System Design Doc

Version 1.0

**Table Of Contents**

# Introduction

## Purpose

The purpose of this document is to describe the design of software that can interface with a hardware device consisting of a raspberry pi that controls various components. These components could include a gyroscope, temperature sensor, motors, or other peripherals.

In addition, this document should enable current and future developers of this software to track development and proper feature usage.

## Scope

This document gives a description of the software architecture. It specifies the structure and design of some of the modules discussed in the SRS. Use cases for these modules will also be given.

## References

The user of this SDD may need the following documents for reference:

Software Requirements Specification. Last modified: Feb 22, 2018

Statement Of Work. Last modified: Fed 12, 2018

## Overview

T

# Design considerations

## Assumptions

The user has knowledge of how the hardware components function and what commands are necessary to make the hardware complete desired tasks.

## Constraints

The software is designed to operate only locally, on the desired user’s raspberry pi and computer. It is not designed to be accessible outside of this environment. It will be designed using Python, Rasperry Pi libraries, and Javascript.

## System environment

The server side of the system is designed to run only on the designated Raspberry Pi (running Raspbian) that is connected to the hardware. The client side of the system, including the web interface, is designed to run on any OS that can run Python and a web browser running Javascript.

## Design Methodology

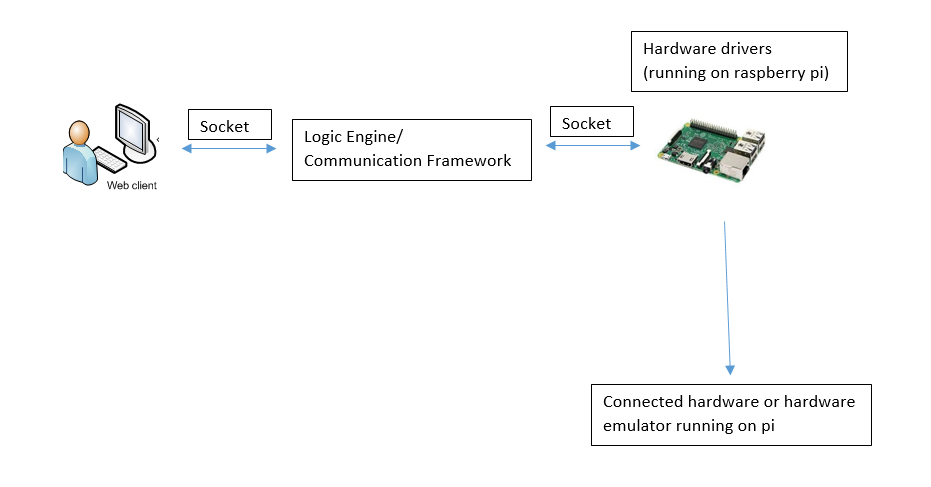
The system is divided into manageable sub-components that can be developed separately. A working code base is maintained in the master branch of a bitbucket repository and functional features are added using branch merges.

# Architecture

## Design Overview

The software includes a web server/client interface to communicate with the software controlling the hardware, lower level software for the pi that controls the hardware, and a driver to simulate the hardware. As part of the server/client interface, sockets handle communication between the server and client and carry errors and alerts.

## High Level Diagram



## Design Rationale

The software is divided into these components because they can be largely be developed separately in a small team. Component functionality which is dependent on other components can be added after those components have been developed. Sockets are used for all communication to/from the client and server to give a central place to transmit and log commands and alerts.

