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

Graduation Project II

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

مشروع تخرج ٢

ONLINE HOME AUTOMATION CONTROL SYSTEM

Reem AlGhamdi, Abeer Ezzat
Nouf AlDajani, Mona AlKhathlan
Doha AlZouhbi, Sarah AlHussain

Supervisor: Dr. Abeer Mahmoud

A Graduation Project Scientific Paper Submitted to
College of Computer and Information Sciences at PNU
in Partial Fulfillment of the Requirements for the
Degree of
Bachelor of Science
in
Computer Sciences

CCIS, PNU
Riyadh, KSA
1440 - 1441H

Abstract

With the recent very rapid progress in technology and automation, and towards easier daily life tasks, there has become a need for remote control of almost all possible aspects of living. Some examples are already available. However, they lack flexibility, which restricts the user to a certain set of hardware with limited variety. The aim of 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 on the lights in their house in Saudi Arabia. However, an active connection to the internet must be present all the time.

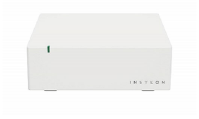
Keywords: Internet of Things (IoT), raspberry pi, cloud computing, home control, smart home.

1 Introduction

2 Problem Statement

3 Related Works

Table 1 shows comparison between similar existing system and the proposed system. *Better quality can be found at Appendix A*

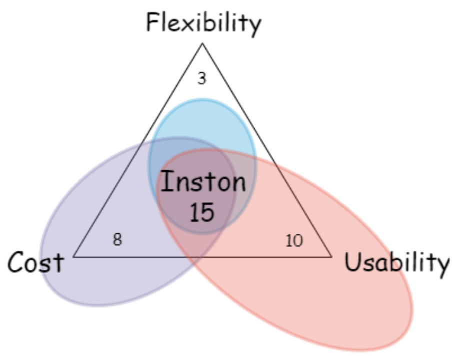
	Raspberry Pi	Insteon	Wink hub 2	Samsung (smart things)
design				
Hub Price	25$	80$	99$	70$
Hardware Price	parts are very cheap	expensive parts	very expensive parts	expensive parts
Flexibility	any hardware that can be physically connected	42 hardwares of basic variety[1]	102 hardwares of different variety[2]	only 6 hardwares support [3]
Installation & Configuration Difficulty	require one expert user. Easy control for all other users via mobile app	easy to install and control hardwares with mobile app	easy to install and control hardwares with mobile app	easy to install and control hardwares with mobile app

Tab. 1: Proposed & Similar System Comparison

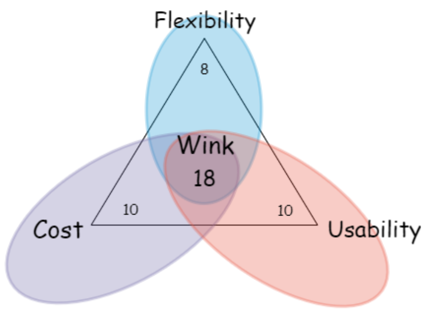
Although similar systems already exists, our system has its own special advantages. The biggest being **hardware freedom**. In other systems, there exists a main hub receiving user command from the mobile app. So far, the ideas and implementation is identical. The previous systems require the consumer 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.

However, this great flexibility comes with a great sacrifice: the steep learning curve. To keep the hardware prices as low as possible while maintaining high flexibility requires the buyer household to have one expert user. For example, in a family of four, all can control the AC. However, one of them must be an expert who can configure the hardware controlling the AC and adding it to raspberry pi manually. Of course, making it easier is possible, but it will result in a higher component cost and less flexibility as hardware components would need to be bought from our company (future implementation) in comparison to buying it from any cheap electronic store (current proposed solution).

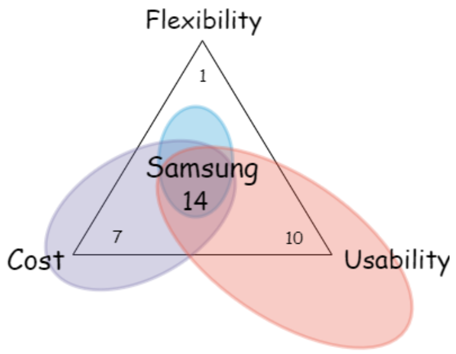
figure 1 shows this relationship between flexibility, cost and usability for the three systems, our proposed solution and the merchant edition(future work).



