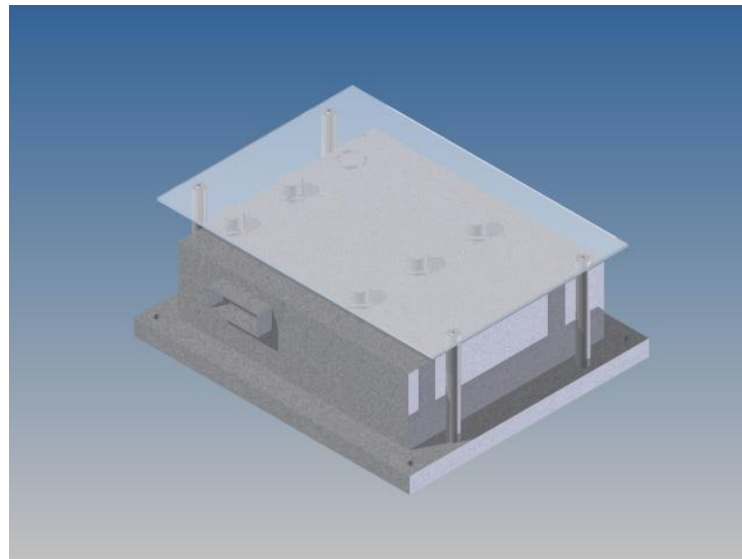




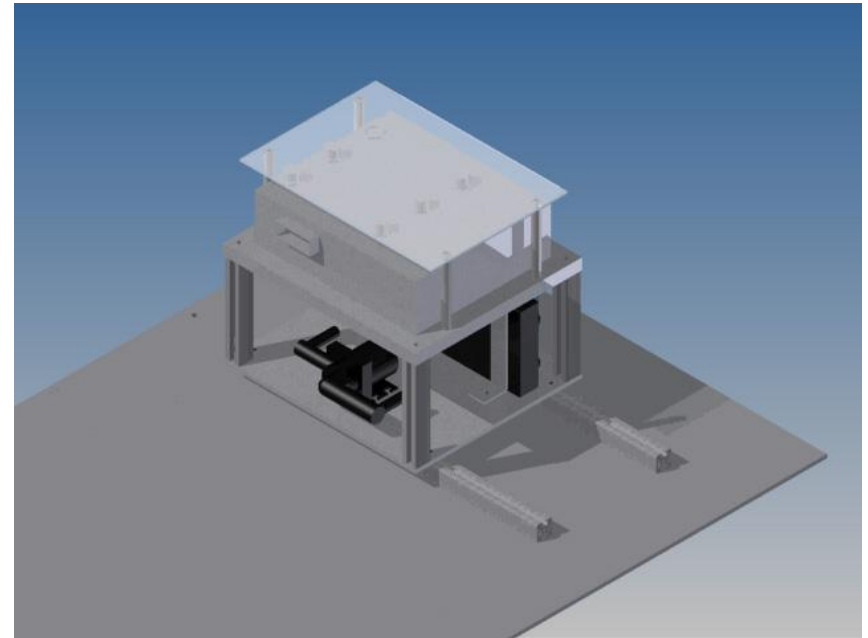
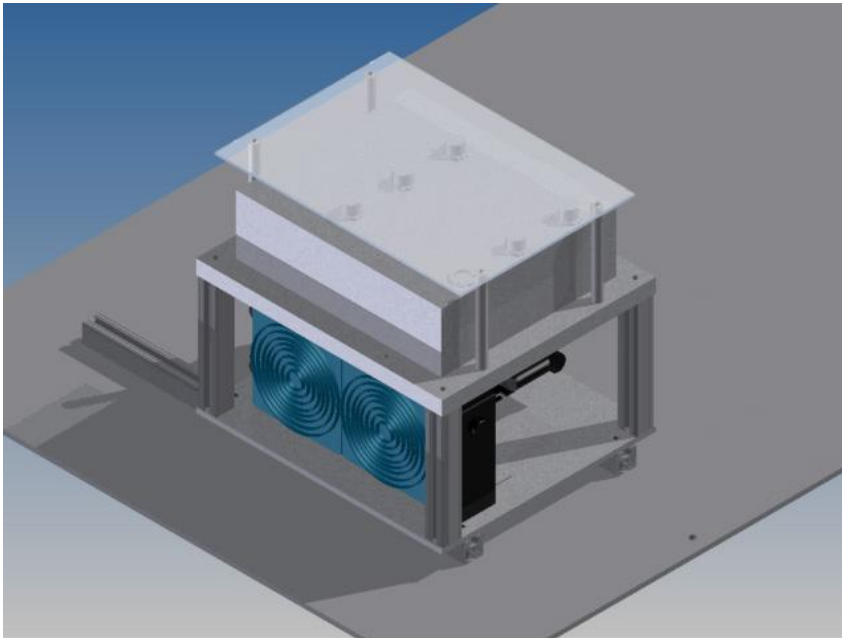
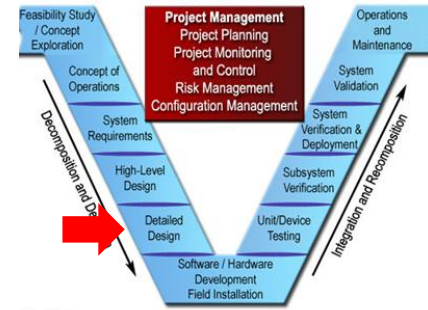
# Safety - Insulating Covers