# Data design

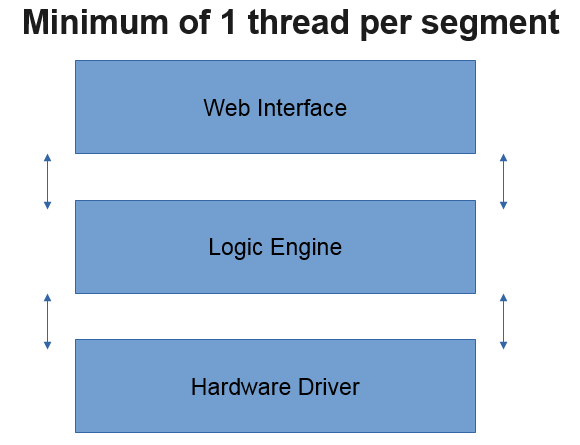
## Data description

<How data is being stored and where>

## Data dictionary

<tabular data description>

# Component Design



## Hardware driver

* + 1. Written in Python and interface with motors and accelerometer/gyroscope via I2C protocol
    2. Uses an MPU-6050 Python library to do I2C communication with the accelerometer/gyroscope
    3. Interfaces with 4 motors in 2/2 pairings for a total of 2 axes
    4. Drives the motors at a rate of 90 degrees/second (minimum) to 135 degrees/second (maximum)
    5. References changes and sends commands to the motor at a rate of 50 Hz (minimum) – 100 Hz (maximum)
    6. Tracks current position of motors and transmits this to the logic engine via socket
    7. If receives a command that would rotate a motor over 90 degrees (within 2.5% tolerance), does not execute it and sends an alert to the logic engine via socket
    8. Sends an alert to the logic engine via socket if a motor is at its rotational limit of 90 degrees (within 2.5% tolerance)

## Web Interface

* + 1. Document-Object-Model (DOM) used with front-facing HTML page connected to Javascript code
    2. Has HTML input boxes for either angle change commands (either offset or destination angles) or 3-dimensional locations to “aim” the gyroscope at
    3. Javascript used to send input to logic engine via socket
    4. Javascript used to track and display currently executing inputs to the DOM
    5. Javascript used to output current angle offsets and alerts received from hardware to the DOM
    6. Displays current angle offsets at a rate equivalent to the driver rate (50 to 100 Hz)
    7. Javascript used to enable downloading of log files from the logic engine
    8. Has an API for submitting automated commands to the logic engine
    9. Javascript used to detect and disable invalid commands (such as angle rotations over 90 degrees)

## Logic Engine

* + 1. Written in Python with both a client and server component that use sockets to communicate
    2. Python code to take angle/point commands received from the interface via socket and translate them into motions to send to the hardware driver
    3. Polls both the interface and driver via sockets for status and alerts
    4. Can accept streamed data over a socket via the web interface API
    5. Python code to support a “balancing mode” in which values from the accelerometer are read as inputs and commands are sent to the driver to level the gyroscope accordingly
    6. Python code to log all alerts and commands sent to/from the driver and web interface
    7. The web interface and the driver should send and receive commands and alerts over different ports

# Human Interface Design

## Overview

HTML page consisting of…

### 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. **…**

# Requirements Mapping Matrix

|  |  |  |
| --- | --- | --- |
| SRS Requirement # | Description | SDD Section Mapping |
| 1 | The driver will interface with 4 motors in 2/2 pairings for a total of 2 axes | 5.1.3 |
| 2 | The driver will drive the motors at a rate of 90 degrees/second (minimum) to 135 degrees/second (maximum) | 5.1.4 |
| 3 | The web interface will allow for the input of either angle change commands (either offset or destination angles) or 3-dimensional locations to “aim” the gyroscope at | 5.2.2 |
| 4 | The web interface will send messages both to and from the user in the form of current inputs, user commands, and outputs | 5.2.3, 5.2.4, 5.2.5 |
| 5 | An API must be supplied to programmatically allow for automated inputs and handling of alerts | 5.2.8 |
| 6 | The logic engine must take angles and desired points and translate them into motions to send to the driver | 5.3.2 |
| 7 | The logic engine must poll both the web interface and driver for status, including new points and alerts | 5.3.3 |
| 8 | The logic engine must accept streamed data from the web interface and send commands to the motor accordingly | 5.3.4 |
| 9 | The logic engine must support a “balancing-mode” in which the engine will read values from the accelerometer as inputs and send commands to the driver to level the gyroscope accordingly | 5.3.5 |
| 10 | The driver will reference changes and send commands to the motor at a rate of 50 Hz (minimum) – 100 Hz (maximum) | 5.1.5 |
| 11 | The driver will allow rotations of up to 90 degrees per axis. There is a 2.5% tolerance on this limit | 5.1.7 |
| 12 | The driver must track net angle changes and throw alerts to other portions of code when the rotation limit is reached | 5.1.6, 5.1.8 |
| 13 | The web interface will display current angle offsets of the device to the user at a rate equivalent to the driver rate (50-100 Hz) | 5.2.6 |
| 14 | The web interface must detect and disable invalid parameters | 5.2.9 |
| 15 | The logic engine must log all operations | 5.3.6 |
| 16 | The web interface will support downloading of log files | 5.2.7 |

# 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