# Introduction

## 1.1 Purpose

The product being implemented is an Internet of Things Garage door opener. The purpose of this product is to create a more convenient way of opening a garage door with either a mobile or stationary device, and then being able to determine which door(s) are open at any given time.

## Scope

This product will work as both a web server and an IP relay. That will serve a thin client architecture user interface.

## 1.3 Acronyms, Abbreviations, Definitions

IoT – Internet of things

IP – Internet protocol

Thin client architecture – A server that hosts an application to the user and the server does the majority of the backend processing

UI – User interface

SDLC – Software Development Life Cycle

GPIO – General Purpose Input/Output

# General Description

## 2.1 Context

There has recently become a need for a more efficient way to close, open, and determine the status of the garage doors at the Klinghagen residence. Our main client is Ron Klinghagen. This product will do exactly this. The IoT garage door opener will have the capacity to open and close each of the three doors from anywhere that an internet connection is available.

## Product Functions

This product will have the capability to open, close, and determine the status of each garage door. The product will communicate over the world wide web with a variety of devices (i.e Mobile or Stationary).

## Constraints

### 2.3.1 Design

This product will be build using an IoT platform. This means that the device will communicate solely over the internet. The product will also be used in a thin client architecture due to its nature.

### 2.3.2 Process

Since there is no need for expedition of this product, the project will take the traditional waterfall approach to the SDLC.

## Assumptions

It is assumed that there will be a reliable internet connection available at this residence and an internet connection on the client device. It is also assumed that there will be a reliable power connection to the hardware at all times.

# 3. Specific Requirements

## 3.1 External Interface Requirements

### 3.1.1 User Interfaces

The UI will be a web application that will have a front-end written with html and CSS. The backend of this web application will use python scripts sent from the html, where a Unix bash script will be executed sending power to the GPIO pins of the device.

### 3.1.2 Hardware Interfaces

The hardware that will be used for this product will be a Raspberry Pi 2 B. This piece of hardware will act as the brain of the entire system. We will be implementing the use of its built in GPIO peripherals. The GPIO peripheral will communicate with an external relay module, that will act as a momentary switch to operate the garage door motor. The Pi will also be acting as the web server for the user interface.

### 3.1.3 Software Interfaces

The software interfaces that will be used in this product are: Python Flask and Raspbian. Python’s Flask Library will be used to act as a web server that will host the User Interface. Raspbian is the standard Raspberry Pi operating system. This is a free distribution of Unix. This will allow us to use the Bash terminal to create the backend scripts for the GPIO.

## 3.2 Functional Requirements

1. The Product must have the ability to open three garage doors

2. The Product must have the ability to connect to the internet.

3. The Product must be able to determine the status of each door at any given time.

## 3.3 Design Requirements

1. The Product must implement a simple and usable UI. That is responsive, since it will be used on devices with a variety of screen resolutions

2. The Product should be discrete after installation.

## 3.4 Quality Requirements

1. The Product must have some form of security, since it will be openly facing the internet.

2. The product should be able to maintain a reliable and fast connection to the internet.

3. The product should be able to handle multiple users on the application simultaneously.