Kingdom of Saudi Arabia
Ministry of Higher Education
Princess Nourah bint Abdulrahman University
Faculty of Computer & Informaion Science

Graduation Project I

المملكة العربية السعودية
وزارة التعليم العالي
جامعة الأميرة نورة بنت عبد الرحمن
كلية علوم الحاسب والمعلومات

مشروع تخرج ١

Online Home Automation Control System

2019-2020 Graduation Project I

ريم علي الغامدي
437004875

ساره خالد آل حسين
436006939

ضحى نضال الزعبي
436200063

عبير أحمد عزت
436200058

منى سعود الخثلان
437004005

نوف عبد الله الدعجاني
437004100

د. عبير محمود

Contents

List of Tables

List of Figures

List of Symbols & Abbreviations

Table 1: List of Symbols & Abbreviations

Symbols & Abbreviations	Meaning
UI	User Interface
API	Application programming Interface
GPIO	General Purpose Input Output
IDE	Integrated Development Environment
IoT	Internet of Things
LED	Light Emitting Diode
REST	Representational State Transfer
SDLC	Software Development Life Cycle
SQL	Structured Query Language
Raspberry Pi	low cost, credit-card sized computer[1].
Linear solenoid	type of electromagnetic actuator that converts an electrical signal into a magnetic field producing a linear motion[2].

Keywords

Table 2: Keywords

Keyword	Definition
Raspberry Pi	low cost, credit-card sized computer[1].
Linear solenoid	type of electromagnetic actuator that converts an electrical signal into a magnetic field producing a linear motion[2].

Abstract & Keywords

The aim for this project is to control light buttons, air conditioners, television or other home appliance regardless of the person's location. The methodology is simple: an android app will send controlling requests to a web server. Raspberry Pi will be getting all the new requests from the server, processing it accordingly and controlling the hardware components connected to it. Such a system will allow someone in the United States to turn the lights in their house in Saudi Arabia on. However, an active connection to the internet must be present all the time.

CHAPTER NO. 1
INTRODUCTION

1 Introduction

1.1 Problem statement & Significance

With the recent very rapid progress in technology and automation, there has become a need for remote control of almost all possible aspects of living, especially the house appliances that surround us, because of how easy it makes the modern humans life and how much it allows them to focus on their main work and be more productive instead of doing these remotely controlled tasks for themselves, and simply, of how convenient it is. Examples we have already encountered and used in our daily lives include using apps to control a cleaning robot or adjust the heating in the house or even make coffee or switch the house lights on or off. For the latter, there have been many applications that can do that, however they all work locally and there hasnt been one yet that uses the internet so it can be used remotely from outside the house to control the lights. It is necessary for an application like that to exist, as a service like this would be important for many people, like, for example, working moms who are outside the house and want to switch the lights on at a certain time to wake their children up, or pet owners who need to have UV lights switched on for their pets at certain times of the day but cant do so immediately and so on. However, the main challenge in creating a device to solve this problem is where the idea of IoT (Internet of Things) comes in; learning how to control this device through the Internet from afar, rather than being controlled by infrared rays locally as is the case with most similar applications.

1.2 Proposed Solution

The created app should enable the user, by clicking on the appropriate buttons, to control a physical apparatus that needs to be pushed to work, such as lights buttons. This will be done by designing and creating an Android application, then using a small laptop, called Raspberry Pi, to control a small piece that will

be pushed forward (on command) to switch the light on or off, the API is a web application hosted on a server.

1.2.1 Aims

At the end of this project, we intend to achieve the following aims:

- Learn how to design a mobile application using previously learned and new knowledge
- Learn how to invoke a web API and use it in our application
- Learn Python programming language to control Raspberry Pi
- Learn Flask web micro-framework

1.2.2 Goals

At the end of this project, we expect to deliver:

- An Android application with a user friendly, simple, clear interface with buttons to control a LED and linear solenoid.
- A physical apparatus composed of the Raspberry Pi connected to and controlling the piece.
- A web application following REST architecture, managing user requests and Raspberry Pi's responses.

1.3 Project Domain & Limitation

1.3.1 Domain

Although the application will be available for all kinds of users to use, we expect that the ones who would make the most use of it would be employees who have long working hours and would need to be able to remotely control appliances in their homes, especially lights.