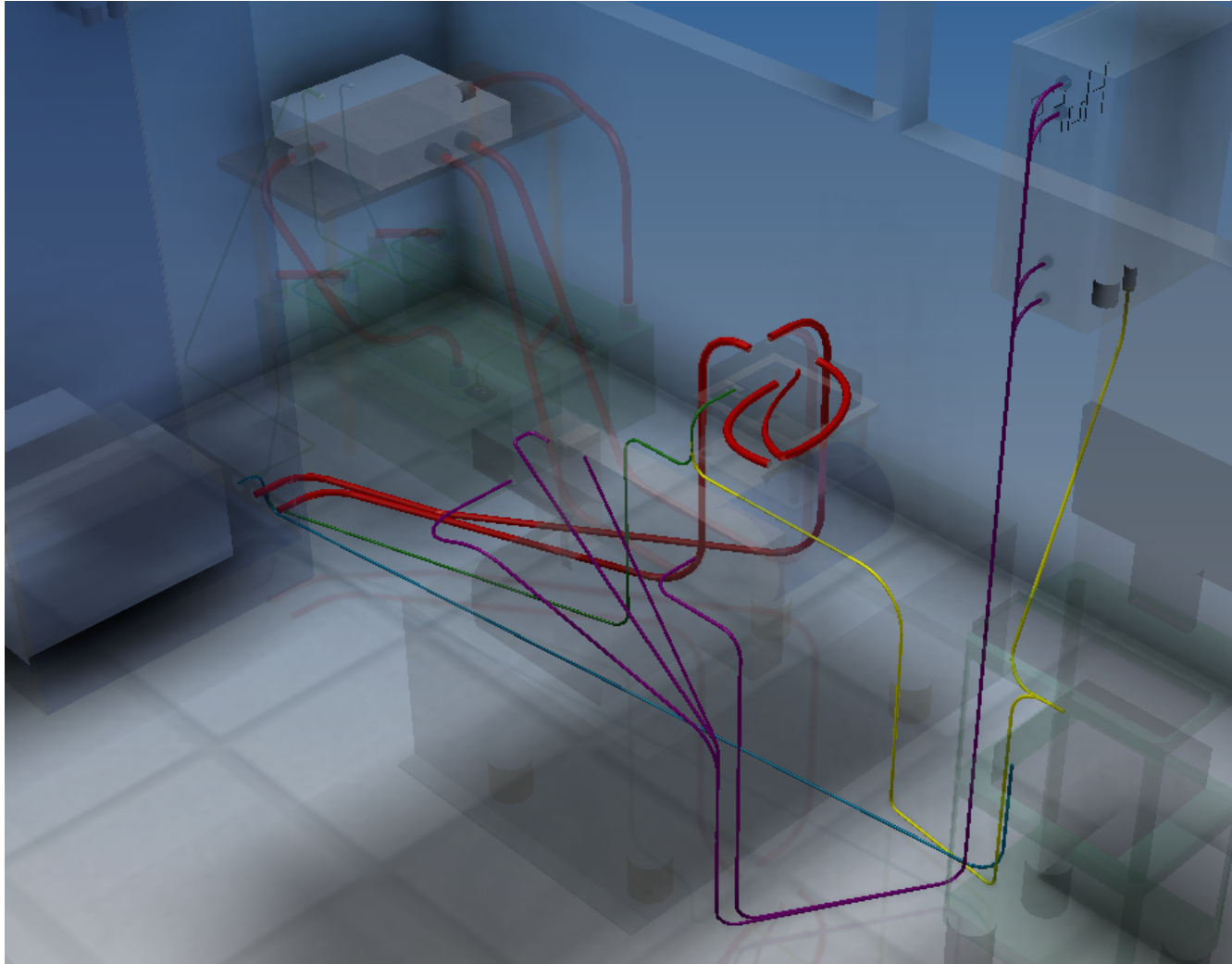
- Plastic Cover
  - Non-conductive
  - Transition temperature higher than cutoff temp
- Aluminum connecting rods



