People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research

University M'Hamed BOUGARA – Boumerdes

Institute of Electrical and Electronic Engineering

Department of Electronics

Final Year Project Report Presented in Partial Fulfilment of the Requirements for the Degree of

MASTER

In **Electrical and Electronic Engineering**

Option: **Computer Engineering**

Title:

Home Automation System using Raspberry Pi

Presented by:

- **Abdelmounaim BOUATTIT.**

Supervisor:

Mr. Ad Mohammed SAHNOUN.

Registration Number:...../2016

Acknowledgement

First and foremost, we would like to take this opportunity to express my eternal gratitude to my supervisor Mr. Ad Mohammed SAHNOUN, for his assistance in guiding me throughout the evaluation of this report and for providing me with the facilities that made this work possible.

Abstract

A growing topic in modern society is the concept of smart homes, which consist of smart automated devices, such as home appliances, that interact with humans and with each other. This work presents the overall design of Home Automation System (HAS) with flexible, low cost system to remote control home appliances. The fundamental control system uses Android and Web Application to provide to the user the full monitoring and controlling capabilities over his home and its devices. Both of android and web application communicates with a common server, which is able to serve them simultaneously and provide data in real time. Raspberry pi embedded system is selected in this work because of its ability of running the given server, allowing the connection of different devices and its intuitive interface.

Table of Contents

List of Figures

Introduction

General Introduction

The rapid development of technology, the continuous cost reduction of electronic devices as well as their small size leads increasingly to using them for the convenience of everyday life of modern man. Specifically, to control devices in your home from anywhere in the world.

Home automation [1] is a method of controlling home appliances automatically for the convenience of users. This technology makes life easier for the user, and saves energy by utilizing devices according to strict requirements. It uses a combination of hardware and software technologies that enable control and management over appliances and devices within a home. A home with an automation system is also known as a "smart home".

A home automation system can involve switching off electrical appliances like air-conditioners or refrigerators when a desired temperature has been reached, then switching on again when the temperature has crossed a certain value. A home automation system can also be used to secure a house this includes your alarm system, and all of the doors, windows, locks, smoke detectors, surveillance cameras and any other sensors that are linked to it.

Home automation can be extended to building automation system.

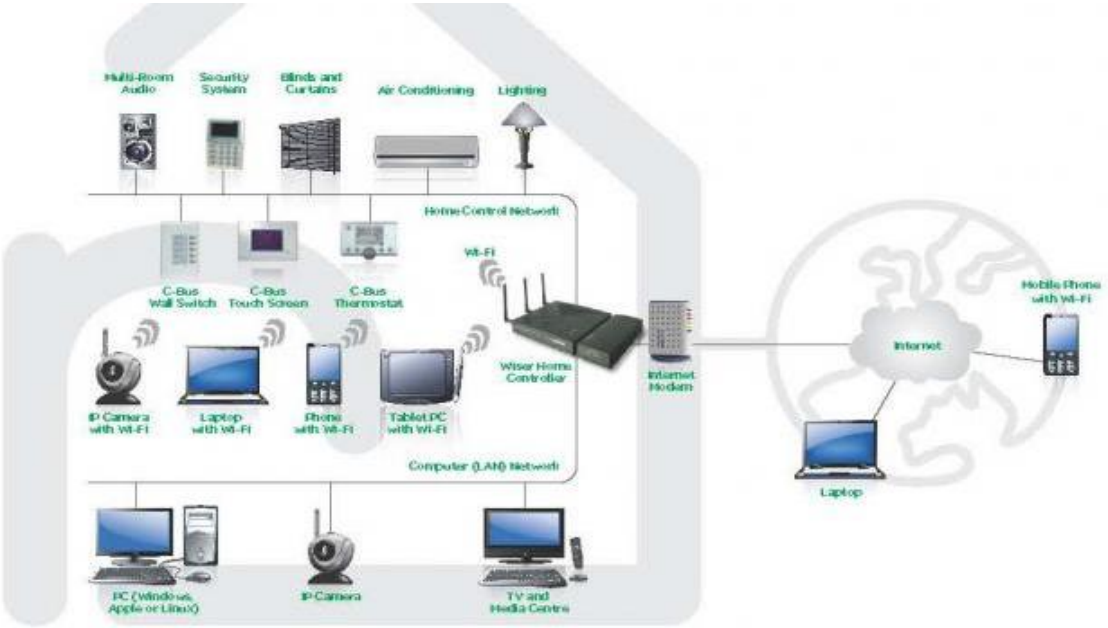

Figure 1: Home Automation System Architecture

Introduction

Building automation [2] is the automatic centralized control of a building's heating, ventilation and air conditioning, lighting and other systems through a building management system or building automation system. The objectives of building automation are improved occupant comfort, efficient operation of building systems, and reduction in energy consumption and operating costs, and improve life cycle of utilities.

Generally, building automation begins with control of mechanical, electrical, and plumbing systems. For instance, the heating, ventilation, and air-conditioning (HVAC) system is almost always controlled, including control of its various pieces of equipment such as: Chillers, Boilers, Air Handling Units, Roof-top Units, Fan Coil Units, Heat Pump Units, Variable Air Volume boxes.