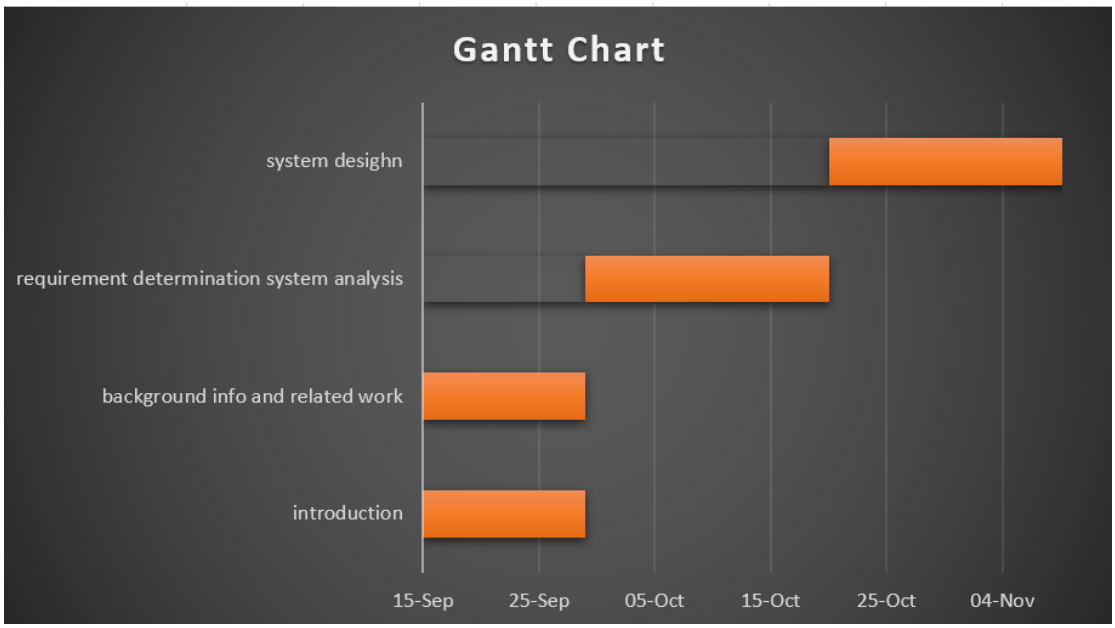
The critical piece in all of this application would be the linear solenoid actuator i.e. the small electrically controlled piece that would be placed very close to the light switch and would, on command, spring forward to press on the switch to turn it on or off.

1.3.2 Limitation

The main limitation of the application is that it will be able to control only a limited type of home appliance. Mainly things than could be pushed to work. A much more advanced application would be able to control most of the other appliances, such as controlling an Air conditioner if the owner is outside, or a timer controlled coffee maker.

1.4 Gantt Chart

Figure 1.1: Gantt Chart



CHAPTER NO. 2
BACKGROUND INFORMATION & RELATED WORK

2 Background Information & Related Work

2.1 Background Information

2.1.1 IoT

The internet of things (IoT) constitutes one of the most important technological development in the last decade. The IoT term was coined by Kevin n 1999[3]. IoT means a world-wide network of interconnected objects uniquely addressable, based on standard communication protocols[4]. in a somewhat simplified manner, we can describe IoT is the ability to connecting as many things by the internet without direct human intervention by using technologies such as cloud computing, Radio Frequency Identification (RFID), wireless communication, sensors, Internet protocol, ultra-low-power processors and others[5].

2.1.1.1 IoT Architecture

IoT Architecture includes three layers Perception layer, Network layer, and Application layer each of them has its own functionality.

- **Perception layer:** is responsible to perceive and identifying objects or things in the environment.
- **Network layer:** is responsible for receiving and transmitting data between layers.
- **Application layer:** is the interface for all previous Layers used to processed and transported data to provide services to the users[6].

2.1.1.2 IoT Applications

The Applications of the IoT are diversified and can be classified into three main categories industry, environment, and society.

- **Industry:** The importance of the industry domain can be seen in transportation, aviation, and automotive (e.g.Tesla automobile).

- **Environment:** The society Domain focused on telecommunication, smart building, home, and medical technology (e.g.connected door locks, Closed-loop (automated) insulin delivery).

- **Society:** The environment Domain focused on recycling, disaster alerting, environmental monitoring (e.g.Forest Fire Detection, Air Pollution)[7].

2.1.2 Hardware

- **Raspberry Pi:** a small general purpose computer. All hardware components will be connected to it. An active connection to the internet is needed for it to fetch data from the server[1].

- **Ubuntu Web Server:** hosts the web application. Digital Ocean severs[8] were chosen for this project.

- **LED:** since the hardware components controlled depends heavily on the user needs, this project main aim will be controlling a small LED. LED stands for light-emitting diode[9]. Basically a small light source.

- **Linear Solenoid:** once the LED works, linear solenoid will be installed for demonstrating the idea[2]. It is a small component that generates a linear motion. It will be used to press in anything, such as lights, TV remote, and coffee machine.

2.1.3 Programming Languages & Frameworks

- **Python:** raspberry pi can be controlled by either c++ or python. Python was chosen because a REST API can be made using it fast.

  - **GPIO:** a library for controlling any hardware component connected to the GPIO pins[10].

  - **Flask:** a lightweight framework to build web applications.

- **Java:** mobile application are made in a native way with either swift or java.

- **Android:** a framework for making android apps.
- **Retrofit:** type-safe HTTP client for Android and Java[11]. It will be used to send and receive commands and status from the web server.

- **PostgreSQL:** an open-source RDBMS[12]. It will be installed on the server.

