Department of Computer Science and Engineering
the University of Texas at Arlington

Architecture Design Specification

BehindtheCurtain Enterprises

Project AVALANCHE

Team Members:
Kyle Burgess
Kyle Crumpton
Austen Herbst
Bilal Nawaz
Jason Sprowl

Last Updated: 2/27/2013 11:00 am
Table of Contents

Document Revision History ............................................................................................................. 7
List of Figures .................................................................................................................................... 8
List of Tables ...................................................................................................................................... 9
1. Introduction .................................................................................................................................. 10
   1.1 Overview .................................................................................................................................. 10
   1.2 Product Overview ..................................................................................................................... 10
   1.3 Product Scope .......................................................................................................................... 10
2. Meta Architecture ......................................................................................................................... 12
   2.1 Architectural Vision ............................................................................................................... 12
   2.2 Guiding Principles ................................................................................................................. 12
   2.3 Tradeoffs .................................................................................................................................. 13
3. Architecture Overview .................................................................................................................. 14
   3.1 GUI Layer ............................................................................................................................... 14
   3.2 Data Processing Layer ............................................................................................................ 15
   3.3 Bluetooth Layer ..................................................................................................................... 15
   3.4 Data Acquisition Layer .......................................................................................................... 15
   3.5 Local Storage Layer ............................................................................................................... 15
   3.6 Network Layer ....................................................................................................................... 15
   3.7 Cloud Layer ............................................................................................................................ 15
4. GUI Layer ..................................................................................................................................... 16
   4.1 Description ............................................................................................................................. 16
   4.2 Purpose .................................................................................................................................... 16
   4.3 Function ................................................................................................................................... 16
   4.4 Dependencies .......................................................................................................................... 16
   4.5 Interlayer Interfaces ............................................................................................................... 17
   4.6 Public Interfaces ...................................................................................................................... 17
   4.7 Processing ............................................................................................................................... 17
<table>
<thead>
<tr>
<th>Subsystem</th>
<th>Function</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Processing</td>
<td>Encode/Package Data</td>
<td>Behind the Curtain Enterprises</td>
</tr>
<tr>
<td></td>
<td>Transfer Data</td>
<td>Local Storage Layer</td>
</tr>
<tr>
<td></td>
<td>Subsystems</td>
<td>Public Interfaces</td>
</tr>
<tr>
<td></td>
<td>Interlayer Interfaces</td>
<td>Dependencies</td>
</tr>
<tr>
<td></td>
<td>Dependencies</td>
<td>Function</td>
</tr>
<tr>
<td></td>
<td>Purpose</td>
<td>Description</td>
</tr>
<tr>
<td></td>
<td>Description</td>
<td>Public Interfaces</td>
</tr>
<tr>
<td></td>
<td>Processing</td>
<td>Interlayer Interfaces</td>
</tr>
<tr>
<td></td>
<td>Data</td>
<td>Dependencies</td>
</tr>
<tr>
<td></td>
<td>Subsystems</td>
<td>Function</td>
</tr>
<tr>
<td></td>
<td>Interlayer Interfaces</td>
<td>Dependencies</td>
</tr>
<tr>
<td></td>
<td>Public Interfaces</td>
<td>Function</td>
</tr>
<tr>
<td></td>
<td>Processing</td>
<td>Dependencies</td>
</tr>
<tr>
<td></td>
<td>Data</td>
<td>Function</td>
</tr>
</tbody>
</table>

AVALANCHE
# Network Layer

## Description

### Purpose

### Function

### Dependencies

### Interlayer Interfaces

### Public Interfaces

### Processing

### Data

## Subsystems

### Device Memory

## Cloud Storage Layer

### Description

### Purpose

### Function

### Dependencies

### Interlayer Interfaces

### Public Interfaces

### Processing

### Data

### Subsystems

#### Transfer

#### Encryption

#### Decryption

### Database
# Document Revision History

<table>
<thead>
<tr>
<th>Revision Number</th>
<th>Revision Date</th>
<th>Description</th>
<th>Rationale</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>1/25</td>
<td>Initial Draft</td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td>1/27</td>
<td>ADS Review Draft</td>
<td>Revised according to peer review input.</td>
</tr>
<tr>
<td>2.0</td>
<td>2/4</td>
<td>Baseline Draft</td>
<td>Revised according to ADS review feedback.</td>
</tr>
<tr>
<td>2.1</td>
<td>2/26</td>
<td>Revised Design Updates</td>
<td>DDS Creation called for changes made to ADS.</td>
</tr>
</tbody>
</table>
## List of Figures

<table>
<thead>
<tr>
<th>Figure #</th>
<th>Title</th>
<th>Page #</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-1</td>
<td>Product Scope Overview</td>
<td>11</td>
</tr>
<tr>
<td>3-1</td>
<td>Layer Block Diagram</td>
<td>14</td>
</tr>
<tr>
<td>4-1</td>
<td>GUI Layer Diagram</td>
<td>16</td>
</tr>
<tr>
<td>5-1</td>
<td>Data Processing Layer Diagram</td>
<td>21</td>
</tr>
<tr>
<td>6-1</td>
<td>Bluetooth Layer Diagram</td>
<td>27</td>
</tr>
<tr>
<td>7-1</td>
<td>Data Acquisition Layer Diagram</td>
<td>30</td>
</tr>
<tr>
<td>8-1</td>
<td>Local Storage Layer Diagram</td>
<td>33</td>
</tr>
<tr>
<td>9-1</td>
<td>Network Layer Diagram</td>
<td>36</td>
</tr>
<tr>
<td>10-1</td>
<td>Cloud Storage Layer Diagram</td>
<td>41</td>
</tr>
<tr>
<td>11-1</td>
<td>System Data Flow Map</td>
<td>44</td>
</tr>
</tbody>
</table>
## List of Tables