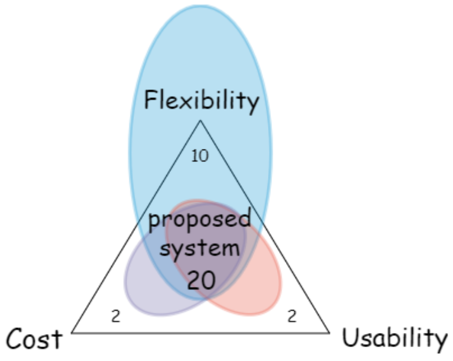
(a) Insteon



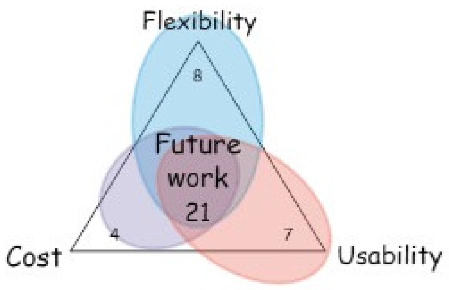
(b) Wink



(c) Samsung



(d) Proposed System



(e) Future Work

Fig. 1: Relationship between cost, flexibility and usability

4 System Design and Analysis

Incremental model was 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[4]. The reason this model was chosen is the pieces will be installed, tested and connected to the system gradually.

System requirements were gathered and selected after discussing the proposed idea thoroughly. Top-down approach was used, in which we discussed the general goal of this project, then broke it down even further to smaller manageable units. *Non-Functional Requirements* are shown in *Table 2*. On the other hand, *Functional Requirements* are realized through many diagrams. Each showing one side of the system. *See Appendix B for more diagrams.*

4.1 Non-Functional Requirements

Requirement	**Description**
Availability	The system shall not be shut down for maintenance for more than 1 minute.
Usability	Except the expert user, all other user of the system shall be able to use the system immediately without much training
Verifiability	The system shall check the user identity.
Performance and Delay	The raspberry pi shall read user commands each 30 seconds to avoid delay or overhead in the system
Flexibility	Any hardware component, such as linear solenoid or an infra-red controller, can be added to the system as long as it can be connected to the electric circuit.
Security	Each user can only communicate with assigned raspberry pi. In a similar manner, raspberry pi only reads commands meant for it and not for other ones.
Efficiency	The web server cleans the database and deletes any unneeded rows routinely.

Tab. 2: Non-functional requirements

Table 2 shows the non-functional requirements for the system and their description. The most important requirement is flexibility, as it is the core feature differentiating the proposed system from other systems. Next comes security and verifiability. They guarantee that no intruders are going to interrupt the system even if they communicated with the REST API directly without the android app because they won't be authorized -or authenticated- to enter the system. The other requirements are manly for performance and ease of use.

4.2 System Architecture

The proposed project has 4 tiers. The Android App, the web app and finally the raspberry pi script and finally the system database. The android app works as the user interface. The raspberry pi app is the one responsible for controlling the actual hardwares. The web app works as a medium for managing user requests and raspberry pi's response. Finally, the database that stores these data. .

Figure 2 shows the four-tier architecture explained above. In our system, the best architecture to use is four-tier architecture because of it is flexibility and scalability to add multiple services. Also, it adopts users experiences that are fast, responsive, and tailored unique needs. Furthermore, multi-tier architecture also reduces traffic on the network[5].