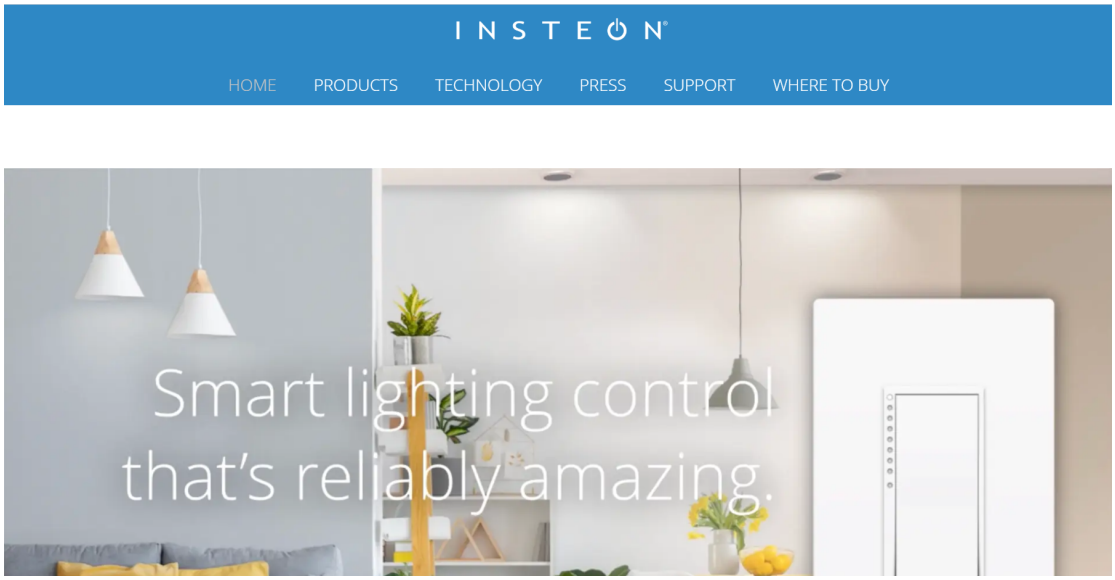
2.1.4 SDLC Model

Incremental model will be used in this project. This model is a process of software development where requirements are broken down into multiple standalone modules of software development cycle. Incremental development is done in steps from analysis design, implementation, testing / verification, maintenance[13]. The reason this model was chosen is the pieces will be installed, tested and connected to the system gradually. First a LED, then a linear solenoid and so on.

2.2 Related Work

2.2.1 Insteon - Insteon Hub

Figure 2.1: Related Work: Insteon



Insteon Hub is a simple and straightforward system that connects you to your home from any smartphone or tablet, anywhere in the world. Control Insteon light bulbs, wall switches, outlets, and thermostats at home or remotely and receive instant email or push notification alerts from motion, door and window, water leak, and smoke sensors while youre away[14].

- **Advantage:**

  1. Control Multiple Devices Simultaneously with a Basic Scene.
  2. Create Schedules to Turn Your Lights On and Off at Specific Times.
  3. Automatically Turn Lights On and Off with Sensors.
  4. Monitor Your Home with Email or Push Notification Alerts.

- **Disadvantage:**

  1. Hub setup takes a couple of minutes and a few moments per light switch, sensor.

2. Its need to connect it to power and your home's internet router so if the internet die all devices need to start over again.
3. fixed the hub take more cost than its original price.
4. There is no database save/restore. You have to recreate all the devices, scenes, schedules if its replaced.

2.2.2 Wink - Wink Hub 2

Figure 2.2: Related Work: Wink

Wink Hub 2 is the worlds first smart home hub created for the mainstream consumer. With industry-leading smart home protocol support, enhanced connectivity features, and a sleek design, Wink Hub 2 brings hundreds of products from best-in-class brands together for a simple, intuitive experience[15].

- **Advantage:**

  1. Support Different platforms such as iOS or Android.
  2. Once you've created an account, Wink has the ability to recognize the products within Wink Bright, guide you through a few simple steps, and then you're ready to go.
  3. Wink works with Cortana Microsofts voice assistant and Amazon Alexa.
  4. One Important features in wink, its can see what youre spending even before the bill arrives.

- **Disadvantage:**

  1. One major problem with the Wink 2 hub is that the device sometimes loses connectivity and must be reset in order for it to connect again.