<table>
<thead>
<tr>
<th>Table #</th>
<th>Title</th>
<th>Page #</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-1</td>
<td>GUI Interlayer Interfaces</td>
<td>17</td>
</tr>
<tr>
<td>4-2</td>
<td>GUI Public Interfaces</td>
<td>17</td>
</tr>
<tr>
<td>4-3</td>
<td>Input Public Interfaces</td>
<td>18</td>
</tr>
<tr>
<td>4-4</td>
<td>Output Interlayer Interfaces</td>
<td>19</td>
</tr>
<tr>
<td>4-5</td>
<td>Output Public Interfaces</td>
<td>19</td>
</tr>
<tr>
<td>4-6</td>
<td>Configuration Public Interfaces</td>
<td>20</td>
</tr>
<tr>
<td>5-1</td>
<td>Data Processing Public Interfaces</td>
<td>22</td>
</tr>
<tr>
<td>5-2</td>
<td>Real Time Processing Public Interfaces</td>
<td>23</td>
</tr>
<tr>
<td>5-3</td>
<td>Post Processing Interlayer Interfaces</td>
<td>24</td>
</tr>
<tr>
<td>5-4</td>
<td>Save Data Interlayer Interfaces</td>
<td>25</td>
</tr>
<tr>
<td>5-5</td>
<td>Sync Interlayer Interfaces</td>
<td>25</td>
</tr>
<tr>
<td>6-1</td>
<td>Bluetooth Interlayer Interfaces</td>
<td>28</td>
</tr>
<tr>
<td>6-2</td>
<td>Sync Interlayer Interfaces</td>
<td>29</td>
</tr>
<tr>
<td>6-3</td>
<td>Transfer Data Interlayer Interfaces</td>
<td>29</td>
</tr>
<tr>
<td>7-1</td>
<td>Data Acquisition Interlayer Interfaces</td>
<td>31</td>
</tr>
<tr>
<td>7-2</td>
<td>Data Acquisition Public Interfaces</td>
<td>31</td>
</tr>
<tr>
<td>7-3</td>
<td>Read CAN Bus Public Interfaces</td>
<td>32</td>
</tr>
<tr>
<td>7-4</td>
<td>Encode/Package Data Interlayer Interfaces</td>
<td>32</td>
</tr>
<tr>
<td>8-1</td>
<td>Local Storage Layer Interlayer Interfaces</td>
<td>34</td>
</tr>
<tr>
<td>8-2</td>
<td>Device Memory Interlayer Interfaces</td>
<td>35</td>
</tr>
<tr>
<td>9-1</td>
<td>Network Layer Interlayer Interfaces</td>
<td>37</td>
</tr>
<tr>
<td>9-2</td>
<td>Encryption Interlayer Interfaces</td>
<td>39</td>
</tr>
<tr>
<td>9-3</td>
<td>Decryption Interlayer Interfaces</td>
<td>39</td>
</tr>
<tr>
<td>10-1</td>
<td>Cloud Storage Layer Interlayer Interfaces</td>
<td>42</td>
</tr>
<tr>
<td>10-2</td>
<td>Database Interlayer Interfaces</td>
<td>43</td>
</tr>
<tr>
<td>11-1</td>
<td>Data Flow Definitions</td>
<td>46</td>
</tr>
<tr>
<td>11-2</td>
<td>Producer-Consumer Relationships</td>
<td>47</td>
</tr>
<tr>
<td>12-1</td>
<td>Requirements Mapping</td>
<td>50</td>
</tr>
</tbody>
</table>
1. Introduction

1.1 Overview

This Architectural Design Document provides a high level design of project AVALANCHE based on system requirements. This document defines the high level system architecture for project AVALANCHE including guiding principles, scope, layer definitions, subsystems, and data flows between subsystems, and finally will cover basic testing considerations. This document’s aim is to cover the high level aspects of the design where the low level design will be covered in the Detailed Design Specification.

1.2 Product Overview

The AVALANCHE is an add-on module for a pre-existing gauge by CPC/Redline Gauges, a gauge that provides a digital display for all sensors of a racing vehicle. The existing CPC/Redline Gauge is targeted towards snowmobiles, but is useable with any racing vehicle so long as the vehicle has the required sensors installed. AVALANCHE will add mobile app functionality to provide racers and maintenance specialists a tool for data logging, vehicle diagnosis, and real time stats feedback, for all existing sensors. The goal is to benefit racing enthusiasts by providing data analysis not offered with previous gauges.

1.3 Product Scope

Team BehindtheCurtain will produce a prototype of Project AVALANCHE consisting of the following components:

1.3.1 Hardware Module:

The hardware module is an add-on component for the existing Redline Gauge that will interface with the existing CAN bus. The module will consist of the following:

CAN bus – The CAN bus is responsible for providing communication from the existing gauge to the hardware module.

Microcontroller – The microcontroller is responsible for providing data acquisition and data packaging for Bluetooth transmission.
Bluetooth chip – The Bluetooth chip provides Bluetooth networking between the hardware module and mobile devices.

1.3.2 Mobile Devices:

This consists of any iOS and/or Android device that have the AVALANCHE mobile application, which will be developed for both the iOS and Android operating systems, installed. The mobile device will communicate wirelessly through Bluetooth to the hardware module.

1.3.3 Server:

The server will provide storage for user accounts and saved runs for the user. Server storage of runs will be handled in real time if 3G networking is available.

1.3.4 Mobile Applications:

The mobile applications will provide data processing and a user interface for the user.

1.3.5 Server Software:

The server software will handle controlling the server hardware and provide account security and real-time access.

1.3.6 Microcontroller Software:

The Microcontroller will provide data acquisition and Bluetooth networking functionality.

Figure 1.1 Product Scope Overview
2. Meta Architecture

2.1 Architectural Vision

Project AVALANCHE will consist of six layers: data acquisition, Bluetooth, data processing, network, storage, and GUI. The design will allow the functionality of each layer to be independent of one another. Data flow within Project AVALANCHE will be clearly defined and flow between layers as needed. The system will maintain a level of integrity; data that flows from one layer to the next will be consistent and accurate. We have designed our project in a way where the responsibilities of each layer allow an easily implementable design.

2.2 Guiding Principles

2.2.1 Accuracy

The system is intended to display accurate gauge information and be fully consistent with the data displayed on the actual gauge. Any processing taking place within the system should not affect the accuracy of any sensor readings.

2.2.2 Reliability

The system shall be designed so that communication between the layers happens in a reliable manner. Minimal data should be lost in transition.

2.2.3 Scalability

The system shall be designed in a way to scale out in regards to both application and functionality. New features should be easily implementable and new sensor data and metrics should be able to be added to the system with minimal effort.

2.2.4 Modularity

The system shall be designed in such a way that each layer and subsystem be as independent as possible. Making changes to one layer or subsystem will not affect other layers or subsystems.

2.2.5 Integrity
The system shall be able to operate in real time consistently while displaying accurate values pulled from the original gauge. Each layer and subsystem will be dependable as a whole.

2.3 Tradeoffs

- Initially we decided to have an input and output layer. After discussion we decided it would be more feasible to consolidate the two into a single GUI layer which encompasses both user inputs and outputs to the mobile device.

- We decided to split the Network Layer into the Bluetooth layer and the Network layer to encompass different types of networking to simplify the interactions between layers.

- We decided to move the majority of the data processing onto the mobile device to simplify hardware design.
3. Architecture Overview

Project AVALANCHE will be divided into 6 distinct Layers: the GUI Layer, Data Processing Layer, Bluetooth Layer, Data Acquisition Layer, Local Storage Layer, and Network Layer.

3.1 GUI Layer
The GUI Layer acts as the intermediary between users and Project AVALANCHE. It displays the data from the physical gauge in a graphical form, providing both sensor readouts and charts and metrics for past runs, as well as parsing user input into the system.

3.2 Data Processing Layer

The Data Processing Layer processes raw data received from the physical gauge via the Bluetooth Layer and uses user inputted configuration data to format the device input into an user readable form which is outputted to the user through the GUI Layer. It will also send this data to be saved in the Local Storage Layer, either locally or on the server via the Network Layer. These saved runs will be able to be analyzed by the Data Processing Layer to create charts, graphs, and metrics for run analysis by the user.

3.3 Bluetooth Layer

The Bluetooth Layer will handle the Bluetooth pairing between the hardware module and the mobile device and will be responsible for transferring gauge sensor data from the Data Acquisition Layer to the Data Processing Layer over Bluetooth.

3.4 Data Acquisition Layer

The Data Acquisition Layer encompasses the hardware module and will be responsible for pulling gauge data off the CAN Bus and packaging that data for transfer over the Bluetooth Layer.