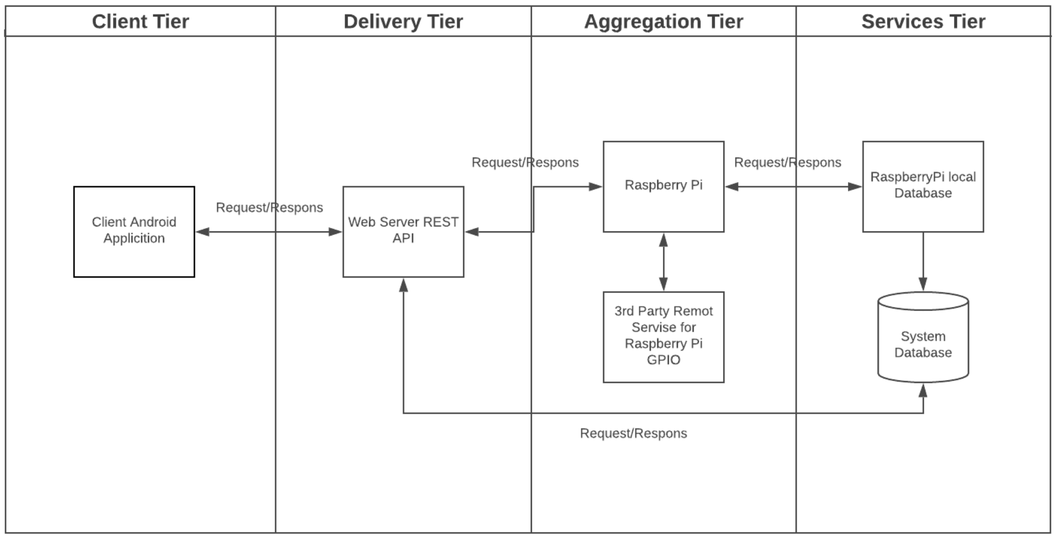
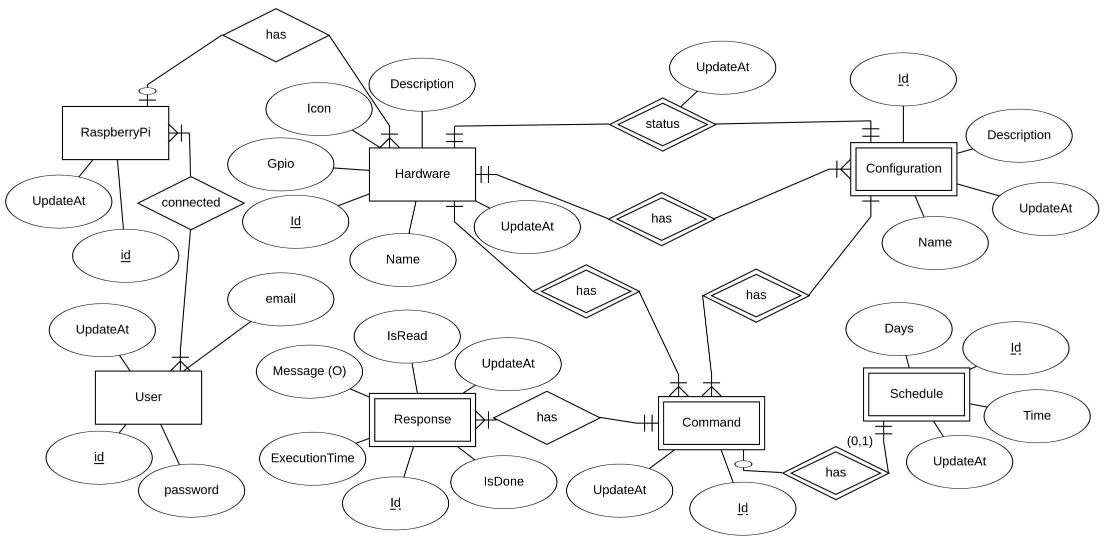


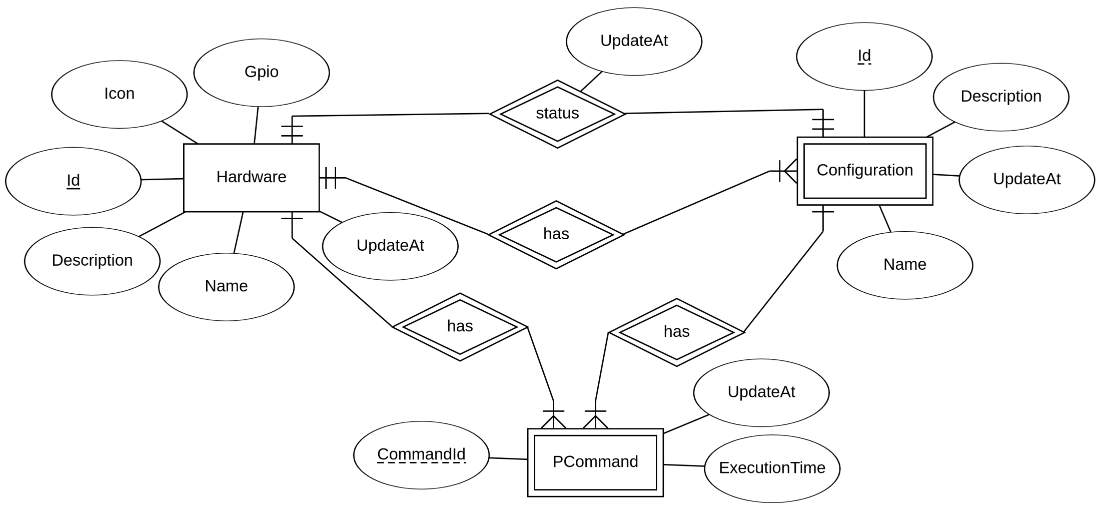
Fig. 2: Architecture Diagram

4.3 Entity Relationship Diagram

Figure 3 shows the Entity-relationship diagram.



(a) System database



(b) Local queue

Fig. 3: Entity-relationship diagrams

5 Results and Future Works

The following are the main input and output screens

5.1 Login Screen

First, the user is required to login or register before using the system. *Figure 4* shows the login interface.