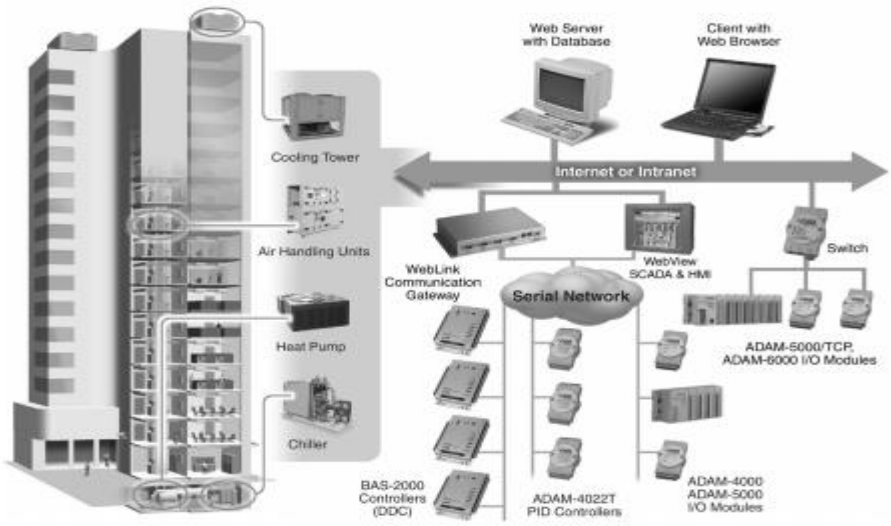

Figure 2: Building Automation Architecture

This work is organized as follows: the first chapter depicts an overview of "home automation system", the second and third chapters discuss the hardware and software implementation of the presented system. Next, the system overview and the conclusion.

Chapter 1:

HOME AUTOMATION SYSTEM.

This chapter provides an introduction to the Raspberry Pi, Relays, Sensors, and the technologies were used to monitor and control Home Automation System. We will explore the Raspberry Pi board, the external hardware needed for it to function properly, then, discussion about Web and Android application.

1.1 Raspberry pi

1.1.1 Overview

Raspberry Pi [3] can be defined as a low cost, credit-card sized computer that is plugged into a computer monitor or TV, and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python. It is capable of doing everything you would expect a desktop computer to do, from browsing the internet and playing high definition video, to making spreadsheets, word-processing, and playing games. It has a great bonding with Arduino.

In this work we use a specific model: Raspberry Pi 2 Model B.

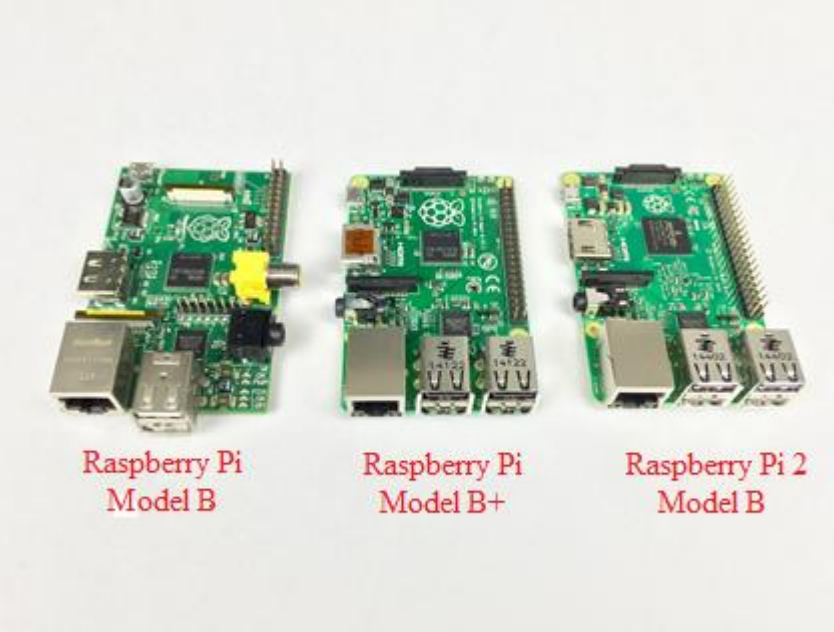

Figure 1.1: *Raspberry Pi Models*

1.1.2 Raspberry Pi 2 Model B

The Raspberry Pi 2 Model B is the second generation Raspberry Pi. It replaced the original Raspberry Pi 1 Model B+ in February 2015. Compared to the Raspberry Pi 1 it has:

- A 900MHz quad-core ARM Cortex-A7 CPU.
- 1GB RAM.

Like the (Pi1) Model B+, it also has:

- 4 USB ports.
- 40 GPIO pins.
- Full HDMI port.
- Ethernet port.
- Combined 3.5mm audio jack and composite video.
- Camera interface (CSI).
- Display interface (DSI).
- Micro SD card slot.
- VideoCore IV 3D graphics core.

Because it has an ARMv7 processor, it can run the full range of ARM GNU/Linux distributions, including Snappy Ubuntu Core, as well as Microsoft Windows 10.

The Raspberry Pi 2 has an identical form factor to the previous (Pi 1) Model B+ and has complete compatibility with Raspberry Pi 1.