3.5 Local Storage Layer

The Local Storage Layer will be responsible for storing saved runs for later analysis. The Local Storage Layer will encompass local device storage on the mobile phone. The Local Storage Layer will be accessed by the Data Processing Layer directly through the mobile device file system.

3.6 Network Layer

The Network Layer is responsible for secure and reliable data transfer between the Data Processing and Cloud Storage Layer. It will use AES and RSA encryption to make sure that all communication between the layers is secure and private and TCP to ensure that the data is transferred successfully between the layers.

3.7 Cloud Layer

The Cloud Layer will be responsible for storing saved runs for later analysis. The Cloud layer will encompass the database storage on the server. The Cloud Storage Layer will be accessed by the Storage Layer directly through the Network Layer. The Cloud Storage Layer will also store user account information on the database for access by any mobile device with the required applications.
4. GUI Layer

4.1 Description

The GUI Layer resides physically on the mobile device. The GUI Layer acts as the intermediary between users and Project AVALANCHE. It displays the data from the physical gauge in a graphical form, providing both sensor readouts and charts and metrics for past runs, as well as parsing user input into the system.

4.2 Purpose

The purpose of the GUI Layer is to provide an easy to use, user interface that displays charts, digital gauges, and metrics for both real time and past runs. This interface also allows the user to setup the AVALANCHE system and configure the layout according to the user’s preference.

4.3 Function

This layer is responsible for displaying a graphical representation of the data and taking user input to adjust system settings and layout configuration.

4.4 Dependencies
The GUI Layer is dependent on the Mobile Device for user input and interface display. It is also dependent on the Data Processing layer to provide real time and post data processing.

### 4.5 Interlayer Interfaces

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GUI Layer</td>
<td>Data Processing Layer</td>
<td>Configuration data</td>
</tr>
<tr>
<td>Data Processing Layer</td>
<td>GUI Layer</td>
<td>Processed data for graphical output</td>
</tr>
</tbody>
</table>

Table 4-1 GUI Interlayer Interfaces

### 4.6 Public Interfaces

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>External (User Input)</td>
<td>GUI</td>
<td>User input from the mobile device</td>
</tr>
<tr>
<td>GUI</td>
<td>External (User Input)</td>
<td>Rendered interface and charts for mobile device display</td>
</tr>
</tbody>
</table>

Table 4-2 GUI Public Interfaces

### 4.7 Processing

This layer will process user input from the mobile device and render the dynamic graphical interface.

### 4.8 Data

The GUI Layer takes data from the Data Processing layer, through either Real Time or Post Processing to render the graphical charts and gauges. Also, when the user interacts with the interface, listeners will pass data to either modify the layout or change the vehicle configuration settings.

### 4.9 Subsystems

#### 4.9.1 Input

##### 4.9.1.1 General

Allows user to configure what data is displayed on the screen and modify AVALANCHE system settings.
4.9.1.2 Assumptions

In order for the Input subsystem to function properly the user must have either an iPhone or Android device with the appropriate app installed.

4.9.1.3 Responsibilities

The Input subsystem is responsible for taking user input from the iPhone or Android device and passing this information to the appropriate subsystem. If the user chooses to modify the layout, the data is passed to the Output subsystem. If the user chooses to modify the vehicle sensor configuration, the data is passed to the Configuration subsystem.

4.9.1.4 Interlayer Interfaces

n/a

4.9.1.5 Public Interfaces

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
<th>Information Required</th>
<th>Information Returned</th>
</tr>
</thead>
<tbody>
<tr>
<td>inputCommand</td>
<td>Listen for input from the user.</td>
<td>User’s menu selection.</td>
<td>none</td>
</tr>
</tbody>
</table>

Table 4-3 Input Public Interfaces

4.9.2 Output

4.9.2.1 General

Display’s a graphical representation of gauge and graph data based on configurations from the Input subsystem.

4.9.2.2 Assumptions

This subsystem will assume that there is a data set, either stored or real time, available to render.

4.9.2.3 Responsibilities

The Output subsystem is responsible for rendering the charts and metrics to be viewed on the user’s mobile device.
4.9.2.4 Interlayer Interfaces

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
<th>Information Required</th>
<th>Information Returned</th>
</tr>
</thead>
<tbody>
<tr>
<td>getRealData</td>
<td>Receive Real Time data from the Data Processing layer.</td>
<td>none</td>
<td>Real time data</td>
</tr>
<tr>
<td>getPostData</td>
<td>Receive Post data from the Data Processing layer.</td>
<td>none</td>
<td>Post processed data</td>
</tr>
</tbody>
</table>

Table 4-4 Output Interlayer Interfaces

4.9.2.5 Public Interfaces

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
<th>Information Required</th>
<th>Information Returned</th>
</tr>
</thead>
<tbody>
<tr>
<td>updateGUI</td>
<td>Update the user interface.</td>
<td>none</td>
<td>Rendered interface graphics and charts.</td>
</tr>
</tbody>
</table>

Table 4-5 Output Public Interfaces

4.9.3 Configuration

4.9.3.1 General

Takes configuration data about user’s physical gauge.

4.9.3.2 Assumptions

In order for the Configuration subsystem to function properly the user must have input the appropriate configuration data for their vehicle.

4.9.3.3 Responsibilities

The Configuration subsystem is responsible for taking vehicle/sensor specific configurations and modifying the AVALANCHE system accordingly.
4.9.3.4 Interlayer Interfaces

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
<th>Information Required</th>
<th>Information Returned</th>
</tr>
</thead>
<tbody>
<tr>
<td>setConfigData</td>
<td>Set the user data that configures the Data Processing Layer to work with the users vehicle.</td>
<td>Physical gauge configuration setup data.</td>
<td>none</td>
</tr>
</tbody>
</table>

Table 4-6 Configuration Interlayer Interfaces

4.9.3.5 Public Interfaces

n/a
5. Data Processing Layer

5.1 Description

The Data Processing Layer resides physically on the mobile device. The Data Processing Layer decodes and processes the data received from the Bluetooth Layer for output through the GUI Layer and for storage in the Local Storage Layer. The Data Processing Layer will have two distinct processing modes, real time and post data processing. Real time data processing will handle parsing data received from either the gauge. Post processing will analyze a prior run saved in storage and produce graphs and metrics based on the data from the run. The Data Processing Layer will have the following subsystems: Real Time Processing, Post Processing, Save, and Sync.

5.2 Purpose

The purpose of the Data Processing Layer is to both process the data stream pulled from the gauge for output in a user readable form and to process past runs to produce analytics for tuning purposes.

5.3 Function
This layer is responsible for both processing the raw data from the physical gauge into a form that will be easily displayed in the GUI Layer and analyzing past runs for a graphical summary of the run.

5.4 Dependencies

The Data Processing Layer is dependent on the Bluetooth Layer to transmit data from the hardware module for real-time processing, the Storage Layer for storage of past runs, and the Network layer to communicate with the server.