# Motor Controller - Cooling

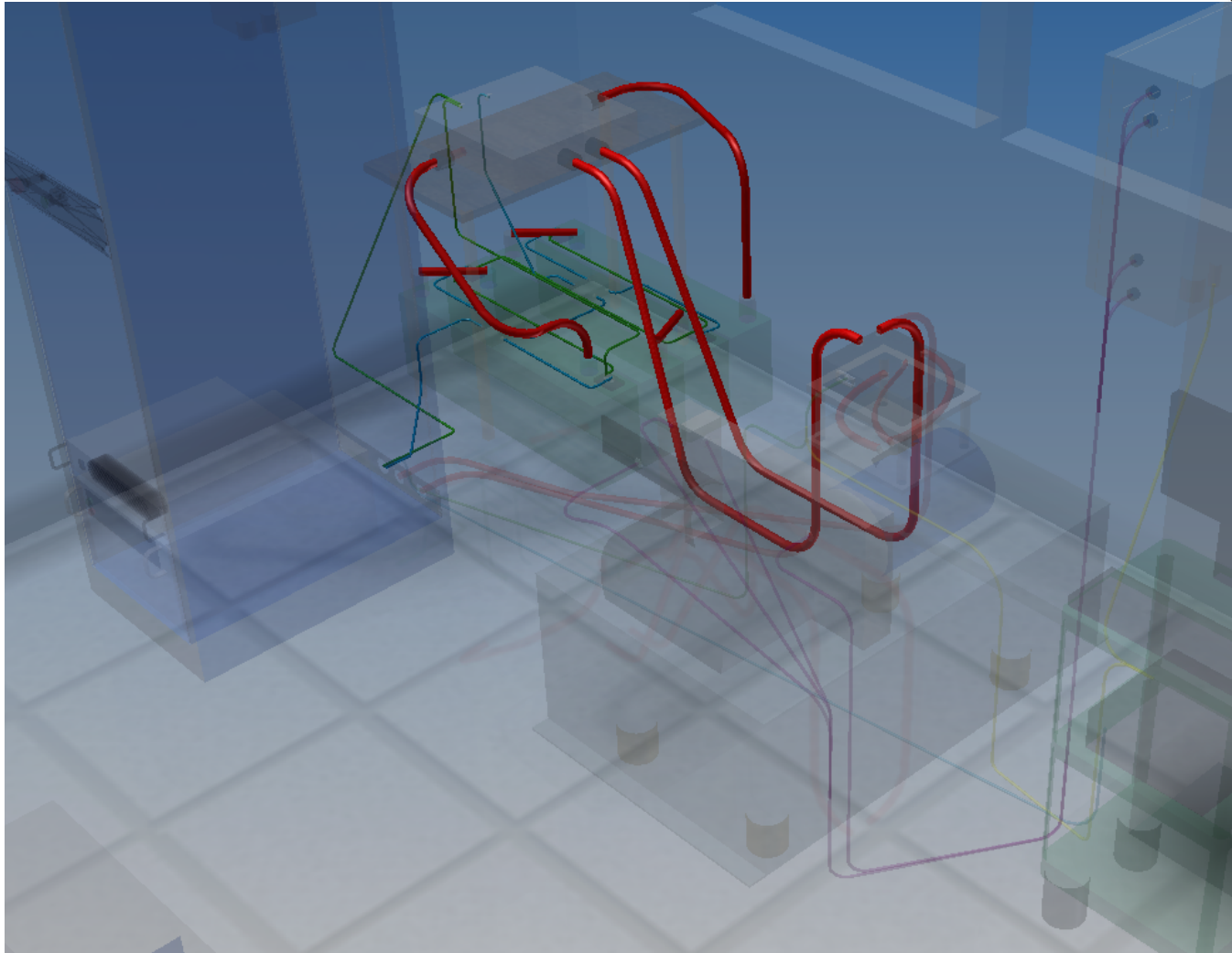
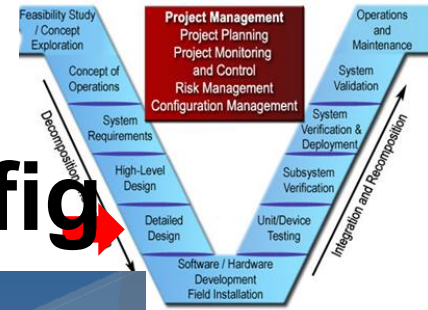