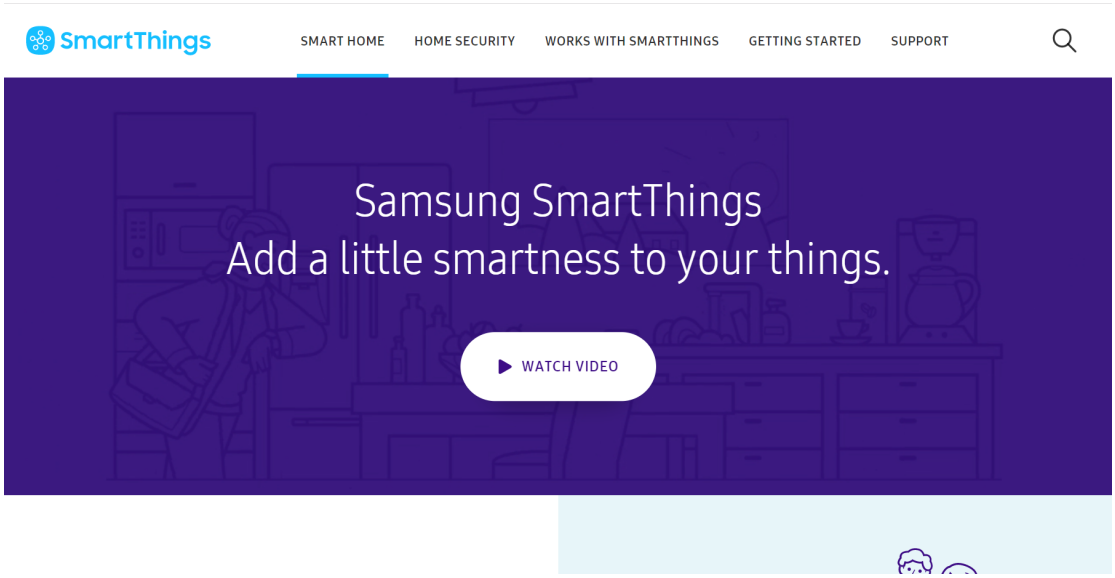
2. Wink app doesn't always let you access other devices' full features.

3. High price.

4. Takes 14 days to arrive.

2.2.3 Samsung Smart Things Hub

Figure 2.3: Related Work: Samsung



Smart things hub Connect wirelessly with a wide range of smart devices and make them work together[16].

- **Advantage:**
  1. Monitor and control connected devices in your home using a single SmartThings app for iPhone or Android.
  2. Manage connected devices in your home with SmartThings Routines for Good Morning, Goodbye, Good Night, and more.
  3. Receive alerts from connected devices when theres unexpected activity in your home.
- **Disadvantage:**
  1. Some compatible components may not work as efficiently or smoothly as you want them to, which may be inconvenient.
  2. Some users report it stops working at times.
  3. Difficult to upgrade from older hub.
  4. In US Only.

2.3 Proposed & Similar System Comparison

Table 2.1: Proposed & Similar System Comparison

	Raspberry Pi	Insteon	Wink hub 2	Samsung (smart things)
design				
Works With Wi-Fi	yes	yes	yes	yes
Price	25$, parts are very cheap	80$, expensive parts	99$, very expensive parts	70$, expensive parts
Installation & Configuration Difficulty	hard to install but doesn't takes time to reinstall and configure	easy to install and hardly takes any time setting up even if you change your home	easy to install and hardly takes any time setting up even if you change your home	easy to install and hardly takes any time setting up even if you change your home

Although similar systems already exists, our system has its own special advantages. The biggest being **hardware freedom**. In other systems, there exists a main hub receiving user command from the mobile app. So far, the ideas and implementation is identical. The previous systems require the consumer to by additional parts for it to work, such as special LED lights that needs installation or a small component controlling air conditioners. Those parts are usually limited in numbers, usage and can get very expensive fast. On the other hand, our system works with any hardware component as long as connecting it to the electrical circuit is possible.

CHAPTER NO. 3
SYSTEM ANALYSIS

3 System Analysis

3.1 Requirement Specification

3.1.1 Overview

The application we mean to create consists of three main systems: the android application which will be the user interface, the Raspberry Pi, which is the small computer where all the hardware pieces will be connected to, and the web server which will host the REST web application and be the connection between the android application and the Raspberry Pi.

The user of the system will have to have the android app installed on his mobile phone. When the app is first opened, the first activity (page in android app) displays the hardwares that are connected to the Raspberry Pi. It gets this list of hardwares from the webserver by using the GET method. When the user clicks on any of the hardware that is there, a new activity opens. In it is mentioned the name of the hardware, the status (e.g. whether it is on or off), the commands and the scheduling configuration. All of this information is obtained from the server.

Next to the commands title, there will be a small button that when clicked on will open a third activity, which gives the option of adding a new command. The command can be instantaneous (for example, switching an LED light immediately) or it can be scheduled for a later date or time. For the instantaneous command, the POST method will be used, and for scheduling the commands, the user will have the option to choose the date and time he wishes the commands to be undertaken in. Whatever the outcome of this process is, a popup message will appear to the user either confirming the success of the command the user issued, or denying it while explaining the reason for that failure.

Under the commands tab, there will be a configuration section where all the scheduled commands will appear, along with their dates and times and options

to edit them or delete them. The edit option will be done by the PUT method and the deleted option by the DELETE method. All the methods work on the data in the servers database i.e. they either add a new command (POST), edit a scheduled command (PUT), or delete an existing command (DELETE).

For the Raspberry Pi, the sequence it works according to is timed. Every 5 minutes, it puts the hardware status to the server so that it can show on the users application hardware list. Also, every 30 seconds, it checks the server for any new commands posted by the user from the android application. If there are any new ones that have to be, it updates its own local database (an action queue) according to the priorities and scheduled dates and times of the commands. This local database is organized according to the time the command was issued (i.e. the instantaneous commands are put at the front of this queue because of their precedence and the scheduled ones are put in the command order) and contains the command ID, which hardware this command was issued for, when this command was issued, and whether the command was successfully done or not, all gotten from the server by the GET method except the successfully done column, which the Raspberry edits according to the hardware.

