System Requirements Specification

Version 1.0

**Table Of Contents**

1. **Introduction………………………………………………………………………………2**
   1. Project Overview……………………………………………………………………...2
   2. Purpose and Scope…………………………………………………………………….2
2. **Overall Description………………………………………………………………………2**
   1. Product Functions/Context…………………………………………………………….2
   2. User Characteristics…………………………………………………………………...2
   3. Assumptions…………………………………………………………………………...3
   4. Constraints…………………………………………………………………………….3
   5. Dependencies/Environment…………………………………………………………...3
3. **External Interfaces…………………………………………………………………….....3**
   1. User Interfaces………………………………………………………………………...3
   2. Network Interfaces…………………………………………………………………….3
   3. API…………………………………………………………………………………….3
   4. Protocols………………………………………………………………………………3
4. **Functional Requirements………………………………………………………………..4**
5. **Nonfunctional Requirements……………………………………………………………4**
   1. User Interface Requirements…………………………………………………………..4
   2. Safety Requirements…………………………………………………………………..5
   3. Documentation Requirements…………………………………………………………5
6. **Appendices………………………………………………………………………………..5**
7. **Introduction**
   1. Project Overview

The task for this project is to develop 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.

* 1. Purpose and Scope

In Scope:  
All development, testing, and documentation for:

* The web server/client interface to communicate with the software controlling the hardware
* The lower level software for the pi that controls the hardware
* The driver to simulate the hardware
* The networking code to handle communication between the server and client

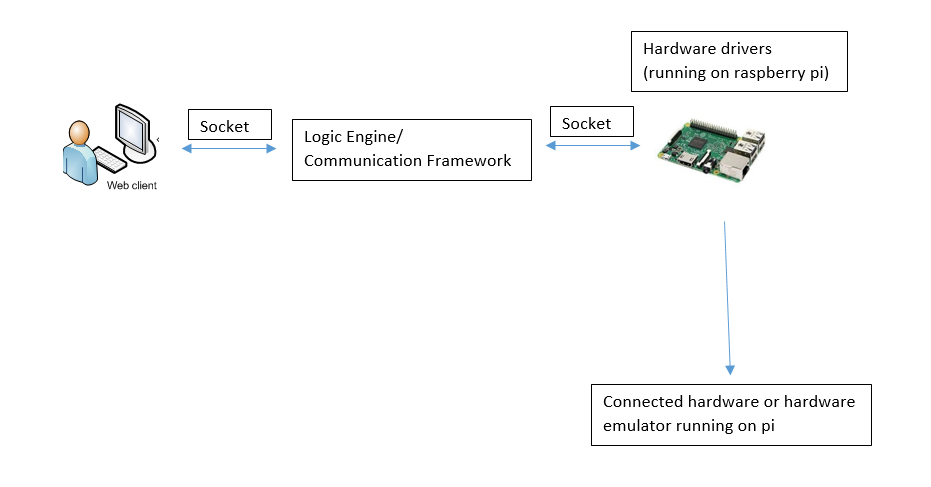
Out of Scope:

Any development or work on the hardware itself

1. **Overall Description**
   1. Product Functions/Context

The interface should load on any computer, and send commands to a server running on a raspberry pi, which in turn controls the hardware via lower level software commands. The hardware may also be emulated when access to it is restricted.

Generalized layout:



* 1. User Characteristics

The user will be the POC, who has knowledge of the hardware, knows the correct commands to make the hardware components accomplish tasks, and has enough technical knowledge to determine if technical requirements are being met correctly.

* 1. Assumptions

It is assumed that:

* The hardware will include, at a minimum, 4 stepper motors and an accelerometer
* The hardware will only occasionally be available for testing
* The user knows the limitations of the hardware
* Raspbian will be the OS on the pi
* All driver, server, client, and communication functionality can be implemented via Python code and raspberry pi libraries
  1. Constraints

Design options are constrained by the following:

* All communication between the client and server should be done via sockets, with no common read/write memory
* There is limited access to the hardware
* The software must utilize libraries already installed on the raspberry pi, which are currently unknown
* All communication between the client and server must be logged
* All QA responsibilities will be shared amongst the dev team
  1. Dependencies/Environments

The web client and interface should run on any modern laptop or desktop PC with Python 3.0 and a browser installed. The server and drivers should run on a raspberry pi running Raspbian with Python 3.0 installed. The low level drivers controlling the hardware must be developed before the client, server, and communication framework can be completed, because handling commands and alerts to and from the driver is the core of their functionality.

1. **External Interfaces**
   1. User Interfaces

A UI will be developed to run in a browser and work in conjunction with the client (written in Python) to accept input in the form of angle offsets, specific angle locations, and xyz offsets

* 1. Network Interfaces

Sockets will be used to communicate between the web client and server. All segments of the software should send and receive via different ports.

* 1. API

A ReST API must be created to allow for automated input to be sent over sockets.

* 1. Protocols

I2C will be the protocol for direct communication with the hardware.

1. **Functional Requirements**

|  |  |  |
| --- | --- | --- |
| Requirement # | Description | Priority |
| 1 | The driver will interface with 4 motors in 2/2 pairings for a total of 2 axes | 1 |
| 2 | The driver will drive the motors at a rate of 90 degrees/second (minimum) to 135 degrees/second (maximum) | 1 |
| 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 | 1 |
| 4 | The web interface will send messages both to and from the user in the form of current inputs, user commands, and outputs | 1 |
| 5 | An API must be supplied to programmatically allow for automated inputs and handling of alerts | 2 |
| 6 | The logic engine must take angles and desired points and translate them into motions to send to the driver | 1 |
| 7 | The logic engine must poll both the web interface and driver for status, including new points and alerts | 1 |
| 8 | The logic engine must accept streamed data from the web interface and send commands to the motor accordingly | 3 |
| 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 | 2 |
| 10 | The driver will reference changes and send commands to the motor at a rate of 50 Hz (minimum) – 100 Hz (maximum) | 1 |
| 11 | The driver will allow rotations of up to 90 degrees per axis. There is a 2.5% tolerance on this limit | 1 |
| 12 | The driver must track net angle changes and throw alerts to other portions of code when the rotation limit is reached | 1 |
| 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) | 1 |
| 14 | The web interface must detect and disable invalid parameters | 1 |
| 15 | The logic engine must log all operations | 2 |
| 16 | The web interface will support downloading of log files | 2 |

1. **Nonfunctional Requirements**
   1. User Interface Requirements

There must be a browser-based WebUI for the user to enter angle offsets, specific angle locations, and xyz offsets and receive alerts.

* 1. Safety Requirements

Commands that would require rotations over 90 degrees must be detected and disabled by both the web interface and the driver. The driver should also throw alerts when the rotation limit is reached.

* 1. Documentation Requirements

A system requirements specification, system design doc, statement of work, working agreement, user manual, and backlog of user stories to be completed should all be created and maintained.

1. **Appendices**