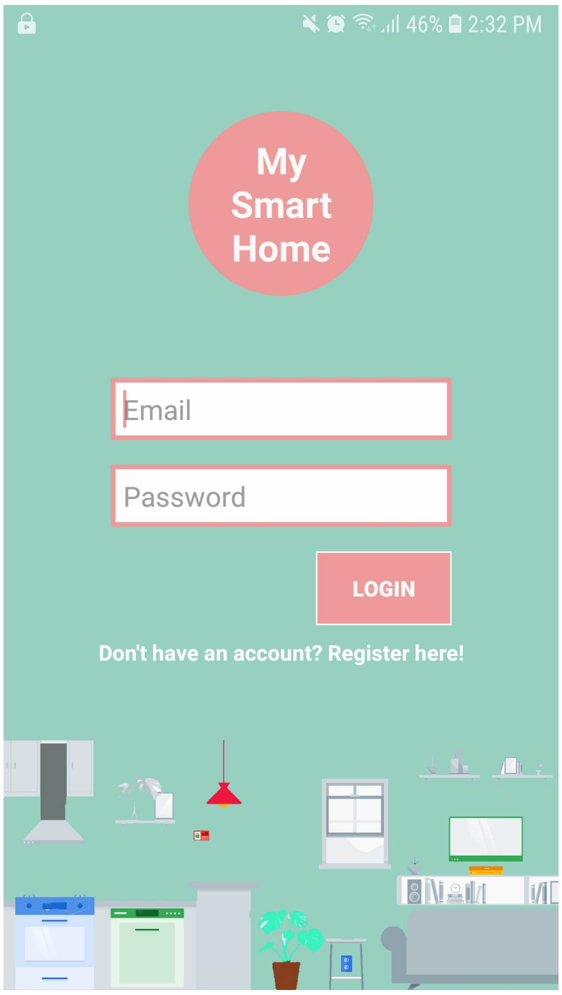
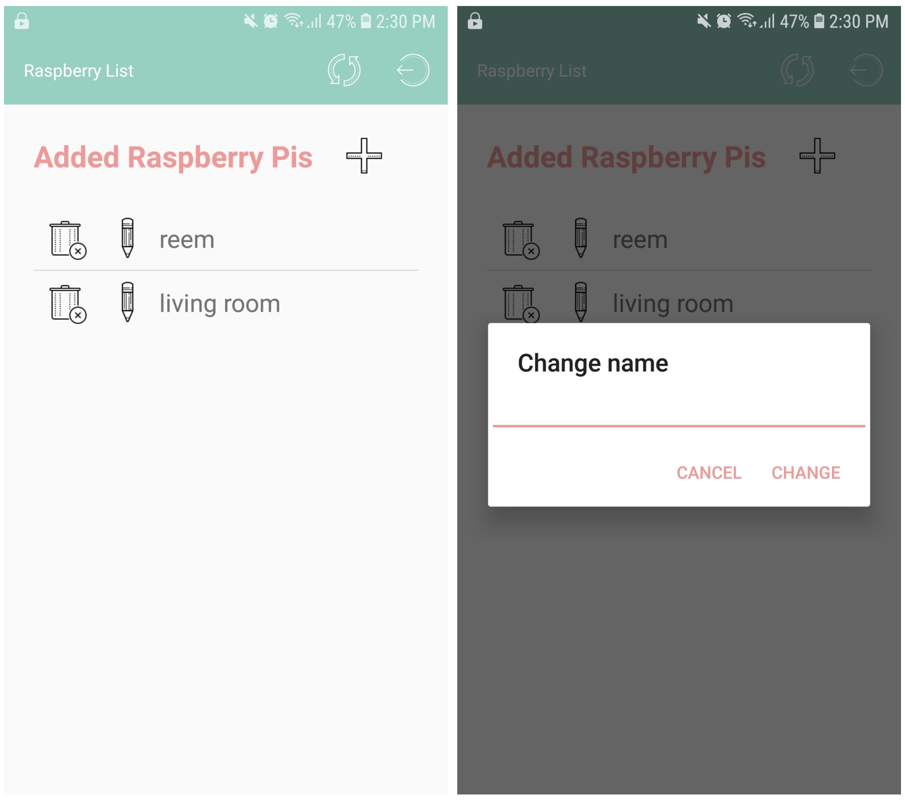


Fig. 4: I/O: Login Screen

5.2 Raspberry Pi Screen

After successful login, the user is redirected to the raspberry pi's list of connected devices. *Figure* 5 shows the Raspberry Pi interfaces.