Whatever the result of the command was, the Raspberry posts the response of the command to the server. The android application gets this response from the server every 5 or 10 minutes, depending on the users choice. The response is displayed as a push notification in the users mobile phone. Depending on this response, the status and configuration information in the app will be updated to reflect the success or failure of the command response. Finally, it is important to note that any new hardware or configuration added to or connected to the Raspberry will have its information posted to the server by the Raspberry computer, where the user can view it then as soon as he opens his application to the first activity.

3.1.1.1 Input

The user command issued using the Android client is the main input. Each command consist of the following:

- chosen hardware.
- configuration wanted.
- optional scheduling information.

The **hardware** is the physical component connected to raspberry pi. Each hardware has a set of possible states that it can be in. Those states are called **configuration**. The **schedule** indicates the time of day and days of week the user might want the command to run at.

3.1.1.2 Output

- a respond

The raspberry pi issues a **respond** indicating whether the command has been successfully done with an optional message. This respond is saved in the webserver, which in turn is read by the android client periodically.

3.2 Requirement Analysis

3.2.1 Software Requirements

- **Languages**
  - Java
  - Python
  - SQL
- **Frameworks & libraries**
  - Android
  - Retrofit
  - GPIO
  - Flask
  - SQLAlchemy
- **IDE**
  - Android studio
  - Pycharm
- **Databases**
  - Postgresql
  - Sqlite
- **Web Server**
  - Nginx
  - uWsgi

3.2.2 Hardware Requirements

- **Raspberry Pi**
  - Raspberry Pi 3 B+.
  - a minimum of 2 GB of RAM.
  - a minimum of 10 GB space in SD card.
  - a monitor, a keyboard and a mouse, alternatively SSH connection could be established.
  - internet connection, either via Wi-Fi or Ethernet cable.
  - breadboard, cables, and resistors for circuit.
  - RGB LED, solenoid, or any other hardware components satisfying user needs.
- **Web Server**
  - Ubuntu 16.04+ web server, we chose digital ocean's.
  - Minimum of 1GB of RAM.
  - Minimum of 10GB of available space.
- **Android mobile phone**