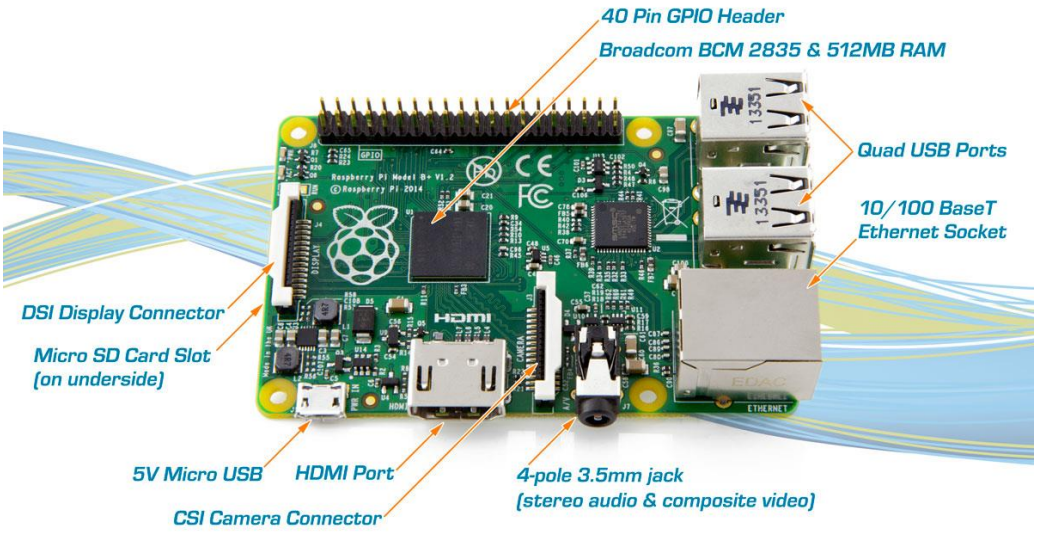

Figure 1.2: *Raspberry Pi 2 Model B*

1.1.3 External Hardware Required for Raspberry Pi

The Raspberry Pi does not work by itself, it requires some other external hardware to properly function, they are listed below:

- **Power Supply**

As mentioned before in the theory portion, Raspberry Pi needs 5V power supply. If the supplied voltage exceeds 5V, then, it is not guaranteed that it will work properly, the power supply also needs to supply at least 500 milliamps (mA), and preferably more like 1 amp (A). If the supply is 500 mA or less, it is likely to have the malfunction of keyboard and mouse. It is not a good idea to power the Raspberry Pi from USB port of computer and hub as they mostly provide a current less than required. Hence, the Raspberry Pi requires a Micro-USB connection which is capable of supplying at least 700 mA (or 0.7 A) at 5V.

- **Storage**

A separate hardware is required for the storage purpose in Raspberry Pi. For this, SD card is used, mostly 4 GB and 8 GB if needed. The operating system and all files are stored in the card.

We can buy a blank SD card and install the operating system or buy a pre-installed one.

- **Input**

External keyboard and mouse are required to provide input to the Raspberry Pi. No other additional software is needed to use the keyboard and mouse.

- **Monitor**

We can use a monitor or TV with HDMI port or DVI inputs as the screen for the Raspberry Pi. For DVI inputs, HDMI-to-DVI converters are required and can be found easily in the market. Monitor is most important for the Raspberry Pi and is the only way to see what we have done on it.

- **Network**

As in laptop or computer, we can access the internet and network in Raspberry Pi as well. For that, we can use wired Ethernet connection which is an easier option or Wi-Fi module to access Wi-Fi in the Raspberry Pi.

1.2 Relays and sensors External Modules

A relay [4] is an electro-mechanical switch made up of an electromagnet and a set of contacts, it consists of two different circuits that are independent. In the first circuit, a switch controls the power supply to the electromagnet. The second circuit is being operated by an armature, which is acting like a switch. Relays are used to control a circuit with a low-power signal, for example, radio or telephone. A sensor [5] is a device that senses or detects some characteristics of the environment, for example temperature or humidity. There are two types of sensors: analog and digital. Raspberry Pi uses digital sensors. However, it is possible to attach analog sensors to Raspberry Pi via a breadboard or an analog-to-digital converter.

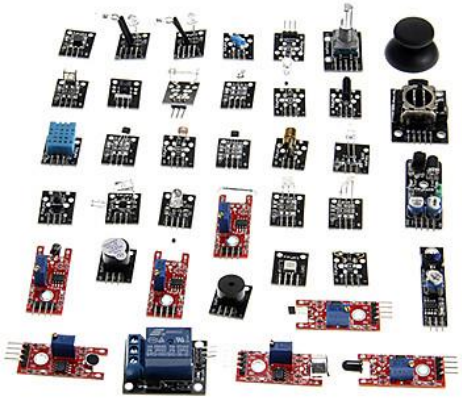

Figure 1.3: **a)** *Relay board* **b)** *Sensors*

1.3 Technologies used to control Smart Home

The user needs to have an easily configured and intuitive interface to fully monitor and control his home devices. This is achieved either from a web application or an Android application.

1.3.1 Web applications

the web allows both developers and users to exploit a variety of applications. Users can retrieve and interact with the data on the web.

Websites were developed over the years and became more dynamic, web pages can run client-side scripts that "change" the Internet browser into an interface for such applications as web mail and interactive mapping software.

Most importantly, modern websites allow the capture, processing, storage and transmission of sensitive customer data for immediate use, this is done through web applications.