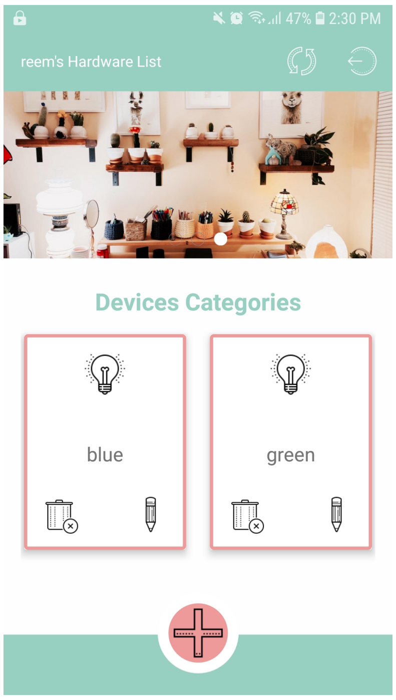


(a) I/O: Raspberry Pi List (b) I/O: Raspberry Pi Edit

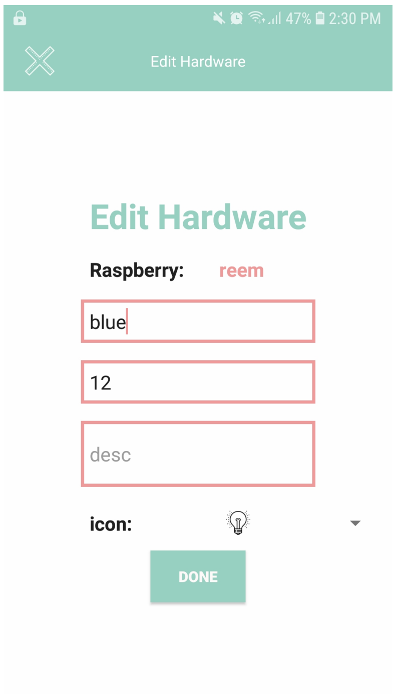
Fig. 5: I/O: Raspberry Pi Screens

5.3 Hardware Screen

After clicking on one of the raspberry pi's, the hardware list appears. *Figure 6* shows the Hardware interfaces.



(a) I/O: Hardware List



Edit Hardware

Raspberry: reem

blue

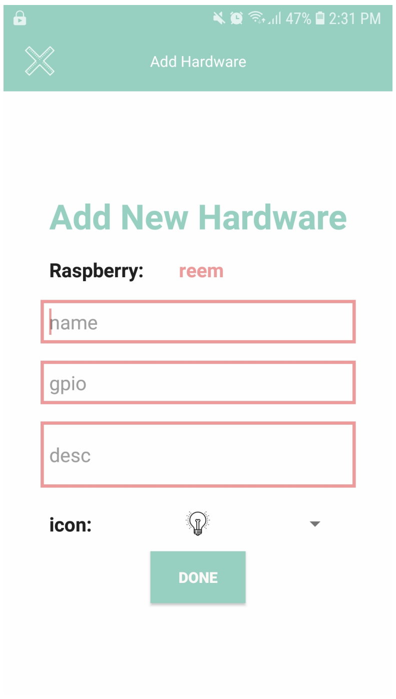
12

desc

icon:

DONE

(b) I/O: Hardware Edit



Add New Hardware

Raspberry: reem

name

gpio

desc

icon:

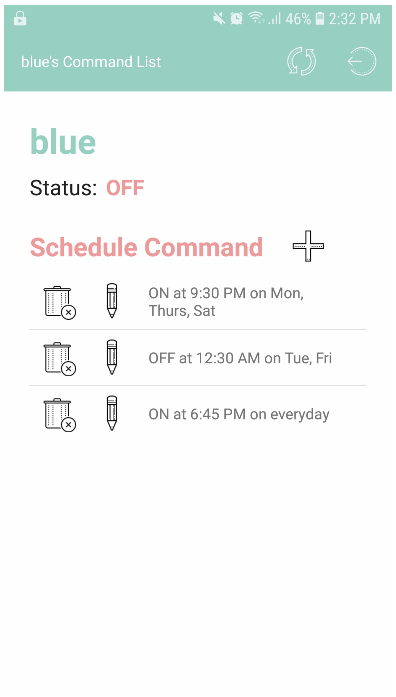
DONE

(c) I/O: Hardware Create

Fig. 6: I/O: Hardware Screens

5.4 Command & Schedule Screen

For each hardware, a user can create a command and see the command list. *Figure 7* shows the Command and Schedule interfaces.