3.2.3 Functional Requirements

3.2.3.1 Android Client's Functionalities:

3.2.3.1.1. Android client should get system hardware list from webserver.

3.2.3.1.2. Android client should get scheduled commands from webserver.

3.2.3.1.3. Android client shall be able to submit a new a command, might be scheduled, to webserver.

3.2.3.1.4. Android client shall be able to delete a scheduled command from webserver.

3.2.3.1.5. Android client shall be able to edit a scheduled command from webserver.

3.2.3.1.6. Android client shall get responses from webserver automatically.

3.2.3.2 Raspberry pi's Functionalities:

3.2.3.2.1. Raspberry pi should get command each 30 sec.

3.2.3.2.2. Raspberry pi shall be able to edit local queue.

3.2.3.2.3. Raspberry pi shall be able to edit hardware's current configuration(status).

3.2.3.2.4. Raspberry pi shall be able to submit a command response to server.

3.2.3.2.5. Raspberry pi shall be able to update server's hardware and configuration lists.

3.2.4 Non-Functional Requirements

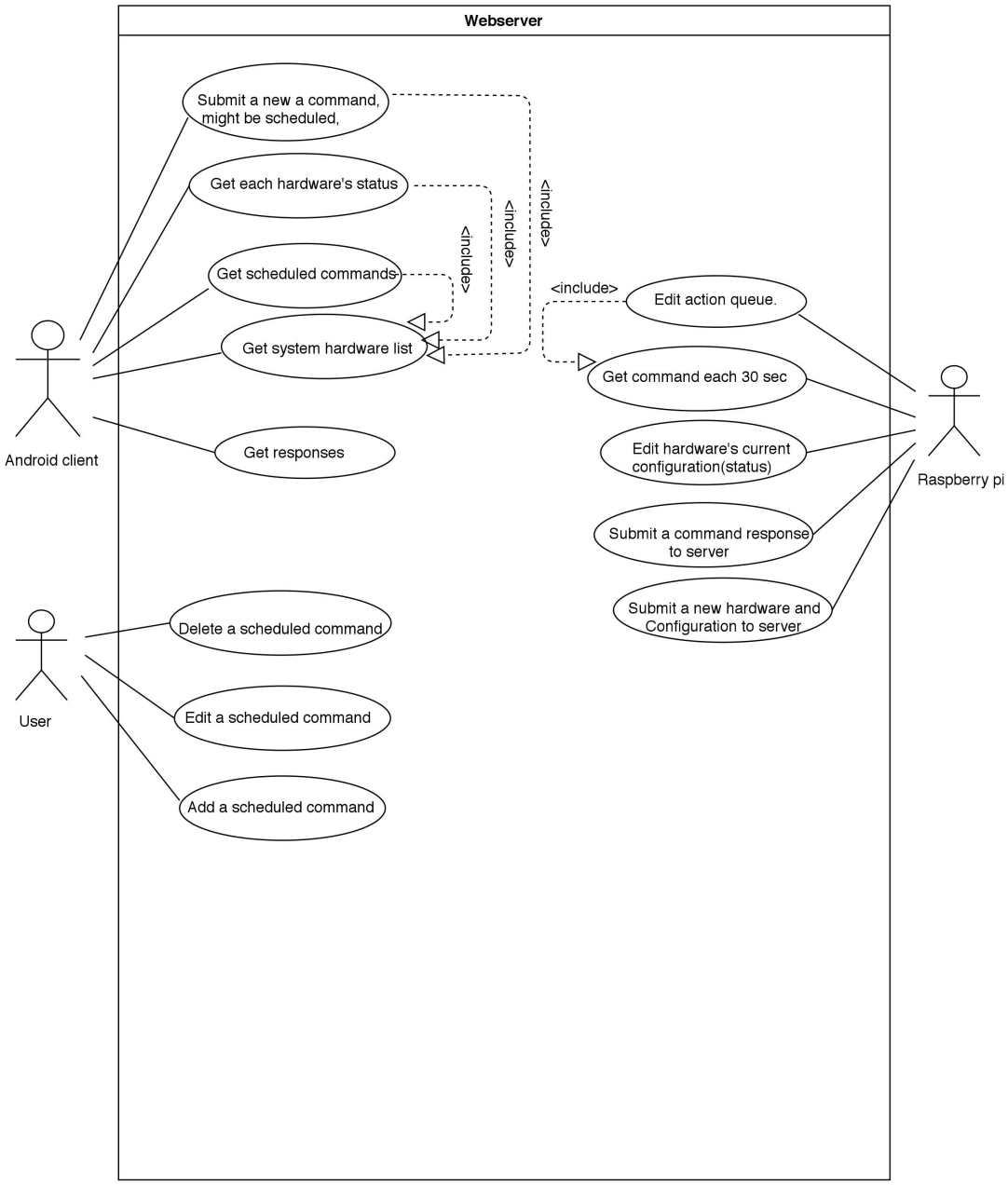
Table 3.1: Non-functional requirements

Requirement	Description
Availability	The system shall not be shut down for maintenance for more than 1 minute.
Usability	The user shall be able to learn the most important 5 functions of application in 2 hours.
Verifiability	When a new version of the main system is released, it shall be possible to upgrade to it from any previous version.
Performance	Any interface between a user and the automated system shall have a maximum response time of three seconds.
Flexibility	Application shall be made of multiple languages. So, user shall be able to nominate their preferred language when entering their personal information.

3.2.5 Structured Diagrams

3.2.5.1 Use Case Diagram

Figure 3.1: Use case diagram



3.2.5.2 Use Case Scenarios

3.2.5.3 Flowchart Diagram

3.2.5.4 Entity Relationship Diagram

In our system, there are two databases: the *system database* stored in the web server, and *the queue*, a local database for the raspberry pi. To make understanding the databases' easier, we represented it using entity relationship diagram. It is a graphical representation that demonstrates the relationship between concept, people, places, objects or events inside a system. The main components are the entities, relationship and cardinality. Entities are concepts or object that need their data stored. Cardinality defines that relationship in terms of numbers[17].

- **System database**: This database is the main database. It will be stored in the web server and gets accessed by the android client and the raspberry pi. There are 5 main entities:

  1. **Hardware:** it will store information related to the physical components to raspberry pi. Such as LED lights, linear solenoid or an infrared controller.
  2. **Configuration:** hardwares can have different states. For example, a LED light could be on or off. An RGB LED can be ON on a certain color, of off. Configurations save the possible states a hardware can be in.
  3. **Command:** users can change raspberry pi's hardwares to be in a certain configuration. These commands are stored here. The users requests could either be instant, or scheduled.
  4. **Schedule:** since commands can be scheduled, those data should be saved here. The user can choose the days and time a command fires.
  5. **Respond:** when raspberry pi finish executing a command, it issues a respond.