# Room Wiring - Testing Config



- HV Lines
- Huff Sensors
- Computer links
- CANbus

# Room Wiring - Integrated Config



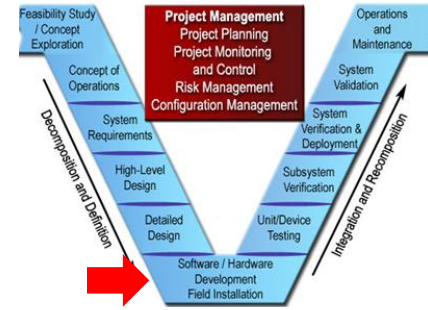
HV Lines

Huff Sensors

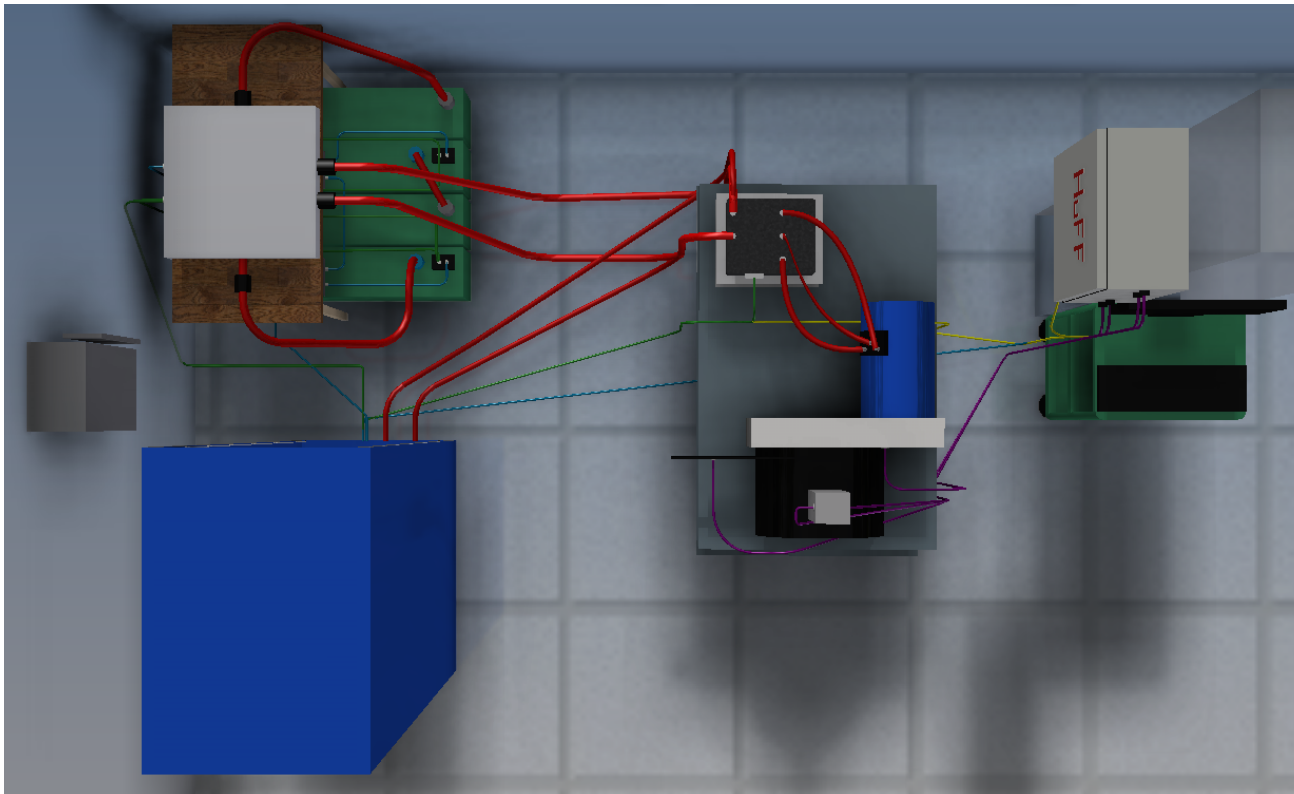
Computer links

CANbus

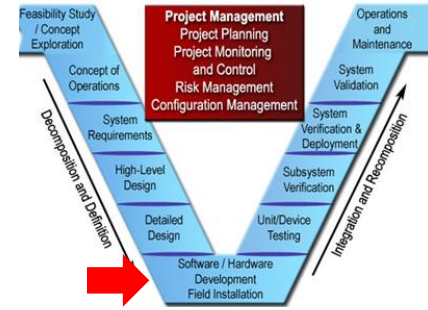
# System Integration



- Two configurations
  - Dyno Testing Configuration
  - Integrated Design Configuration



# Dynamometer

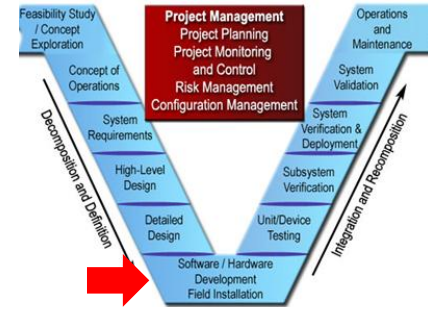


- Dyno Testing Configuration
  - Utilizes Windows PC with software
  - Records RPM and Torque
  - Controls dynamometer
- Integrated Design Configuration
  - Utilizes VSCADA computer
  - Records RPM and Torque
  - Controls dynamometer



# System Wide

- VSCADA
  - CAN data acquisition
  - Dyno data acquisition and control
  - Throttle
- TSI
  - Galvanic isolation from TSV packs
- TSV
  - Supplies power

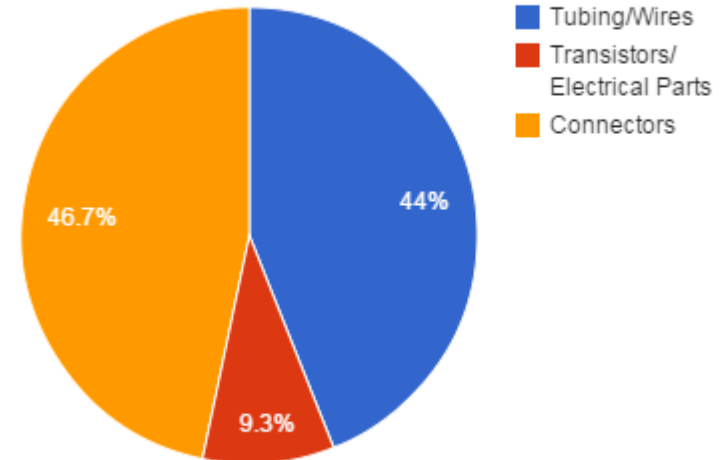


# Budget



- Wires/Tubing: \$203.92
  - Water cooling tubes - \$15.30
  - Power supply cables - \$188.62
- Transistor/Electrical Parts - \$43.22
  - Button switches - \$15.76
  - Transistors - \$27.46
- Connectors - \$216.55
  - Power locks - \$214.36
  - Pin Connector - \$2.19
- Total - \$463.69