blue's Command List

blue

Status: OFF

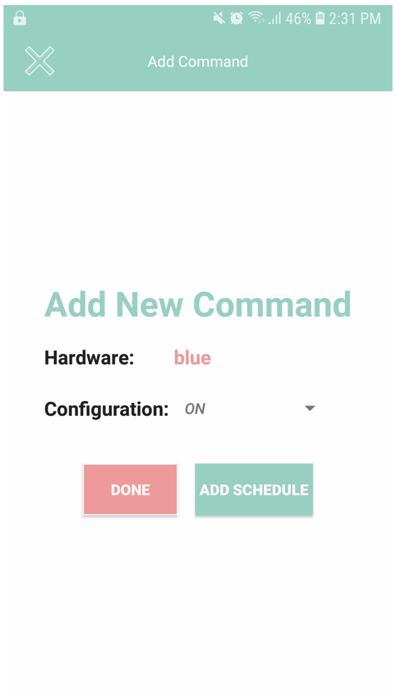
Schedule Command

ON at 9:30 PM on Mon, Thurs, Sat

OFF at 12:30 AM on Tue, Fri

ON at 6:45 PM on everyday

(a) I/O: Command List

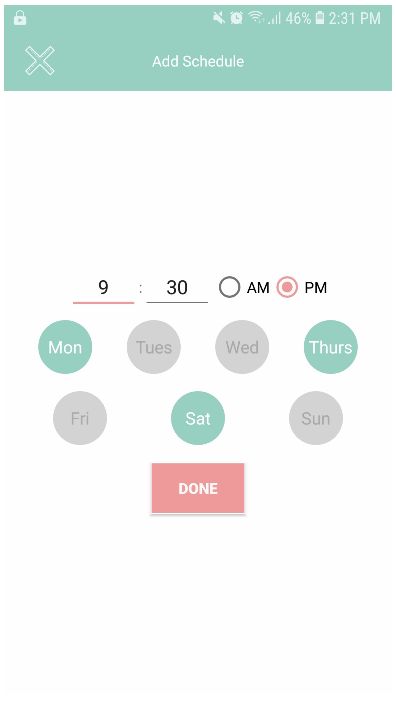

Add Command

Add New Command

Hardware: blue

Configuration: ON

(b) I/O: Command Create


Add Schedule

9 : 30 AM [✓ PM]

(c) I/O: Schedule Create

Fig. 7: I/O: Command Screens

5.5 Future Work

As with every system, there are definitely improvements and enhancements to be made. Besides what was mentioned in *Section 3*, more configuration can be added to hardwares, like multiple LED colors or different fan speeds. Also, making it wireless will further reduce its cost and remove the hassle of wires connecting, maintenance and wires going bad.

6 Conclusion

In summary, this project aimed to create a system that controls electronic household items from afar. In short, an android application was used as an interface for the user to issue control commands, which are in turn sent to and stored in the server. They are gotten from the server by a small computer (Raspberry Pi) that controls an electric circuit, to which the desired household items are connected and manipulated as per the commands. The project is a direct application of the concept of Internet of things and automated house systems.

References

[1] "All products - insteon." `https://www.insteon.com/all-products`.

[2] "Wink — buy and view smart home products." `https://www.wink.com/products/`.

[3] "Samsung smartthings - for your connected smart home — samsung uk." `https://www.samsung.com/uk/smartthings/#products`.

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

[5] P. Nommensen, "Its time to move to a fourtier application architecture." `https://www.nginx.com/blog/time-to-move-to-a-four-tier-application-architecture/`, February 2015.