5.5 Interlayer Interfaces

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GUI Layer</td>
<td>Data Processing Layer</td>
<td>Configuration data</td>
</tr>
<tr>
<td>Data Processing Layer</td>
<td>GUI Layer</td>
<td>Processed data for graphical output</td>
</tr>
<tr>
<td>Storage Layer</td>
<td>Data Processing Layer</td>
<td>Saved runs for post processing</td>
</tr>
<tr>
<td>Data Processing</td>
<td>Local Storage Layer</td>
<td>Save runs</td>
</tr>
<tr>
<td>Network Layer</td>
<td>Data Processing Layer</td>
<td>Saved runs for post processing</td>
</tr>
<tr>
<td>Bluetooth Layer</td>
<td>Data Processing</td>
<td>Byte stream containing gauge sensor data.</td>
</tr>
<tr>
<td>Data Processing</td>
<td>Bluetooth Layer</td>
<td>Bluetooth protocol data.</td>
</tr>
</tbody>
</table>

Table 5-1 Data Processing Interlayer Interfaces

5.6 Public Interfaces

n/a

5.7 Processing

The layer will process the data being pulled from the Bluetooth Layer in real-time for output in the GUI Layer in gauge form. It will then be able to process saved data into chart and graph form.

5.8 Data

The layer will receive real-time gauge data from the Bluetooth Layer. The layer will send processed real-time to both the GUI Layer and the Storage Layer, either locally or through the Network Layer.
layer will be able to pull the saved data from the Storage Layer, either locally or through the Network Layer.

5.9 Subsystems

5.9.1 Real Time Processing

5.9.1.1 General

The Real Time Processing subsystem will handle real time data pulled from the physical gauge that is transferred over the Bluetooth Layer. It will use the configuration data inputted in the GUI layer to decode this data in a user readable form and then transfer the data for output in the GUI layer and the Save Data Subsystem for storage.

5.9.1.2 Assumptions

In order for the Real Time Processing Layer to function properly, it needs to have gauge configuration data from the GUI Layer and an ongoing data stream from the Bluetooth layer containing run data from the physical gauge. The subsystem also assumes that the processed data will be storage space in the Storage Layer to save the processed run to.

5.9.1.3 Responsibilities

The Real Time Processing subsystem is responsible for taking in run data from the Bluetooth Layer and decoding it using the configuration data obtained from the GUI Layer so that it is in an user readable form. This will allow the user to view sensor data in real time.

5.9.1.4 Interlayer Interfaces

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
<th>Information Required</th>
<th>Information Returned</th>
</tr>
</thead>
<tbody>
<tr>
<td>outputGaugeData</td>
<td>Send sensor data to GUI Layer to be outputted.</td>
<td>Requested sensor data</td>
<td>Processed sensor data</td>
</tr>
<tr>
<td>gaugeConfigure</td>
<td>Acquire the user data that configures the Data Processing Layer to work with the users vehicle.</td>
<td>Physical gauge configuration setup data.</td>
<td>none</td>
</tr>
<tr>
<td>gaugeSensorInput</td>
<td>Pull sensor data from Bluetooth Layer</td>
<td>Sensor data from Bluetooth Layer</td>
<td>none</td>
</tr>
</tbody>
</table>

Table 5-2 Real Time Processing Interlayer Interfaces
5.9.1.5 Public Interfaces

n/a

5.9.2 Post Processing

5.9.2.1 General

The Post Processing Subsystem will reanalyze saved runs to create graphs, tables, and metrics for tuning purposes. The subsystem will pull the requested run to analyze from the Storage Layer, either saved locally or accessed from the server through the Network Layer, based on user configurations in the GUI Layer. Once the run is accessed the subsystem will process the data to create the requested graphs, tables, and metrics.

5.9.2.2 Assumptions

The subsystem will assume that there will be runs in the Storage Layer to analyze.

5.9.2.3 Responsibilities

The Post Processing Subsystem is responsible for processing past runs and creating the graphs, tables, and metrics that the user requested for tuning analysis.

5.9.2.4 Interlayer Interfaces

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
<th>Information Required</th>
<th>Information Returned</th>
</tr>
</thead>
<tbody>
<tr>
<td>outputPostData</td>
<td>Output post processed data to the GUI layer.</td>
<td>A saved run to analyze.</td>
<td>Analyzed run data.</td>
</tr>
<tr>
<td>loadRun</td>
<td>Load a run to analyze from the Storage Layer.</td>
<td>Whether the run is saved locally or on the server</td>
<td>None.</td>
</tr>
</tbody>
</table>

Table 5-3 Post Processing Interlayer Interfaces

5.9.2.5 Public Interfaces

n/a

5.9.3 Save Data

5.9.3.1 General
The Save Data Subsystem will receive run data from the Real Time Processing Subsystem and then save that data locally in the Storage Layer.

5.9.3.2 Assumptions

The run data being saved will already be processed for the physical gauge that the data was pulled from. Also data will be saved locally first before being transported to the server.

5.9.3.3 Responsibilities

This subsystem is responsible for saving real time run data locally in the Storage Layer.

5.9.3.4 Interlayer Interfaces

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
<th>Information Required</th>
<th>Information Returned</th>
</tr>
</thead>
<tbody>
<tr>
<td>saveData</td>
<td>Save run data locally.</td>
<td>Processed real time data.</td>
<td>Data to be saved.</td>
</tr>
</tbody>
</table>

Table 5-4 Save Data Interlayer Interfaces

5.9.3.5 Public Interfaces

n/a

5.9.4 Sync

5.9.4.1 General

The Sync subsystem will sync the mobile device with the Bluetooth Layer via Bluetooth.

5.9.4.2 Assumptions

The Bluetooth chip on the hardware module will be open for syncing.

5.9.4.3 Responsibilities

This subsystem is responsible for syncing with the Bluetooth Layer and for confirming the status of the sync for the entire layer.

5.9.4.4 Interlayer Interfaces

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
<th>Information Required</th>
<th>Information Returned</th>
</tr>
</thead>
<tbody>
<tr>
<td>sync</td>
<td>Sync the Data Processing Layer and Bluetooth Layer.</td>
<td>Bluetooth id of the Bluetooth chip on the hardware module.</td>
<td>Sync Status</td>
</tr>
</tbody>
</table>

Table 5-5 Sync Interlayer Interfaces
5.9.4.5 Public Interfaces

n/a
6. Bluetooth Layer

![Bluetooth Layer Diagram](#)

6.1 Description

The Bluetooth Layer resides physically on the hardware module. The Bluetooth Layer communicates internally with the Data Acquisition Layer and the Data Processing Layer. The Bluetooth Layer has two sub systems: Sync and Transfer Data.

6.2 Purpose

The purpose of the Bluetooth layer is to wirelessly transfer data from the Data Acquisition Layer to the Data Processing Layer.

6.3 Function

The Bluetooth Layer will receive packaged data from the Data Acquisition Layer. It will then take the data and wirelessly transmit it to the Data Processing layer. The Bluetooth layer does not do any processing on the data. Its sole function is for data transmission.
6.4 Dependencies

The Bluetooth Layer is dependent on the Data Acquisition Layer to deliver packaged data.

