System Design Doc

Version 1.0

**Table Of Contents**

# Introduction

## Purpose

The purpose of this document is to provide a detailed description of the API designed to support the <Name> hardware component being used in avionic controls. In addition the document should provide reference to the hardware developer in order to track development and proper feature usability.

## Scope

Details discussed in the document encompass how the software will interface with the hardware, message passing interface methods, user access, and feature abstraction. Perspective will generally be geared towards the API but will also consider user perspective when discussing feature usability and security implementations

## References

* + 1. **-**

## Overview

# Design considerations

## Assumptions

User has knowledge of the hardware component.

## Constraints

The API is constructed for limited access. The intention is to isolate the login user to someone who is familiar with the intended use and knowledge of the hardware component being manipulated.

## System environment

System should be functional only in response to the complimentary hardware.

## Design Methodology

Using an Agile approach, a working code base is maintained in the master branch of a bit-bucket repository and functional features are added using branch merges.

# Architecture

## System Design

## System decomposition

### Functional decomposition tree

# Data design

## Data description

<How data is being stored and where>

## Data dictionary

<tabular data description>

# Component Design

Each component should use at least one separate thread, maybe separate processes for stability?

## Hardware driver

* + 1. Responsible for communicating with product hardware
    2. Submitting commands
    3. Receiving alerts

## Web Interface

* + 1. Responsible for communications with external points
    2. Users via webUI
    3. API for receiving automated commands

## Logic Engine

* + 1. Responsible for calculating product motion
    2. Calculating angles / segments
    3. Handling input streams

# Software Interface Design

## User Interface design

<web client designer should discuss this>

### Web Pages in a tree

<hierarchical view of web page layout>

* + - 1. Description

<description of each node>

* + - 1. Objects and Actions

<description of the actions needed to get from a parent node to any of it's child nodes>

### User Interface

<General 'flavor' of GUI and actions accessible to the user>

* + - 1. Screen Images

<screen shot of each node, web page>

## Module Interface Design

<how has the design maintained MVC standards>

# Class Diagrams

## Basic folders for class diagrams

## Basic class diagram

## Full class diagram

--these will differ vastly from the sample. Sample implements a login system--

* 1. **…**
  2. **…**
  3. **…**
  4. **…**

# Appendices

## A. Setup and Configuration

## B. Tool set and environment

## C. Implementation list

<should correlate with subsections 7.4->7.7>

## D. Log sheet

<time sheet of hours contributed by each team member>

## E. Test Report

-Subsections should focus on specific elements such as secure logon, feature error checks (inputting invalid parameters and not breaking the system)

## F. Test Cases

# Glossary

<tabular structure of acronyms used and their descriptions>

# Bibliography