"Web applications are, therefore, computer programs that allow website visitors to submit and retrieve data to/from a database over the Internet. The data is then presented to the user within their browser as the information is generated dynamically by the web application through a web server. For the more technically oriented, Web applications query the content server (essentially a content repository database) and dynamically generates web documents to serve to the client. The web browser interprets and runs all scripts while displaying the requested pages and content". [6]

1.3.2 Android Application

An Android application [7] is a software application running on the Android platform. Because the Android platform is built for mobile devices, a typical Android application is designed for a smartphone or a tablet PC running on the Android OS. It has good separation (and corresponding security) from other applications:

- Each application runs in its own process.
- Each process has its own separate Virtual Machine.
- Each application is assigned a unique Linux user ID – by default files of that application are only visible to that application (can be explicitly exported).

Android applications are written in the Java programming language and use Java core libraries.

There are four different types of application components. Each type serves a distinct purpose and has a distinct lifecycle that defines how the component is created and destroyed. Here are the four types of application components:

- Activities: visual user interface focused on a single thing a user can do.
- Services: no visual interface – they run in the background.
- Broadcast Receivers: receive and react to broadcast announcements.
- Content Providers: allow data exchange between applications.

Chapter 2:

Hardware Implementation.

This chapter aims firstly to provide a quick of system overview and after how to set up the Raspberry Pi. Then, to show how can the other components of the circuit be connected properly to the mentioned embedded system. In the end of this chapter we will see a backup battery system to substitute the primary power supply when it is unavailable.

2.1 System overview

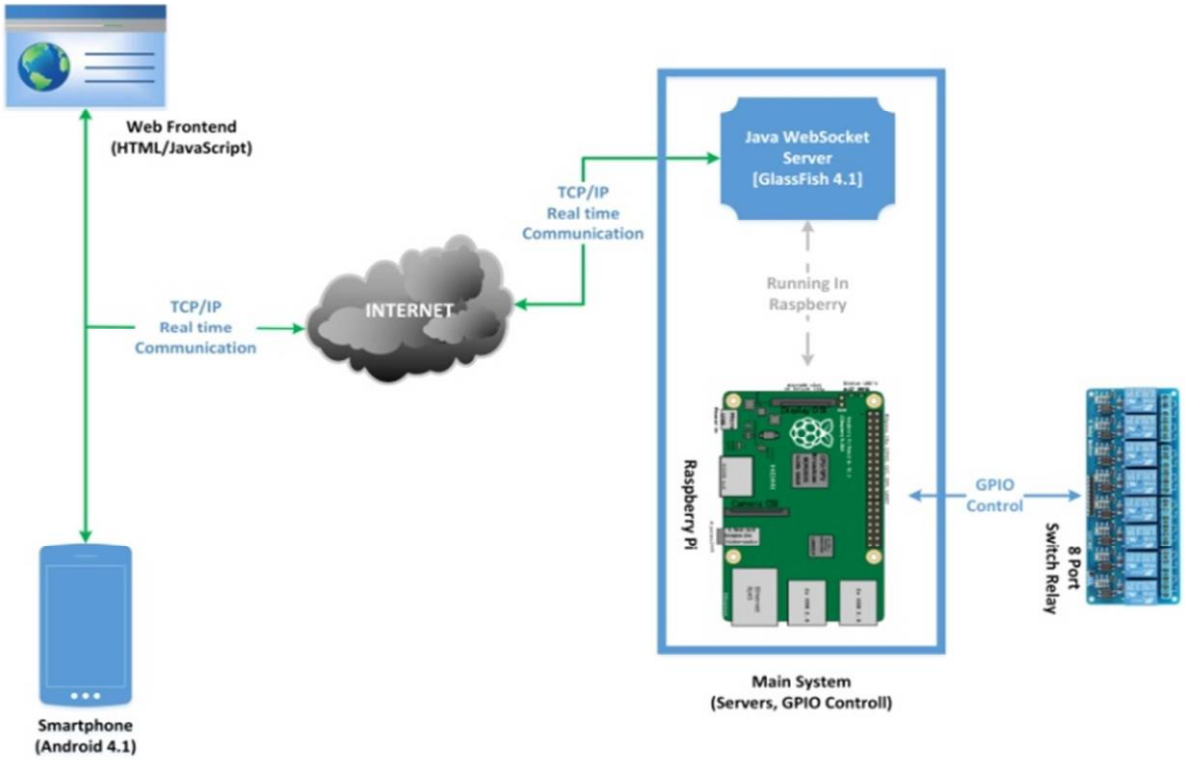

Figure 2.1: *System's architecture.*

The architecture of the proposed system is illustrated in Figure 2.1, the objective was to develop an easily configured and intuitive interface, yet powerful and reliable to provide its user with full monitoring and controlling skills over his home and the devices within. This system is composed, essentially, by three main components: Web-Frontend, Android application and the hardware component that hosts the main server. The main server links the individual applications

(Web-Frontend and Android) with Raspberry Pi and checks the GPIO ports, while providing, in real time, operational status data to two applications.

Using the Java Web Sockets JSR-356 standard we managed to implement a WebSocket server that is able to serve simultaneously the Web and the Android application (Figure 2.1), without the need of having two different servers or methods to control GPIOs. Communication is very simple and follows the rules of WebSockets dictating the use of text messages. Text messages in our case are in the form of JSON. After we managed to create a central point of communication between applications/devices, we had to give the ability to the server to control the connected devices. This was achieved by using the library PI4J.