6.5 Interlayer Interfaces

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th>Data Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Processing</td>
<td>Bluetooth</td>
<td>Bluetooth protocol data</td>
</tr>
<tr>
<td>Bluetooth</td>
<td>Data Processing</td>
<td>Encoded Byte Stream containing a single data point for all sensors</td>
</tr>
<tr>
<td>Data Acquisition</td>
<td>Bluetooth</td>
<td>Encoded Byte Stream containing a single data point for all sensors.</td>
</tr>
</tbody>
</table>

Table 6-1 Bluetooth Interlayer Interfaces

6.6 Public Interfaces

n/a

6.7 Processing

n/a

6.8 Data

The Bluetooth layer sends packaged data for each of the Redline Gauges twelve (12) sensors.

6.9 Subsystems

6.9.1 Sync

6.9.1.1 General

The purpose of this sub system is to establish a connection between the hardware module and the mobile device.

6.9.1.2 Assumptions

Bluetooth functionality will be enabled on the mobile device

6.9.1.3 Responsibilities
This subsystem will be responsible for pairing the mobile device to the hardware layer

6.9.1.4 Interlayer Interfaces

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Information Required</th>
<th>Information Returned</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sync</td>
<td>Pair hardware device to mobile device</td>
<td>Bluetooth Protocol Data</td>
<td>Sync Status</td>
</tr>
</tbody>
</table>

Table 6-2 Sync Interlayer Interfaces

6.9.1.5 Public Interfaces

n/a

6.9.2 Transfer Data

6.9.2.1 General

The purpose of this subsystem is to receive data from the microcontroller and send it to the mobile device

6.9.2.2 Assumptions

Power will be supplied by the CAN bus line. The microcontroller will send properly packaged data.

6.9.2.3 Responsibilities

This subsystem will be responsible for sending all of the data from the hardware device to the mobile device.

6.9.2.4 Interlayer Interfaces

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Information Required</th>
<th>Information Returned</th>
</tr>
</thead>
<tbody>
<tr>
<td>Receive Data</td>
<td>Data from the microcontroller is received.</td>
<td>Packaged Byte Stream representing all sensors</td>
<td>Packaged Byte Stream representing all sensors</td>
</tr>
<tr>
<td>Send Data</td>
<td>Send packaged data via Bluetooth to mobile devices</td>
<td>Bluetooth Protocol Data</td>
<td>Packaged Byte Stream representing all sensors</td>
</tr>
</tbody>
</table>

Table 6-3 Transfer Data Interlayer Interfaces

6.9.2.5 Public Interfaces

n/a
7. Data Acquisition Layer

7.1 Description

The Data Acquisition Layer will run on the hardware module. The Data Acquisition Layer will act as a median between the existing gauge and the Bluetooth layer to acquire the data from the CAN Bus, and to encode/package the data for Bluetooth transfer.

7.2 Purpose

Obtain the data from the existing gauge and process it so we can send the data wirelessly over Bluetooth.

7.3 Function

The Data Acquisition Layer obtains data from the CAN bus line, then encodes/packages the data for Bluetooth transfer, and then sends the data to the Bluetooth Layer.

7.4 Dependencies

The Data Acquisition layer will be dependent on the CAN Bus to gather the data, and power.
7.5 Interlayer Interfaces

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th>Data Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Acquisition</td>
<td>Bluetooth</td>
<td>Encoded Byte Stream containing a single data point for all sensors.</td>
</tr>
</tbody>
</table>

Table 7-1 Data Acquisition Interlayer Interfaces

7.6 Public Interfaces

<table>
<thead>
<tr>
<th>Source Layer</th>
<th>Destination Layer</th>
<th>Data Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exterior</td>
<td>Data Acquisition</td>
<td>Raw Byte Stream from CAN Bus line.</td>
</tr>
</tbody>
</table>

Table 7-2 Data Acquisition Public Interfaces

7.7 Processing

The Data Acquisition Layer will process a raw byte stream to return a packaged and encoded byte stream that is ready for data transfer.

7.8 Data

The Data Acquisition layer receives raw data for each of the Redline Gauges twelve (12) sensors, and returns an encoded/packaged byte stream representing the same 12 sensors.

7.9 Subsystems

7.9.1 Read CAN Bus

7.9.1.1 General

The purpose of this sub system is to read the raw data off of the CAN Bus

7.9.1.2 Assumptions

We will assume that the CAN Bus is transferring data properly. We will assume that we are receiving power from the CAN Bus.

7.9.1.3 Responsibilities

This subsystem will be responsible for reading the raw data off of the CAN Bus

7.9.1.4 Interlayer Interfaces
7.9.1.5 Public Interfaces

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Information Required</th>
<th>Information Returned</th>
</tr>
</thead>
<tbody>
<tr>
<td>acquireData</td>
<td>Data from the CAN Bus is read into the microcontroller</td>
<td>Raw Byte Stream from the CAN Bus line</td>
<td>Raw Byte Stream from the CAN Bus line</td>
</tr>
</tbody>
</table>

*Table 7-3 Read Can Bus Public Interfaces*

7.9.2 Encode/Package Data

7.9.2.1 General

The purpose of this sub system is to package end encode the raw data for wireless data transfer.

7.9.2.2 Assumptions

We will assume that our sponsor will give us the format in which the data is coming off of the CAN Bus.

7.9.2.3 Responsibilities

This subsystem will be responsible for packaging and encoding the raw data.

7.9.2.4 Interlayer Interfaces

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Information Required</th>
<th>Information Returned</th>
</tr>
</thead>
<tbody>
<tr>
<td>outputData</td>
<td>Send packaged data to Bluetooth Layer</td>
<td>Bluetooth Protocol Data</td>
<td>Packaged Byte Stream representing all sensors</td>
</tr>
</tbody>
</table>

*Table 7-4 Encode/Package Data Interlayer Interfaces*

7.9.2.5 Public Interfaces

n/a
8. Local Storage Layer

![Local Storage Diagram](image)

**Figure 8-1 Storage Layer Diagram**

8.1 Description

The Local Storage Layer is responsible for storing past run data. The Local Storage Layer will reside both on the mobile device, and the Device Memory Subsystem.

8.2 Purpose

The main purpose of the Local Storage Layer is accumulation of the previous race statistics and gauge data for future analysis. Local Storage Layer is also responsible for communicating with the Data Processing Layer as well as the Network Layer to store records that might be retrieved at a later time or by a different consumer. The Local Storage Layer will be accessed by the Data Processing Layer directly through the mobile device file system.

8.3 Function

The Local Storage Layer has the principal functionality of acting as the temporary data storage location for the whole system. The Local Storage Layer mainly stores the past race statistics and gauge data for later analysis and presentation onto the mobile applications and handles run data requests from the Data Processing Layer.

8.4 Dependencies

The Local Storage Layer is dependent upon the Data Processing Layer to receive data to save into the Device Memory for temporary storage. The Local Storage Layer is also dependent upon the Network Layer for communication to and from the cloud based server.
8.5 Interlayer Interfaces

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local Storage Layer</td>
<td>Data Processing</td>
<td>Receives data from Save Data subsystem of Data Processing Layer</td>
</tr>
<tr>
<td>Local Storage Layer</td>
<td>Network Layer</td>
<td>Sends data to Encryption subsystem of Network Layer</td>
</tr>
<tr>
<td>Data Processing Layer</td>
<td>Local Storage Layer</td>
<td>The query acknowledged from the Data Processing Layer is processed and the specified path is determined.</td>
</tr>
</tbody>
</table>

Table 8-1 Local Storage Layer Interlayer Interfaces

8.6 Public Interfaces

n/a

8.7 Processing

The Local Storage Layer will process requests from the Data Processing Layer and will find the run on the local storage.