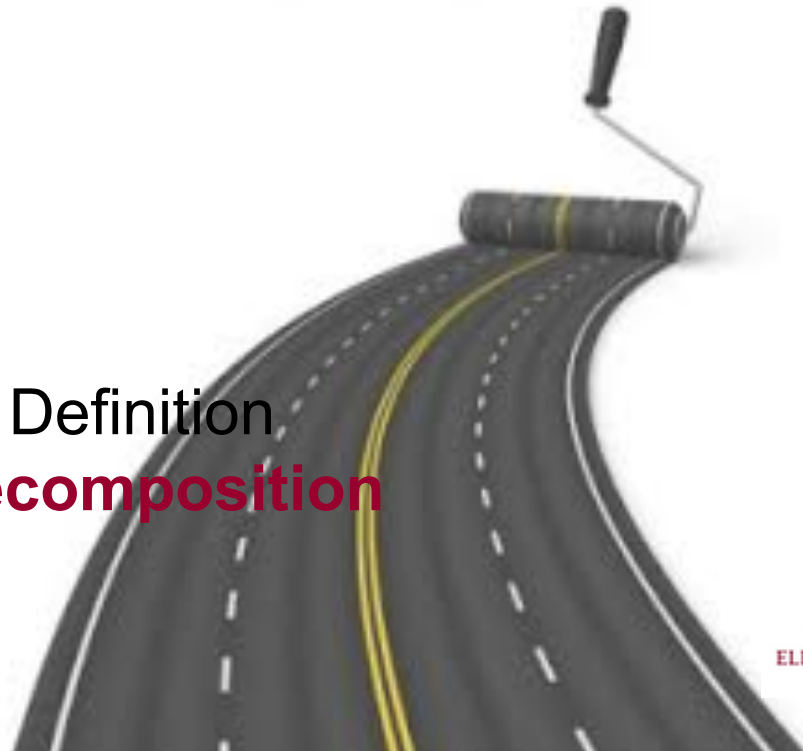
Dyno Budget Breakdown





# Roadmap

10. Meet the Afternoon Teams
11. Interface Control and Assemblies Review
12. Vehicle Supervisory Control and Data Acquisition (VSCADA)
  - a. Daemon
  - b. Interfacing
  - c. User Applications
  - d. Data Storage
13. Dynamometer (DYNO)
  - a. Decomposition and Definition
  - b. Integration and Recomposition**



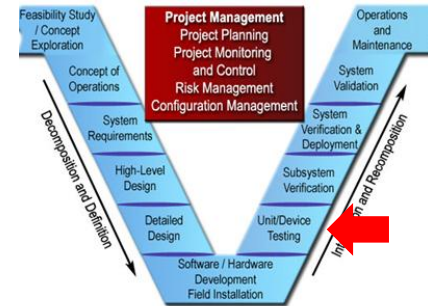
# Unit/Device Testing

- Throttle Testing
- Safety Shutoff
- Galvanic Isolation



# Throttle Testing

- Independant Solution - T001-2
  - Using Arduino system
  - Sweep the throttle in increments of 100 RPM
  - Value must be within 5 RPM with 90% confidence



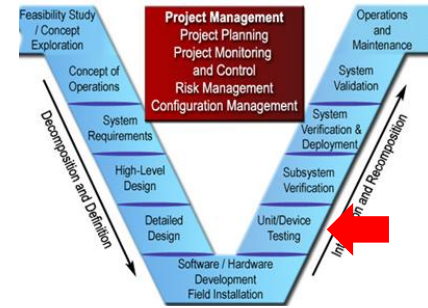
# Safety Shutoff



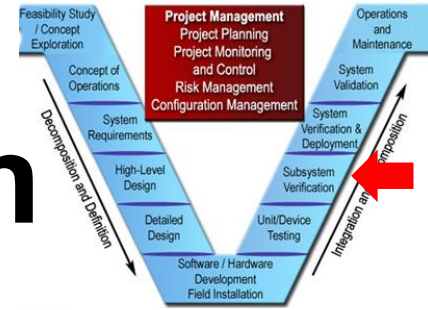
- Emergency Stop - T000-1
  - Press the stop, check that the system powers down
- Oil Temp. Shutoff - T000-3
  - Heat up sensor, check that the system powers down

# Galvanic Isolation

- Tested using an ohmmeter to measure resistance
  - Ensure ground on both sides
  - Grounds are not connected