The core of the home automation system consists of Raspberry Pi 2 Model B board. Figure 2.2 shows the integration of electrical components into the Raspberry Pi. The electronic components that we used are: Raspberry Pi 2 Model B, power supply for Raspberry Pi, 830 Tie Points MB102 Breadboard, SainSmart 8 Channel DC 5V Relay Module for Arduino Raspberry Pi, One Android 4.1 Smartphone, a wireless router for smartphone-server (Raspberry Pi) connection.

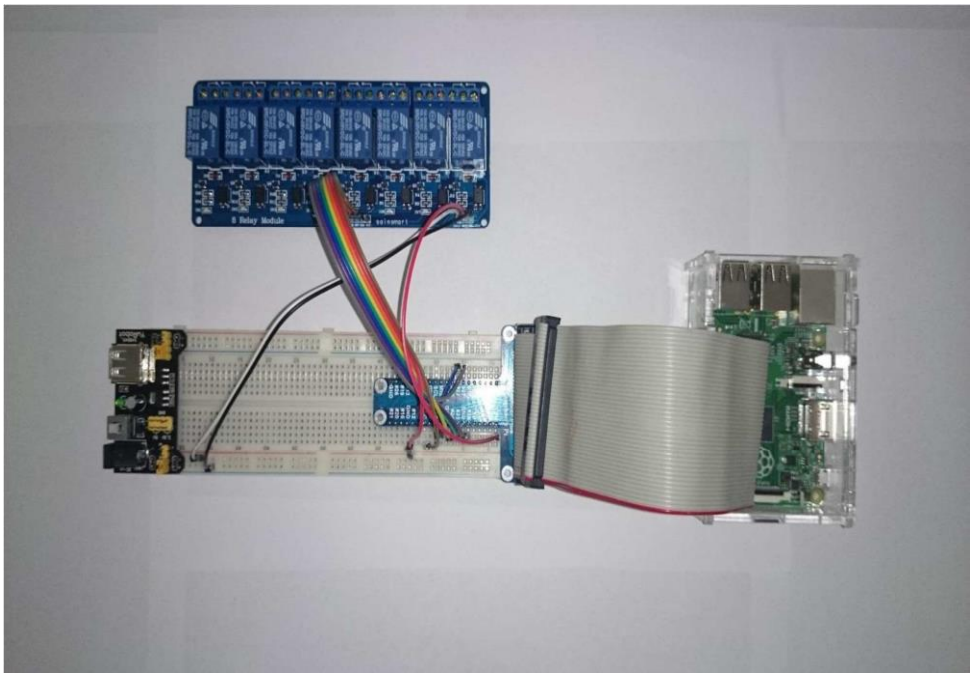

Figure 2.2: Hardware implementation

2.2 Raspberry Pi Configuration

Raspberry Pi is a small computer; hence an operating system should be installed. As the Raspberry does not have a hard drive, operating system is installed in the external memory. For that, a memory card (SD card) is used for the installation of operating system. All the required software and supporting files are stored in the same SD card.

There are different types of operating systems but we preferred to talk about Raspbian (the Foundation's official supported operating system), it comes pre-installed with plenty of software for education, programming and general use. It has Python, Scratch, Sonic Pi, Java, Mathematica and more. We can either buy a pre-installed SD card or empty SD card. In pre-installed SD card, Raspbian is already copied and ready to boot.

In order to use the laptop screen, keyboard and mouse, two software need to be downloaded, installed and configured, namely PuTTY [8] and Xming [9].

At the end of the procedure, the Raspberry Pi desktop is displayed as shown in figure 2.3:

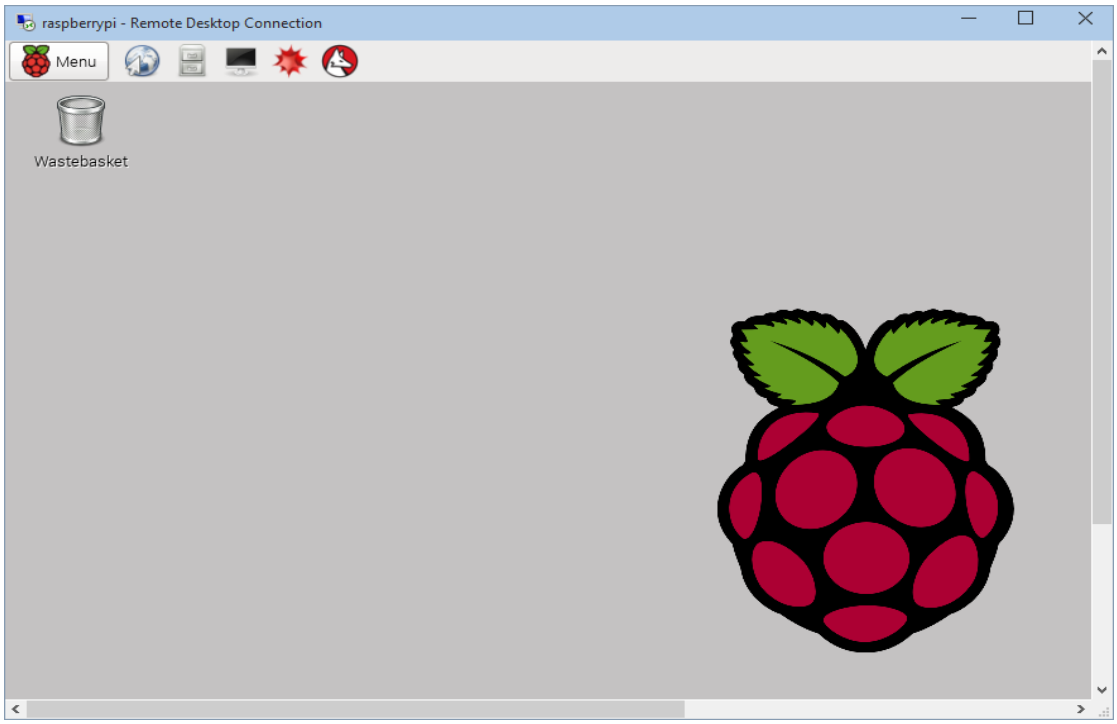

Figure 2.3: *Raspberry Pi Desktop on your Laptop Display*

2.3 System Implementation:

Before starting the implementation, we needed to install the library Pi4J in our configured Raspberry Pi to be able to control the devices after connecting them to the appropriate pin.

Features of this library include: export & unexport GPIO pins, configure GPIO pin direction, configure GPIO pin edge detection, control/write GPIO pin states, pulse GPIO pin state, read GPIO pin states, listen for GPIO pin state changes (interrupt based; not polling), automatically set GPIO states on program termination (GPIO shutdown), triggers for automation based on pin state changes, send & receive data via RS232 serial communication, I2C communication, SPI communication, extensible GPIO Provider interface to add GPIO capacity via expansion boards, access system information and network information from the Raspberry Pi, wrapper classes for direct access to WiringPi Library from Java.

The diagram below illustrates the GPIO pinout using the Pi4J/WiringPi [10] GPIO numbering scheme.

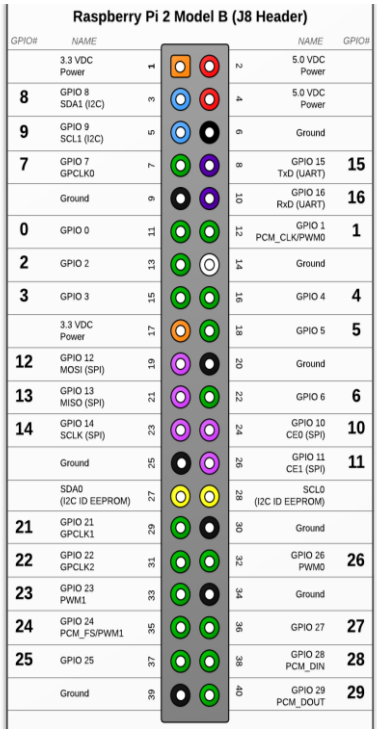

Figure 2.4: *GPIO pinout using the Pi4J/WiringPi*

The implementation was started by testing our components, which are: light and temperature sensors, 8 channel relay module.

- **Light Sensor:**

The Light Sensor module (Figure 2.5) is very sensitive to ambient light, and is very suitable for detecting brightness of ambient light. The output signal can trigger Raspberry Pi, microcontroller like Arduino, or other digital relay modules.

This light sensor module has 3 wires: VCC, GND and Signal.

Technical specification:

- using a high quality photoresistor.
- working voltage: 3.3~5V.
- output: digital switching (LOW or HIGH voltage on D pin) and analog signal (voltage output on A pin).
- adjustable potentiometer to adjust the sensitivity

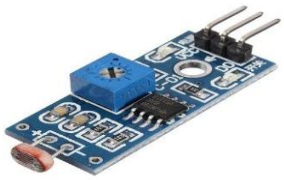

Figure 2.5: *light sensor*

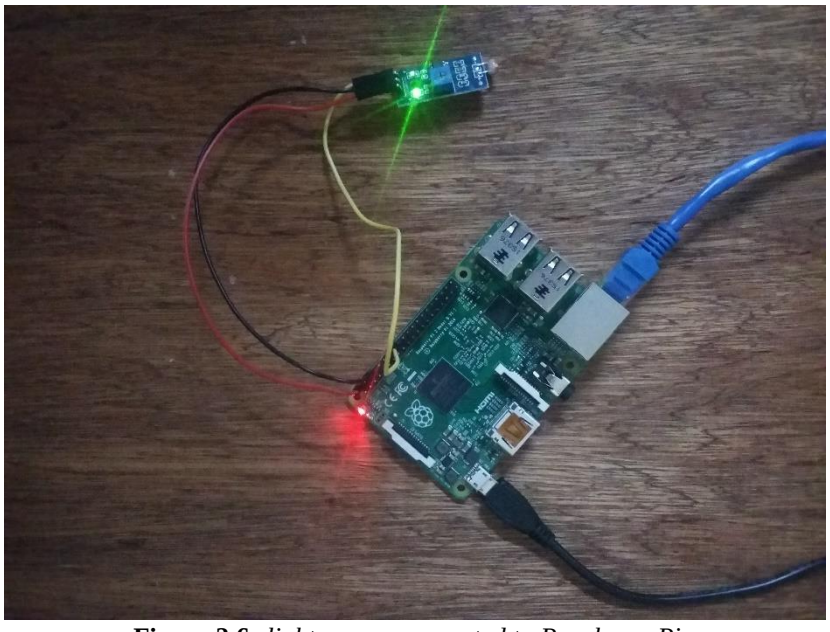

Figure 2.6: *light sensor connected to Raspberry Pi.*

- **DHT11 HUMIDITY & TEMPERATURE SENSOR:**

The DHT11 sensor includes a resistive-type humidity measurement component, an NTC temperature measurement component and a high-performance 8-bit microcontroller inside, and provides calibrated digital signal output.