8.8 Data

The Local Storage Layer sends and receives byte streams comprising statistics from the gauge input or the database to use for either analytics or storing.

8.9 Subsystems

8.9.1 Device Memory

8.9.1.1 General

The Device Memory Subsystem will reside physically on the mobile device. The Device Memory basically interacts with the Save Data subsystem of the Data Processing. The data from Save Data subsystem of the Data Processing provides the Device Memory subsystem with the real-time data to be saved locally.
8.9.1.2 Assumptions

Data coming in from the Save Data subsystem of the Data Processing Layer is of the right type and format.

8.9.1.3 Responsibilities

- Reception of data from the Data Processing Layer
- Forwarding the data to the Encryption subsystem of the Network Layer

8.9.1.4 Interlayer Interfaces

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
<th>Information Required</th>
<th>Information Returned</th>
</tr>
</thead>
<tbody>
<tr>
<td>receiveFromSaveData</td>
<td>Receives data from Save Data subsystem of the Data Processing Layer</td>
<td>Byte Stream</td>
<td>N/A</td>
</tr>
<tr>
<td>sendToEncryption</td>
<td>Sends data to the Encryption subsystem of Network Layer</td>
<td>Byte Stream</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Table 8-2 Device Memory Interlayer Interfaces

8.9.1.5 Public Interfaces

None
9. Network Layer

9.1 Description

The entirety of the Network Layer will be implemented on both the mobile devices, iOS and Android, and on the cloud server. The Network layer interfaces with the Data Processing layer and the Storage layer to bring a cloud storage aspect to Project AVALANCHE. The Network layer has three subsystems: the Encryption Subsystem, Transfer Subsystem, and Decryption Subsystem.

9.2 Purpose

The purpose of the network layer is to transport packets between the mobile device and the cloud.

9.3 Function
The Network Layer receives packets from the Data Processing Layer which is part of the mobile device. The data received from this layer is decrypted, read, encrypted, and transferred to the Storage Layer.

### 9.4 Dependencies

The Network Layer is dependent upon the Data Processing Layer to receive encrypted data to transfer to the cloud that exists in the Storage Layer. The Network Layer requires an internet connection to be able to send packets to the cloud.

### 9.5 Interlayer Interfaces

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local Storage Layer</td>
<td>Network Layer</td>
<td>Network layer receives a packet from the storage layer to send to encrypt and send to data processing layer</td>
</tr>
<tr>
<td>Data Processing Layer</td>
<td>Network Layer</td>
<td>Network layer receives a packet from the data processing layer to write to encrypt and send to the storage layer.</td>
</tr>
<tr>
<td>Network Layer</td>
<td>Data Processing layer</td>
<td>Network layer sends packet received from storage layer to data processing layer for analysis</td>
</tr>
<tr>
<td>Network Layer</td>
<td>Local Storage Layer</td>
<td>Network layer sends packet received from data processing layer to send to the storage layer for later use.</td>
</tr>
<tr>
<td>Network Layer</td>
<td>Local Storage Layer</td>
<td>Network layer encrypts packet it received from the data processing layer.</td>
</tr>
<tr>
<td>Network Layer</td>
<td>Local Storage Layer</td>
<td>Server side decrypting of packets transported to the storage layer through the network layer.</td>
</tr>
</tbody>
</table>

Table 9-1 Network Layer Interlayer Interfaces

### 9.6 Public Interfaces

n/a

### 9.7 Processing

The Network Layer will process incoming and outgoing packets to and from both the storage and data processing layers by encrypting, decrypting, sending, and receiving packets.
9.8 Data

The Network Layer sends and receives network packets containing information from the gauge or the database to use for either analytics or storing.

9.9 Subsystems

9.9.1 Transfer

9.9.1.1 General

The transfer subsystem interfaces with both the mobile application and the server to send and receive packets based on whether a read or write is being performed.

9.9.1.2 Assumptions

The transfer subsystem assumes a 3G connection is available to transfer data to the server. This subsystem assumes a TCP connection.

9.9.1.3 Responsibilities

The transfer subsystem is responsible for regulating the flow of data between the data processing layer and the storage layer.

9.9.1.4 Interlayer Interfaces

n/a

9.9.1.5 Public

n/a

9.9.2 Encryption

9.9.2.1 General

The Encryption Subsystem encrypts username and password data for secure transfer between the data processing and storage layers. Only username and password data is encrypted; sensor data is passed through without change.

9.9.2.2 Assumptions

Assumes the use of an encryption key to specify the manner in which the message will be encoded. Assumes only authorized users can read encrypted data.

9.9.2.3 Responsibilities
The encryption subsystem is responsible for running AES 128 encryption on data and uses RSA 1024 encryption for key exchange before sending it through the outside world to the cloud.

9.9.2.4 Interfaces

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
<th>Requires</th>
<th>Information Returned</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encrypt Packet</td>
<td>Runs AES 128 encryption converting the packet to cipher-text and uses RSA 1024 encryption for key exchange to be read only by authorized users.</td>
<td>n/a</td>
<td>Encrypted cipher-text</td>
</tr>
</tbody>
</table>

Table 9-2 Encryption Interlayer Interfaces

9.9.2.5 Public Interfaces

n/a

9.9.3 Decryption

9.9.3.1 General

The decryption subsystem is responsible for decrypting packets which were encrypted and sent to the storage layer through the network layer. If the data being handled is unencrypted sensor data then the data is passed through without change.

9.9.3.2 Assumptions

Assumes the packet was properly encrypted and in cipher text format which can be decoded using a decryption algorithm.

9.9.3.3 Responsibilities

The decryption subsystem is responsible for converting data sent from the network layer back to a readable format for authorized users using the shared AES key before being stored into the storage layer.

9.9.3.4 Interfaces

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
<th>Requires</th>
<th>Information Returned</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decrypt Packet</td>
<td>Decrypts a packet using the AES key to decode cipher-text into a readable format.</td>
<td>A proper decryption algorithm.</td>
<td>Decoded cipher-text</td>
</tr>
</tbody>
</table>

Table 9-3 Decryption Interlayer Interfaces
9.9.3.5 Public

n/a
10. Cloud Storage Layer

10.1 Description

The Cloud Storage Layer is responsible for storing past run data on the server. The Storage Layer will reside both in the cloud and the Database Subsystem.

10.2 Purpose

The main purpose of the Cloud Storage Layer is accumulation of the previous race statistics and gauge data for future analysis. Cloud Storage Layer is also responsible for communicating with the Storage Layer through the Network Layer to store records that might be retrieved at a later time or by a different consumer. The Cloud Storage Layer will be accessed by the Storage Layer directly through the Network Layer.

10.3 Function

The Cloud Storage Layer has the principal functionality of acting as the long term data storage location for the whole system. The Cloud Storage Layer mainly stores the past race statistics and gauge data for later analysis and presentation onto the mobile applications and handles run data requests from the Storage Layer.

10.4 Dependencies
The Cloud Storage Layer is dependent upon the Storage and Network Layers to receive data to save into the database for long term storage. The Cloud Storage Layer is also dependent upon the Storage Layer for communication to and from the mobile device.