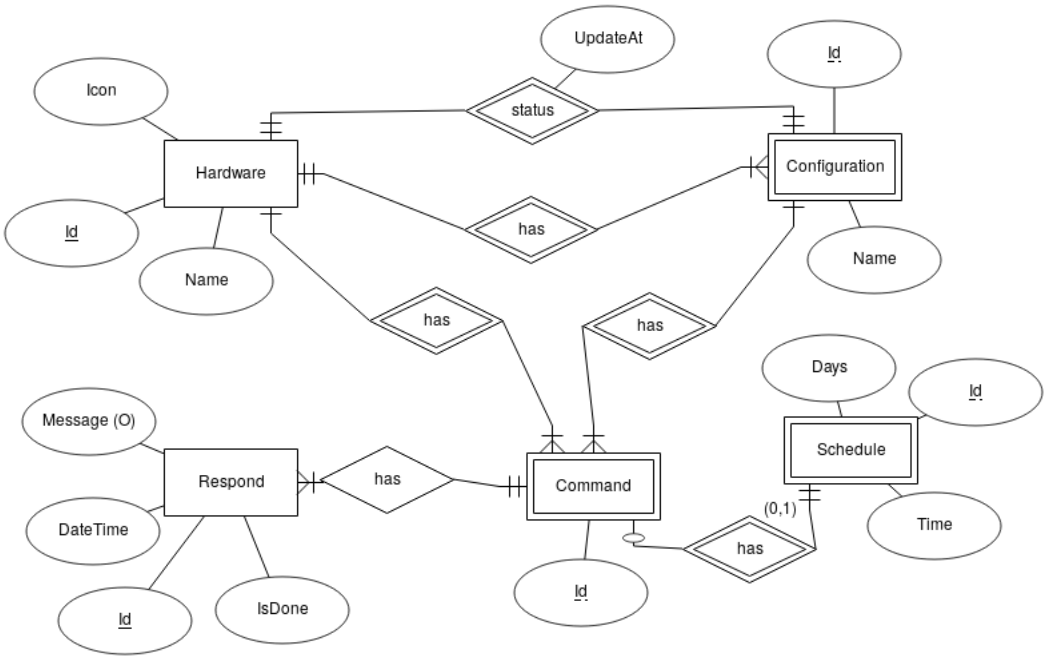
- **Local Queue**: This database is stored locally in raspberry pi. After raspberry gets user commands from server, it process them and order them based

on their execution time.

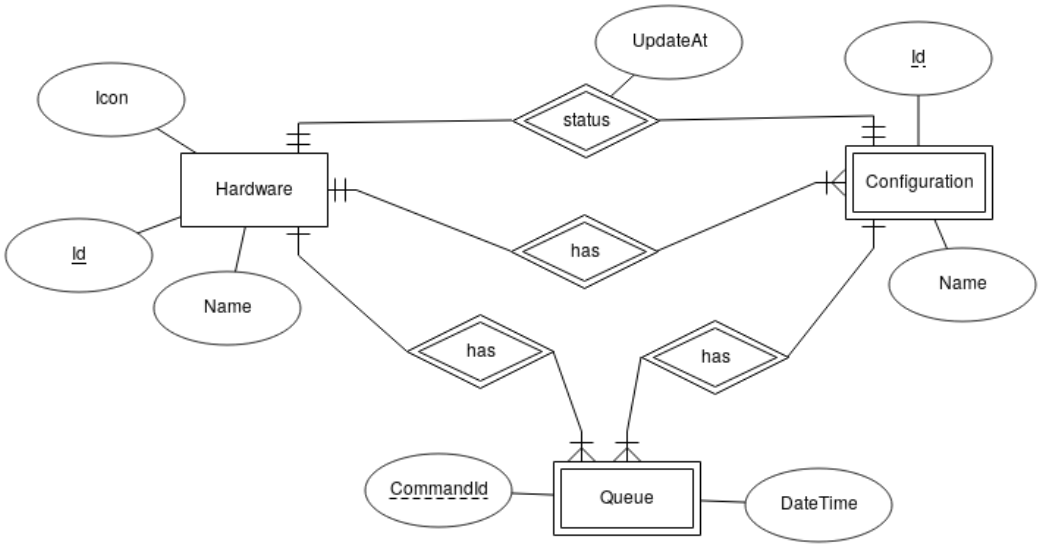
1. **Hardware:** similar to the server's hardware list, this entity store the hardware connected. The system database gets data related to hardware from this entity
2. **Configuration:** this table stores the possible configuration. When the table is edited, the server database is updated.
3. **Queue:** only the commands to be executed are stored here, once a command is executed, it gets deleted from the queue. This entity contains processed commands. i.e each queue stores the exact date and time a command will be executed instead of the schedule information.

Figure 3.2 shows the Entity-relationship diagram.