- Power Supply： 3.3~5.5V DC.
- Output： 4 pin single row.
- Measurement Range： Humidity 20-90%RH, Temperature 0~50℃.

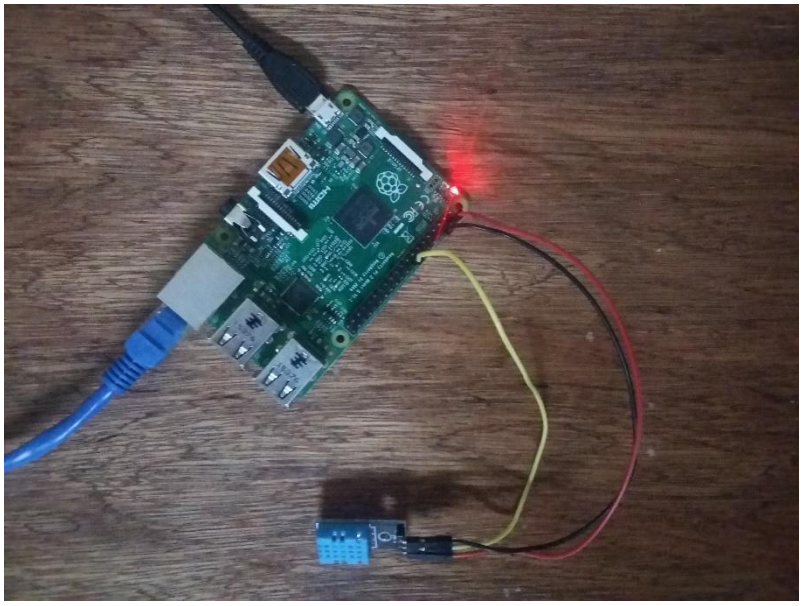

Figure 2.7: *DHT11 connected to Raspberry Pi.*

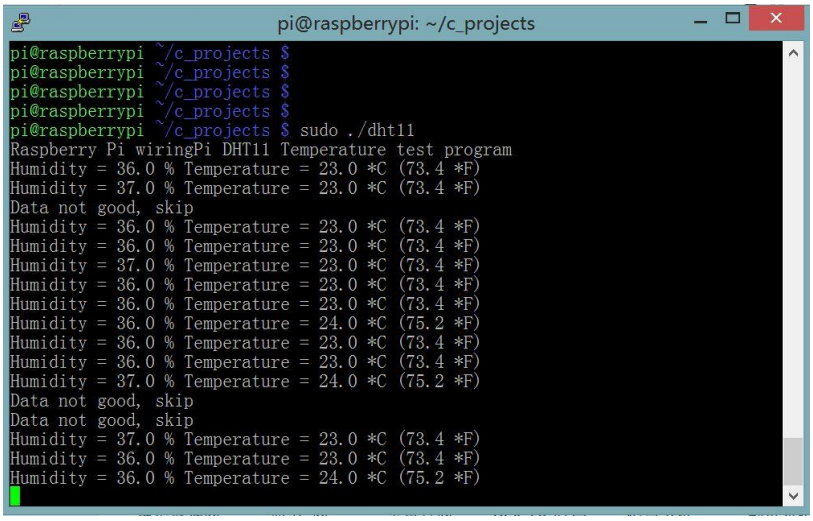

Figure 2.8: *Simulation results of Temperature and Humidity.*

- **8 Channel Relay Module**

A 5V 8-Channel Relay interface board, able to control various appliances, and other equipment's with large current. It can be controlled directly by Micro-controller (Raspberry Pi, Arduino , 8051, AVR, PIC, DSP, ARM, ARM, MSP430, TTL logic).

Feature:

- 5V 8-Channel Relay interface board, and each one needs 15-20mA Driver Current
- Equipped with high-current relay, AC250V 10A; DC30V 10A
- Standard interface that can be controlled directly by microcontroller (Arduino, 8051, AVR, PIC, DSP, ARM, ARM, MSP430, TTL logic)
- Indication LED's for Relay output status