# Subsystem Verification

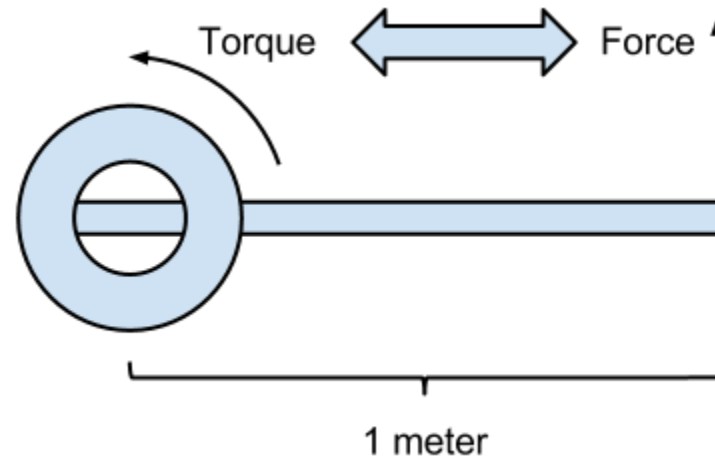


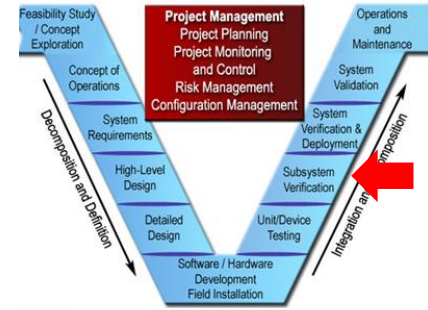
- Data Acquisition
  - Verification tests for:
    - Torque
    - Velocity
    - Current
    - Voltage
    - Temperature
    - Load control
- Simulation Results Comparison



# Data Acquisition

- Verify sensor accuracy - T001-1
  - Torque - verified with first principles
    - Calibrate with weights on the arm
    - Verify calibration with different weights



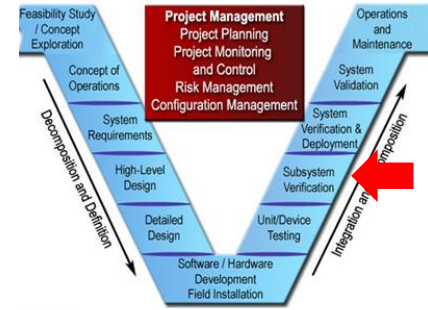


# Data Acquisition

- Verify sensor accuracy - T001-2
  - Motor Velocity - redundant measurements
    - Dynamometer encoder
    - Motor encoder
    - Handheld tachometer
  - Verified statistically







# Data Acquisition

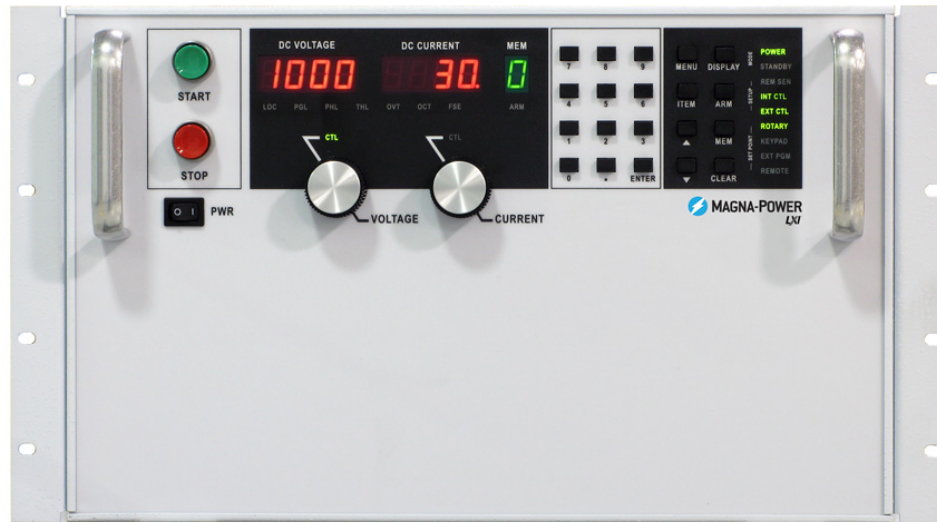
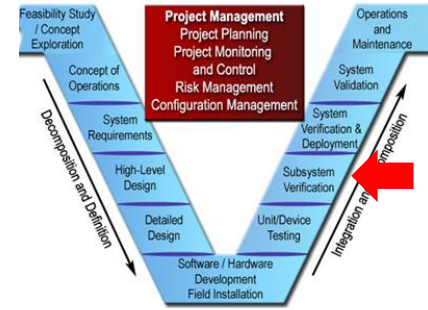
- Verify sensor accuracy - T001-3
  - Motor Current - redundant measurements
    - Motor controller output
    - Clamp sensor
  - Verified statistically



<http://www.hiokiusa.com/images/products/m9709.gif>

# Data Acquisition

- Verify sensor accuracy - T001-4
  - Motor Voltage - redundant measurements
    - Motor controller output
    - Power supply reading
  - Verified statistically



<http://www.magna-power.com/products/programmable-dc-power-supplies/ts-series>

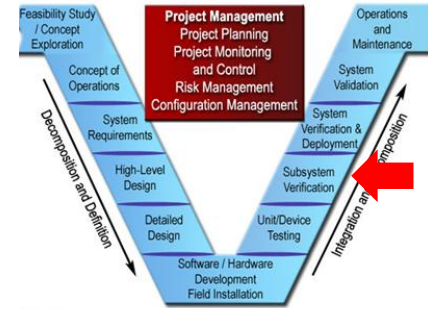


# Data Acquisition

- Verify sensor accuracy - T001-5
  - System Temperature - redundant measurements
    - Motor controller output (Motor/Controller temp.)
    - Handheld sensor
  - Verified statistically