### 10.5 Interlayer Interfaces

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cloud Storage Layer</td>
<td>Network Layer</td>
<td>Sends data to Encryption subsystem of Network Layer</td>
</tr>
<tr>
<td>Network Layer</td>
<td>Cloud Storage Layer</td>
<td>Stores data to Device Memory or Database subsystem of Cloud Storage Layer</td>
</tr>
</tbody>
</table>

**Table 10-1 Cloud Storage Layer Interlayer Interfaces**

### 10.6 Public Interfaces

n/a

### 10.7 Processing

The Storage Layer will process requests from the Data Processing Layer and will find the run either on the local storage or server database storage. It will also have to process requests to the database into SQL queries.

### 10.8 Data

The Cloud Storage Layer sends and receives byte streams comprising statistics from the gauge input or the database to use for either analytics or storing.

### 10.9 Subsystems

#### 10.9.1 Database

##### 10.9.1.1 General

The Database Subsystem will run on the cloud server. It will store saved runs to be accessed by the user or anyone the user gives access to.

##### 10.9.1.2 Assumptions

- Data provided by the Decryption subsystem of the Network Layer is of the right type and format so the data can be easily stored.
- There is enough memory for the storage of all the data that is to be stored in the database.
10.9.1.3 Responsibilities

- Data storage.
- Reception of data from the Decryption subsystem of the Network Layer
- Forwarding the data to the Encryption subsystem of the Network Layer

10.9.3.4 Interlayer Interfaces

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
<th>Information Required</th>
<th>Information Returned</th>
</tr>
</thead>
<tbody>
<tr>
<td>receiveFromDecryption</td>
<td>Receives data from Decryption subsystem of Network Layer</td>
<td>Byte Stream</td>
<td>N/A</td>
</tr>
<tr>
<td>sendToEncryption</td>
<td>Sends data to Encryption subsystem of Network Layer</td>
<td>Byte Stream</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Table 10-2 Database Interlayer Interfaces

10.9.1.5 Public Interfaces

n/a
11. Relationship Mapping

This section describes the relationships and data flows between the subsystems of Project AVALANCHE. Both the purpose and description each data flow and the producer consumer relationships between subsystems will be defined.

Figure 11-1 System Data Flow Map

11.1 Overview

This section describes the relationships and data flows between the subsystems of Project AVALANCHE. Both the purpose and description each data flow and the producer consumer relationships between subsystems will be defined.
Figure 11-1 gives an overview of all the data flows between subsystems throughout the entire system. Each of these data flows will be described in detail in detail below in Table 11-1. Each data flow is described by a letter and number combination corresponding to the layer in which the flow originates. Data flows originating in the Data Acquisition Layer will begin with A, Bluetooth Layer will begin with B, Data Processing Layer will begin with P, GUI Layer will begin with G, Network Layer will begin with N, and Storage Layer will begin with S.

### 11.2 Data Flow Definition

<table>
<thead>
<tr>
<th>Data Flow</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>Real time unformatted byte stream data being pulled from the CAN Bus on the existing gauge describing the status of each gauge sensor.</td>
</tr>
<tr>
<td>A2</td>
<td>Real time unformatted sensor data that was pulled from the CAN Bus.</td>
</tr>
<tr>
<td>A3</td>
<td>Gauge sensor data that has been packaged and formatted for Bluetooth transmission.</td>
</tr>
<tr>
<td>B1</td>
<td>Gauge sensor data that has been transmitted through the Bluetooth Layer.</td>
</tr>
<tr>
<td>B3</td>
<td>Sync State is delivered to MCU.</td>
</tr>
<tr>
<td>P1</td>
<td>Processed real time sensor data to be saved.</td>
</tr>
<tr>
<td>P2</td>
<td>Processed real time sensor data to be outputted in the GUI Layer.</td>
</tr>
<tr>
<td>P3</td>
<td>Post processed run data that has been put analyzed for graphs and metrics.</td>
</tr>
<tr>
<td>P4</td>
<td>Run data that is being stored locally in the Storage Layer.</td>
</tr>
<tr>
<td>P5</td>
<td>Gauge Sensor data that has been transmitted to Data Processing Layer</td>
</tr>
<tr>
<td>P6</td>
<td>Saved run data information</td>
</tr>
<tr>
<td>G1</td>
<td>User input detailing configurations, preferences, and desired output formats.</td>
</tr>
<tr>
<td>G2</td>
<td>Description of user desired output format.</td>
</tr>
<tr>
<td>G3</td>
<td>User configuration settings.</td>
</tr>
<tr>
<td>G4</td>
<td>Vehicle specific configuration settings needed to process raw sensor data.</td>
</tr>
</tbody>
</table>
Details the requested run to be analyzed.

GUI output to the user. The GUI will only be able to display up to 12 sensors at one time.

Current Configuration to Display

Saved run data to be analyzed from local storage.

Saved run data that is being transferred to the Network.

Saved run data that has been requested by Post Processing Subsystem being transferred via the Network Layer. Being sent from the cloud to the mobile device.

Run data to be sent over the internet using TCP. Will be encrypted if data contains usernames or passwords.

Run data being sent over the internet.

Decrypted run data to be post processed. Transferred from the cloud to the mobile device.

Run data to be saved in server database. Transferred from the mobile device to the cloud.

Table 11-1 Data Flow Definitions

11.3 Producer-Consumer Relationships

Figure 11-2 defines the producer-consumer relationships between subsystems. Further definitions of the specific data flows are defined further above in Table 11-1.
## Table 11-2 Producer-Consumer Relationships

<table>
<thead>
<tr>
<th>Producer-Consumer Relationships</th>
<th>Consumer Subsystems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Acquisition Layer</td>
<td>Bluetooth Layer</td>
</tr>
<tr>
<td>Read CAN-Bus</td>
<td>Sync</td>
</tr>
<tr>
<td>Encode/Package Data</td>
<td>X</td>
</tr>
<tr>
<td>Bluetooth Layer</td>
<td>Sync</td>
</tr>
<tr>
<td>Transfer Data</td>
<td>X</td>
</tr>
<tr>
<td>Real Time Processing</td>
<td>X</td>
</tr>
<tr>
<td>Post Processing</td>
<td>X</td>
</tr>
<tr>
<td>Save Data</td>
<td>X</td>
</tr>
<tr>
<td>Sync</td>
<td>B2</td>
</tr>
<tr>
<td>Input</td>
<td></td>
</tr>
<tr>
<td>Output</td>
<td></td>
</tr>
<tr>
<td>Configure</td>
<td></td>
</tr>
<tr>
<td>Device Memory</td>
<td></td>
</tr>
<tr>
<td>Database</td>
<td></td>
</tr>
<tr>
<td>Encryption</td>
<td></td>
</tr>
<tr>
<td>Transfer</td>
<td></td>
</tr>
<tr>
<td>Decryption</td>
<td></td>
</tr>
</tbody>
</table>

February 26, 2013

BehindtheCurtain Enterprises
12. Requirement Mapping

The purpose of the Requirement Mapping is to give a general overview of the requirements that our team specified in our System Requirements Specification document, and to demonstrate that our System Architecture fully covers and accounts for all of our requirements.