Figure 3.2: Entity-relationship diagrams



(a) System database



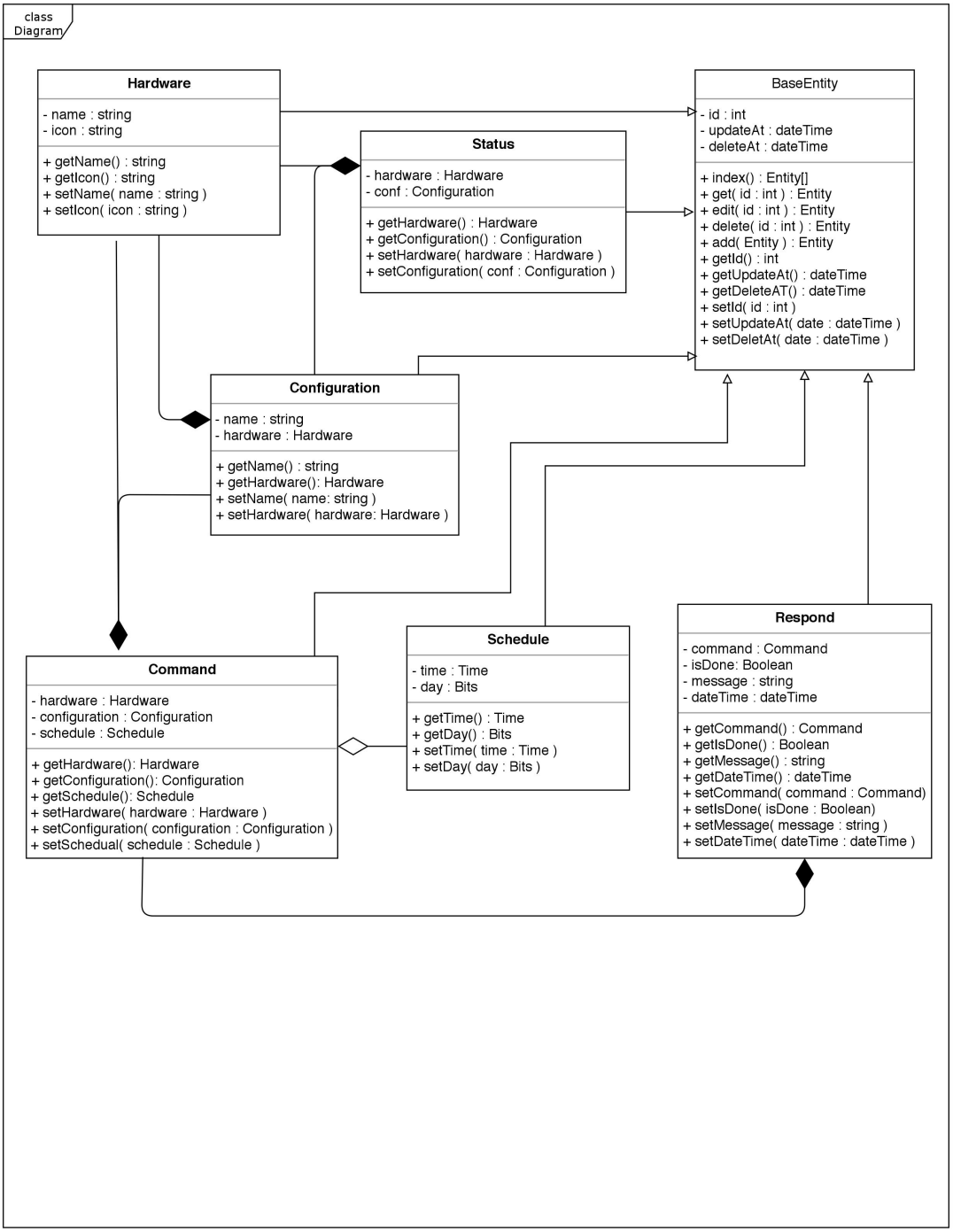
(b) Local queue

3.2.6 Object-Oriented Diagrams

3.2.6.1 Sequence Diagrams

3.2.6.2 Class Diagram

Figure 3.3: Class diagram



References

[1] "What's raspberry pi?." https://www.raspberrypi.org/help/what-%20is-a-raspberry-pi/.

[2] "Linear solenoid actuator." https://www.electronics-tutorials.ws/io/io_6.html.

[3] M. G. Samaila, M. Neto, D. A. Fernandes, M. M. Freire, and P. R. Inácio, "Challenges of securing internet of things devices: A survey," *Security and Privacy*, vol. 1, no. 2, p. e20, 2018.

[4] D. INFSO, "Networked enterprise & rfid infso g. 2 micro & nanosystems," *Co-operation with the Working Group RFID of the ETP EPOSS, Internet of Things in*, vol. 2020, 4.

[5] F. Samie, L. Bauer, and J. Henkel, "Iot technologies for embedded computing: A survey," in *Proceedings of the Eleventh IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, p. 8, ACM, 2016.

[6] M. R. Abdmeziem, D. Tandjaoui, and I. Romdhani, "Architecting the internet of things: state of the art," in *Robots and Sensor Clouds*, pp. 55–75, Springer, 2016.

[7] R. Porkodi and V. Bhuvaneswari, "The internet of things (iot) applications and communication enabling technology standards: An overview," in *2014 International Conference on Intelligent Computing Applications*, pp. 324–329, IEEE, 2014.

[8] "Droplets on digitalocean - more than just virtual machines." https://www.digitalocean.com/products/droplets/.

[9] "How light emitting diodes work — howstuffworks." https://electronics.howstuffworks.com/led.htm.

[10] "Gpio - raspberry pi documentation." https://www.raspberrypi.org/documentation/usage/gpio/.

[11] "Retrofit." https://square.github.io/retrofit/.

[12] "Postgresql: The world's most advanced open source database." https://www.postgresql.org/.

[13] "Incremental model in sdlc: Use, advantage & disadvantage." https://www.guru99.com/what-is-incremental-model-in-sdlc-advantages-disadvantages.html.

[14] "Insteon." https://www.insteon.com/.

[15] "Wink — about us." https://www.wink.com/about/.

[16] "Smartthings. add a little smartness to your things.." https://www.smartthings.com/.

[17] T. M. Connolly, I. Jacobson, and C. E. Beg, *Database Systems: A Practical Approach to Design, Implementation, and Management*. Pearson, 2005.