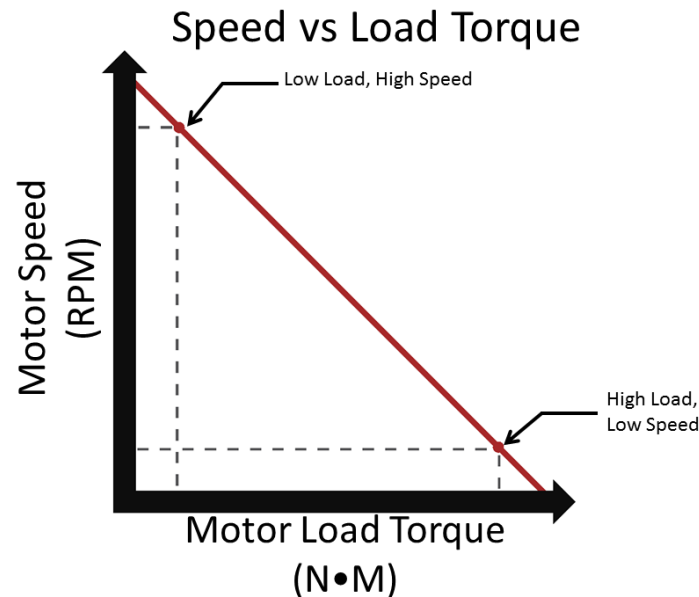


[http://www.chinoinc.com/products/sensors/mc1000/img/sensor\\_mc1000.jpg](http://www.chinoinc.com/products/sensors/mc1000/img/sensor_mc1000.jpg)



# Data Acquisition

- Verify sensor accuracy - T001-6
  - Load Variance - check torque response
    - Use steady motor RPM
    - Vary load and check that torque varies

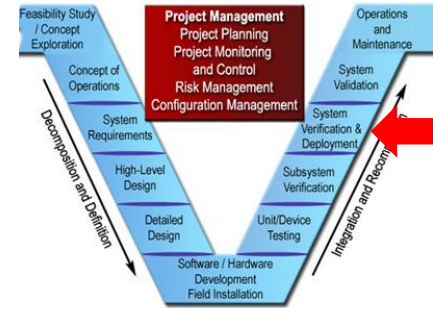


<http://curriculum.vexrobotics.com/sites/default/files/7.7.1%20Torque%20vs.%20Speed.PNG>

# Simulation Results Comparison

- Using the graph of torques that were inputted to the motor in simulation to create scripts
- Serially control the valve of the dynamometer to follow a similar curve.
- Correlate the data acquired from model and test set-up

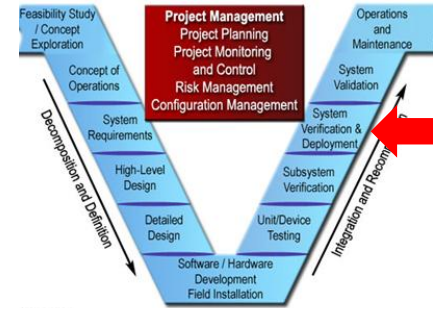
# System Verification



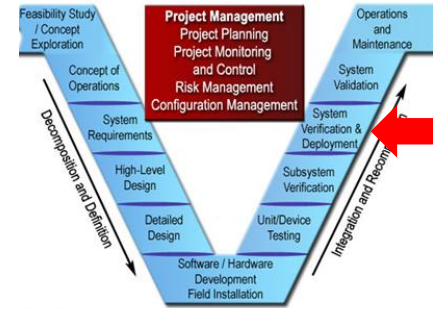
- VSCADA Tests
  - Data acquisition
  - Throttle control
- TSI Tests
  - Checks for galvanic isolation
- TSV Pack Tests
  - With 4 packs
  - With 1 pack

# VSCADA Tests

- VSCADA data acquisition via CAN
- VSCADA data acquisition from Huff Box
- VSCADA throttle control
  - Uses data from CAN and Huff Box
  - Control Dynamometer valve via Huff Box



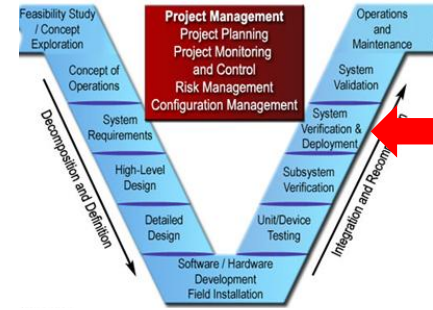
# TSI Tests



- Run the dynamometer with the TSI attached
  - T002-4 (assumes the TSI works properly)
  - If the system powers down:
    - It is not galvanically isolated
  - If the system runs:
    - It is properly isolated



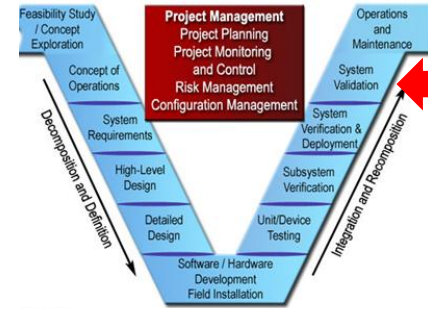
# TSV Tests



- Run the dynamometer with TSV power
  - If 4 packs have been completed:
    - Connect the packs to the TSI
  - If only 1 pack has been completed:
    - Connect the pack in series with the power supply
    - Power supply will make up voltage difference