The following table contains all of our architecture related requirements, and for each specific requirement will dictate the layers that are affected by this requirement. If we have designed our system correctly, we should be able to observe that our system is flexible, and fully covers all of our requirements.
<table>
<thead>
<tr>
<th>Req #</th>
<th>Requirement</th>
<th>Data Acquisition Layer</th>
<th>Bluetooth Layer</th>
<th>Data Processing Layer</th>
<th>GUI Layer</th>
<th>Local Storage Layer</th>
<th>Cloud Storage Layer</th>
<th>Network Layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Read Data from CAN BUS</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.2</td>
<td>Profile Each Run</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.3</td>
<td>Bluetooth Capability</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.4</td>
<td>Mobile iPhone App</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.5</td>
<td>Mobile Android App</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.6</td>
<td>Microcontroller Data Acquisition</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.7</td>
<td>Microcontroller Packaged Output</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.8</td>
<td>Interface to CAN BUS</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.9</td>
<td>Configuration Support Page</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>4.10</td>
<td>Multi-Gauge Graphical User Interface</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>4.11</td>
<td>App Consistency</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.1</td>
<td>Real-Time Output</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.2</td>
<td>Reliable Data Transfer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>6.3</td>
<td>Mobile Cross-Compatibility</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Table 12-1 Requirements Mapping</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.4 Multi-threading</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9.1 Statistics Database</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9.2 User Accounts</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9.3 Encryption of Web Traffic</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9.4 Salt and Hash Passwords</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9.5 Accurate Gauge Display</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
13. Operating System Dependencies

13.1 Data Acquisition Layer

The Data Acquisition layer will be developed in C on a microcontroller. The chip will be required to interface with the CAN Bus line and pull data from the existing gauge. The customer will be providing preexisting assembler source, object code, source, and headers which will need to interface with the CAN Bus line.

13.2 Bluetooth Layer

The Bluetooth layer will be developed on the SMD-Module RN-41.

13.3 Data Processing Layer

The Data Processing Layer is dependent on both iOS and Android multi-threading capabilities as well as Objective C and Java math libraries. The Objective C portion will be developed using Xcode while the Android portion will be developed using Eclipse.

13.4 Network Layer

The network will be implemented on the server as well as both iOS and Android. It will be dependent on Java and Objective C TCP implementation.

13.5 Local Storage Layer

The storage layer will be dependent on Android and iOS application storage for the device memory subsystem. The database subsystem will be dependent on Java and Apache Tomcat as well as MySQL for the actual database storage.

13.6 Cloud Storage Layer

The database subsystem will be dependent on Java and Apache Tomcat as well as MySQL for the actual database storage.

13.7 GUI Layer
The GUI Layer will be dependent on the GUI libraries for both iOS and Android. The iOS app will be dependent on the Cocoa Touch UIKit. The Android application will be dependent upon the android.view.accessibility, android.graphics, and android.view.inputservice packages.
14. Testing Considerations

14.1 General Overview

The system architecture will be tested for requirement fulfillment with respect to the specifications in the System Requirement Specifications document. The modular design of the Project AVALANCHE System Architecture ensures that each layer will be tested independent of one another. Each testing consideration will be exclusive to the layer specified below.

14.2 Testing Approach

Each layer will be tested in a modular fashion to ensure it meets functional requirements. Testing will occur in incremental fashion as new components are added to the system. Among the tests we will be testing each layer for data integrity, integration testing, and data flows between adjacent layers. The following sections will cover the testing procedures specific to each layer individually.

14.3 Data Acquisition Layer

The Data Acquisition Layer is required to read the raw data off of the CAN Bus line, encode and package the data, and then send the byte stream to the Bluetooth Layer for wireless transfer.

14.3.1 Read CAN Bus

Send known dummy data across the CAN Bus and determine if what we receive is what we know the dummy data to be.

14.3.2 Encode/Package Data

After we Encrypt and Package the data, run the exact opposite on the package to determine that we get the same raw byte stream that we started with.

14.4 Bluetooth Layer

The Bluetooth Layer is required to wirelessly transfer the packaged data that we receive from the Data Acquisition Layer to the Data Processing Layer.

14.4.1. Sync

Ensure that both the Android and IPhone mobile apps are able to connect to our hardware module.
14.4.2 Transfer Data

We will transmit a known dummy package via Bluetooth to the Data Processing Layer. Since we will know the contents of the dummy package, we can validate that the Bluetooth Layer is sending the proper data.

14.5 Data Processing Layer

The Data Processing Layer will have to be tested for correctness of output, Bluetooth connection reliability, and speed of processing. The Data Processing Layer should be able to process incoming data reliably from the Bluetooth Layer in a real time fashion. The layer will also have to be tested to make sure that both the Android and iOS applications function correctly and in a similar function.

14.5.1 Real Time Processing

The Real Time Processing subsystem will have to be tested for correctness of output, and speed of calculations.

14.5.2 Post Processing

The Post Processing subsystem will need to be tested to make sure that is calculating the graphs and metrics correctly based on the loaded saved run.

14.5.3 Sync

The Sync subsystem will need to be tested to make sure that the Bluetooth connection is made correctly and is reliable.

14.5.4 Save Data

The Save Data subsystem will be tested to make sure that the processed data is saved correctly.

14.6 Network Layer

The Network Layer is required to transport packets between the Storage Layer and the Data Processing Layer. The Network Layer will be tested on both the data flows between layers, integrity of the data, and the encryption and decryption of data.

14.6.1 Transport

Verify that data sent to or received from other layers is consistent and remains unchanged.

14.6.2 Encryption

Verify that encrypted data can be successfully decrypted.

14.6.1 Decryption
Verify that decrypted data is fully decoded and in a readable format.

14.7 Local Storage Layer

The Storage layer is accountable for storing as well as maintaining data used for real-time and post processing along with debugging. The primary function of the Storage layer is to interconnect with Network Layer as well as Data Processing Layer, and accordingly, it is important for the system to ensure that a query is correctly entered and the particular data are outputted as they should be.

14.7.1 Device Memory

Testing technique will be used to make sure what is anticipated to be retrieved from the database is what is actually received. Database access will also be authenticated to make sure that correct exceptions are prepared when an effort is made to put unacceptable information into the system’s database.

14.8 Cloud Storage Layer

14.8.1 Database

Testing will be done to make sure that the data intended to be retrieved follows the anticipated path and is efficiently acknowledged by the appropriate module. Database access will be authenticated for this subsystem as well to make sure that correct exceptions are prepared when an effort is made to put unacceptable information into the system’s database.

14.9 GUI Layer

The GUI layer is required to act as the intermediary between users and Project AVALANCHE. It displays the data from the physical gauge in a graphical form, providing both sensor readouts and charts and metrics for past runs, as well as parsing user input into the system.

14.9.1 Input

Ensure that the interface is functioning properly by entering a series of different menu commands and checking the desired output.

14.9.2 Output

Verify that the charts, digital gauges, interface render properly by comparing the data points in the data processing layer with the output GUI.

14.9.3 Configure

Input many differences sets of vehicle and sensor settings to see if the appropriately modified results are output.
Appendix

List of Acronyms

CAN Bus- Controller Area Network bus