Appendices

A Similar systems hub design

(a) Insteon

(b) Wink

(c) Samsung

(d) Raspberry Pi

Fig. 8: Similar systems: hub design

B Diagrams

B.1 Use-Case Diagram

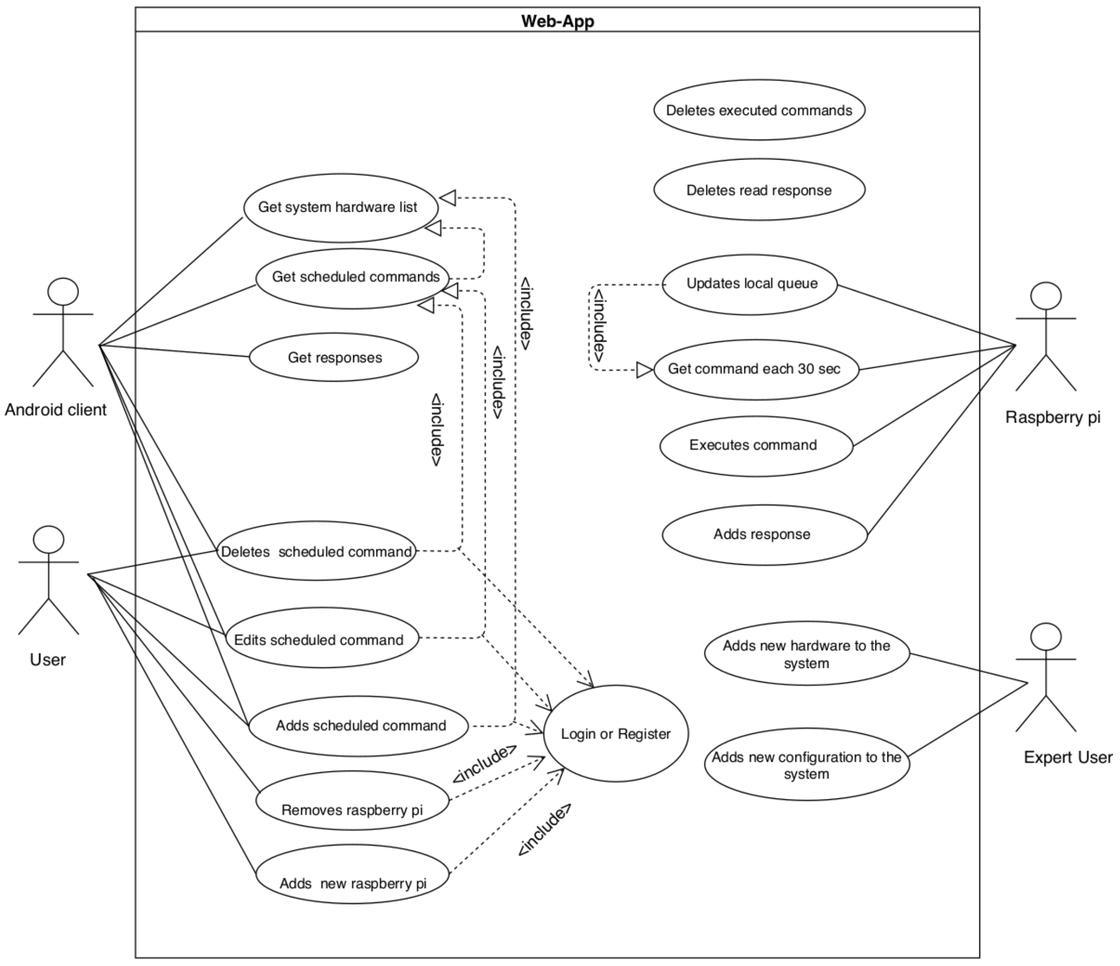


Fig. 9: Use case diagram

B.2 Flowchart Diagram

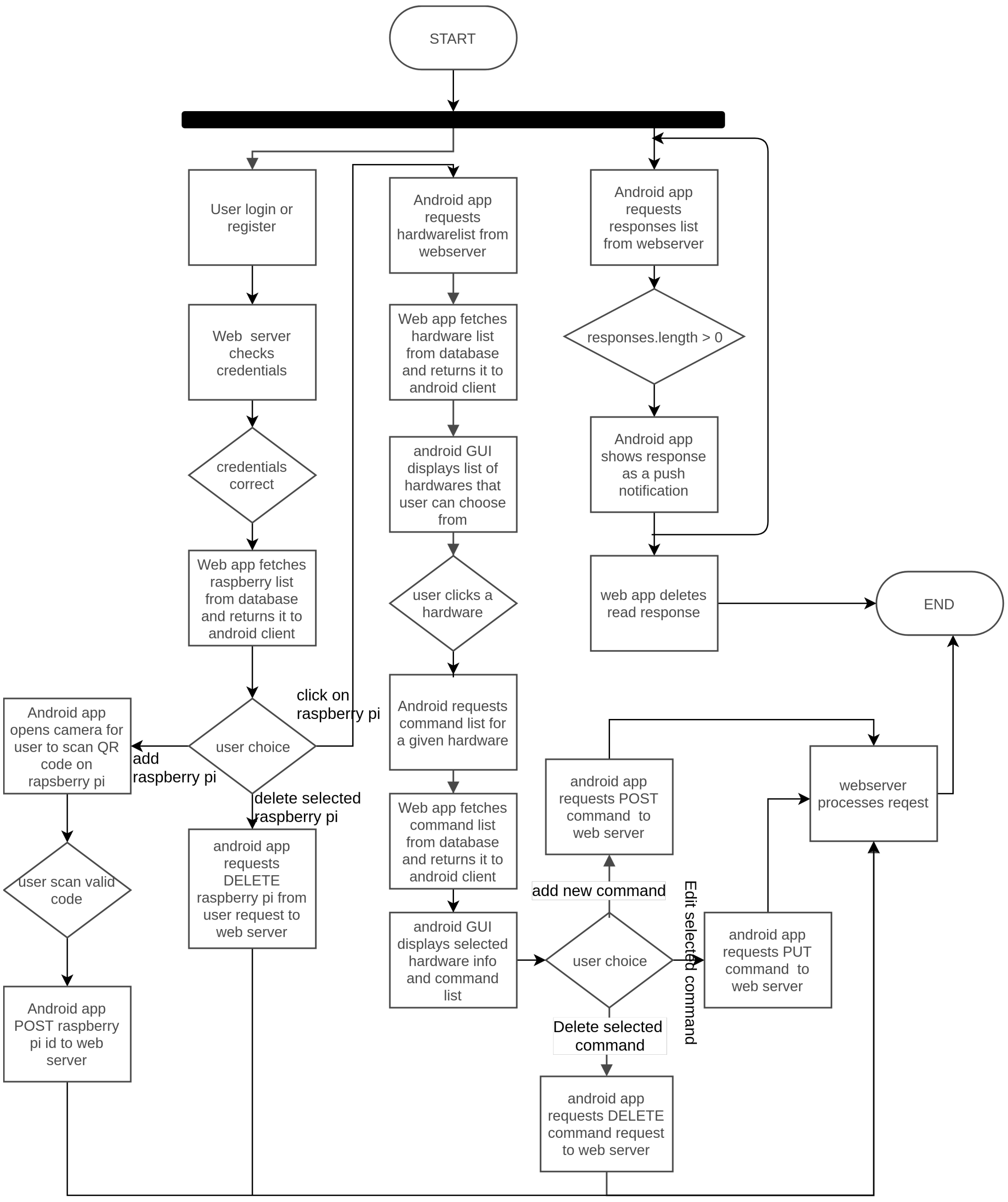

START
User login or register
Web server checks credentials
credentials correct
Web app fetches raspberry list from database and returns it to android client
user choice
click on raspberry pi
add raspberry pi
delete selected raspberry pi
Android app opens camera for user to scan QR code on rapsberry pi
user scan valid code
Android app POST raspberry pi id to web server
android app requests DELETE raspberry pi from user request to web server
Android app requests hardwarelist from webserver
Web app fetches hardware list from database and returns it to android client
android GUI displays list of hardwares that user can choose from
user clicks a hardware
Android requests command list for a given hardware