# System Validation

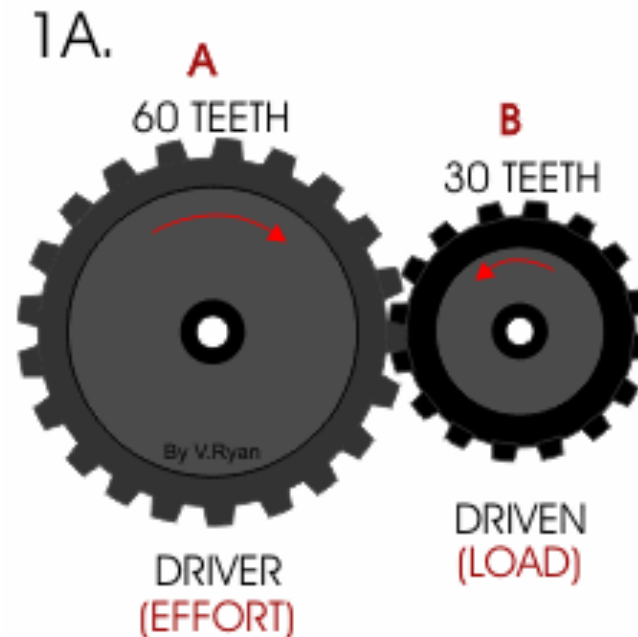
- Gear Ratio
- Torque Curve
- Final Demonstration



# Gear Ratio

$$R = \frac{T_B}{T_A}, \quad R = \frac{\omega_A}{\omega_B}$$

- Single Gear
- Top speed 80 mph
- 6500 rpm



<http://www.technologystudent.com/gears1/gears5.htm>

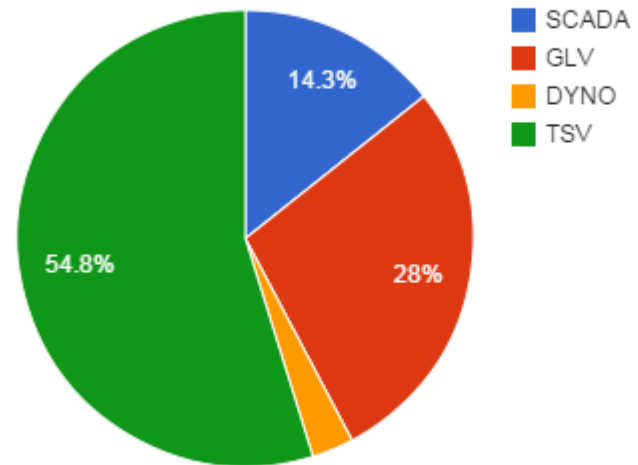
# Torque Curve

- At minimum:
  - Must show velocity vs. torque at an estimated load
- Goal:
  - Will show velocity vs. torque at several load points
  - Will show power consumption
- Ideal:
  - 3D graph of velocity, torque, and load

# Budget

- Initially Allocated Money: \$5000
  - Dyno - \$148
  - SCADA - \$715
  - GLV - \$1397.90
  - TSV - \$2739.10

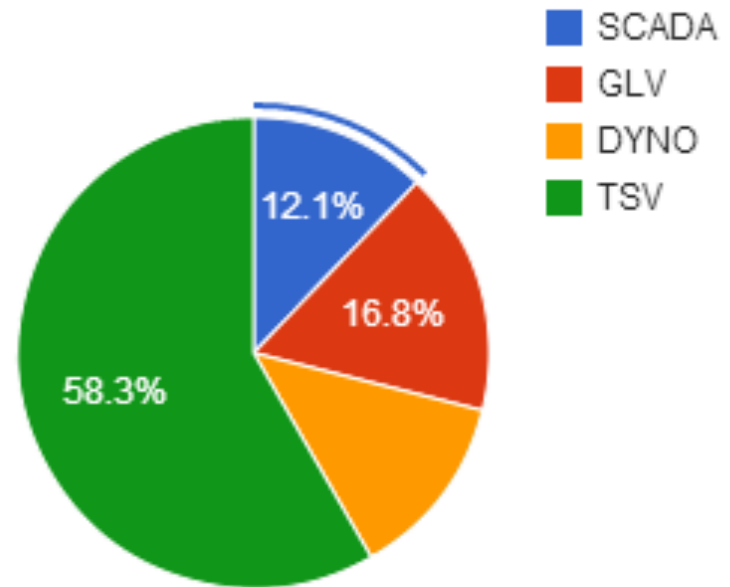
Budgeted Money



# Budget

- Money Spent So Far
  - Dyno - \$471.37
  - SCADA - \$448.37
  - GLV - \$618.99
  - TSV - \$2152.08
- Total Spent - \$3691

Money Spent



# Budget

- Money Remaining:
  - DYNO - \$323.37
  - SCADA - \$266.63
  - GLV - \$778.91
  - TSV - \$587.02
- Total Remaining:  
\$1309