Web app fetches command list from database and returns it to android client
android GUI displays selected hardware info and command list
user choice
add new command
Edit selected command
Delete selected command
android app requests POST command to web server
android app requests PUT command to web server
android app requests DELETE command request to web server
webserver processes reqest
Android app requests responses list from webserver
responses.length > 0
Android app shows response as a push notification
web app deletes read response
END

Fig. 10: Flow chart - Android app

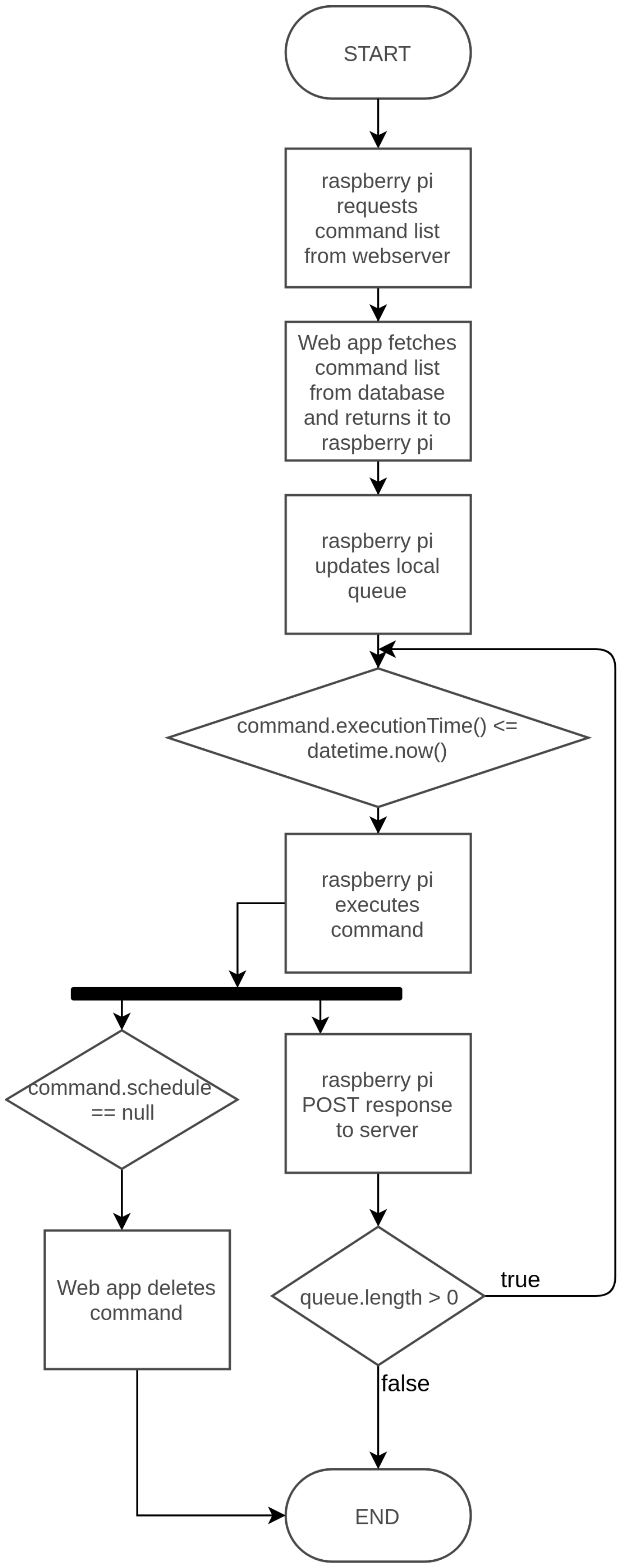

START
raspberry pi requests command list from webserver
Web app fetches command list from database and returns it to raspberry pi
raspberry pi updates local queue
command.executionTime() <= datetime.now()
raspberry pi executes command
command.schedule == null
raspberry pi POST response to server
Web app deletes command
queue.length > 0
true
false
END

Fig. 11: Flow chart - Raspberry pi