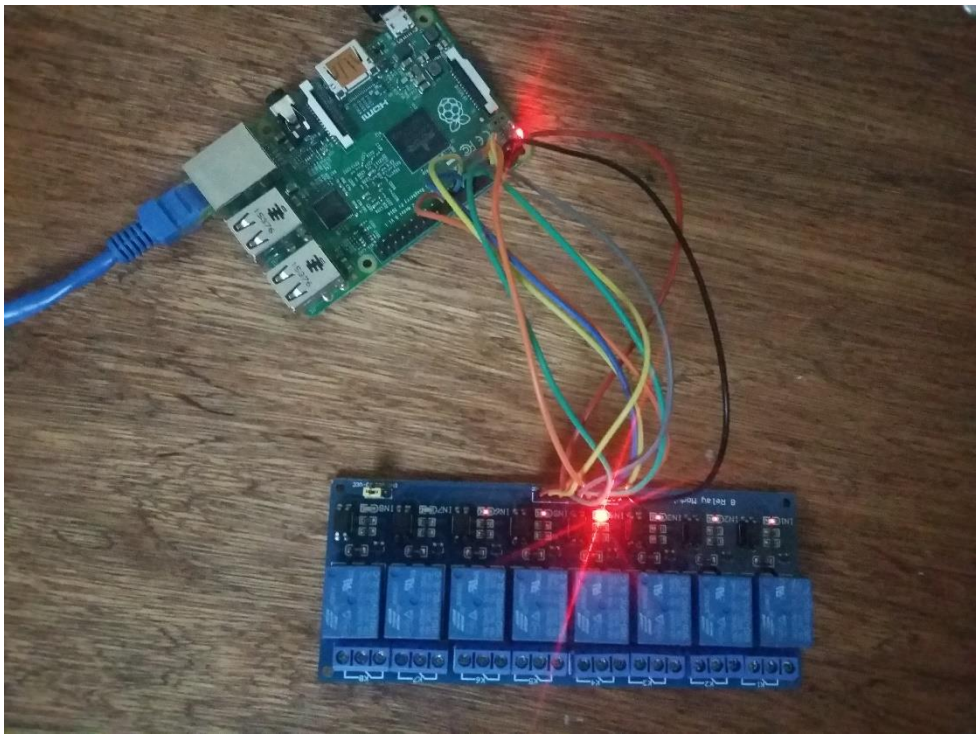

Figure 2.9: *Relay Board connected to Raspberry Pi.*

2.4 Backup Battery System

A backup battery [11] provides power to a system when the primary source of power is unavailable. Backup batteries range from small single cells to retain clock time and date in computers, up to large battery room facilities that power uninterruptible power supply systems for large data centers. Small backup batteries may be primary cells; rechargeable backup batteries are kept charged by the prime power supply.

we can propose the system shown in Figure 2.8 as a backup battery system for our smart home.

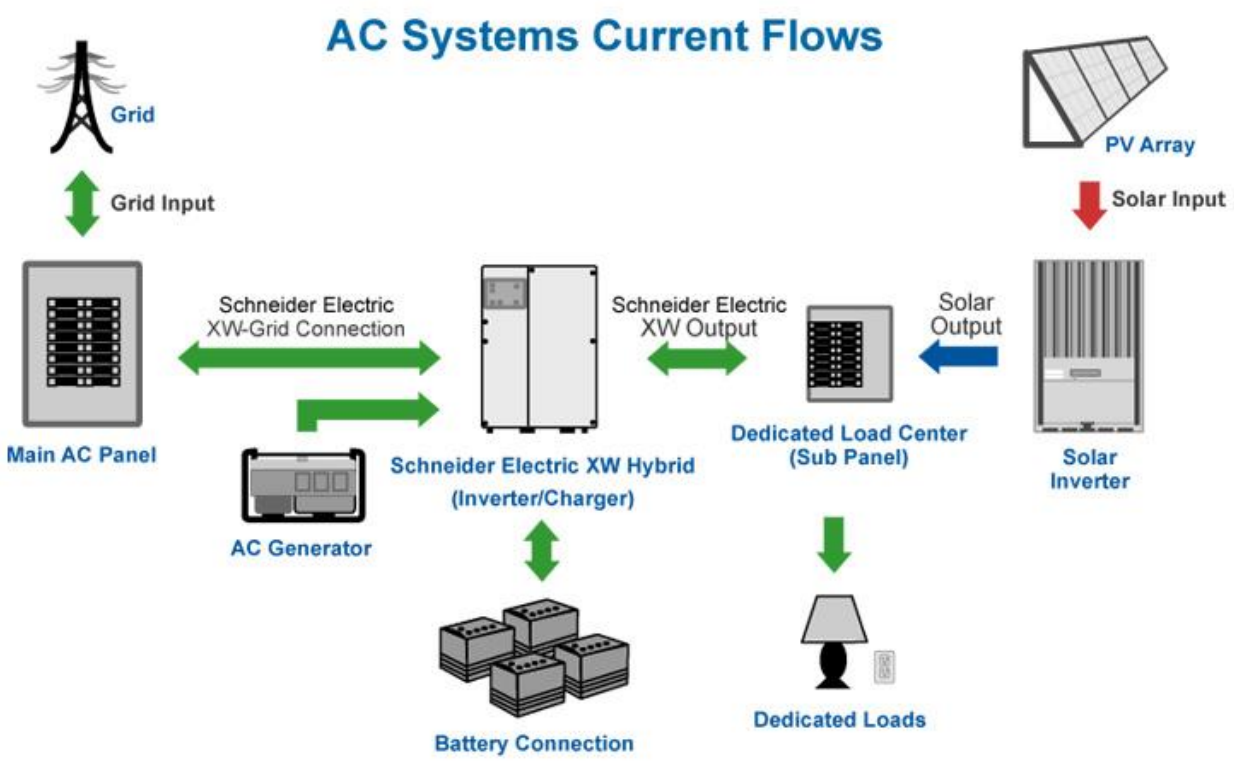

Figure 2.10: *Architecture of a Backup Battery System.*

Chapter 3:

Software Implementation.

This chapter deals with the software part that we used and developed which concerns the WebSocket server, as well as the Android and Web applications.

3.1 WebSocket Server

WebSockets [12] are an emerging web technology that provides full-duplex, bi-directional communication channels using TCP connections over the internet. As already mentioned for the implementation of the server we used the Java WebSockets JSR-356 standard.

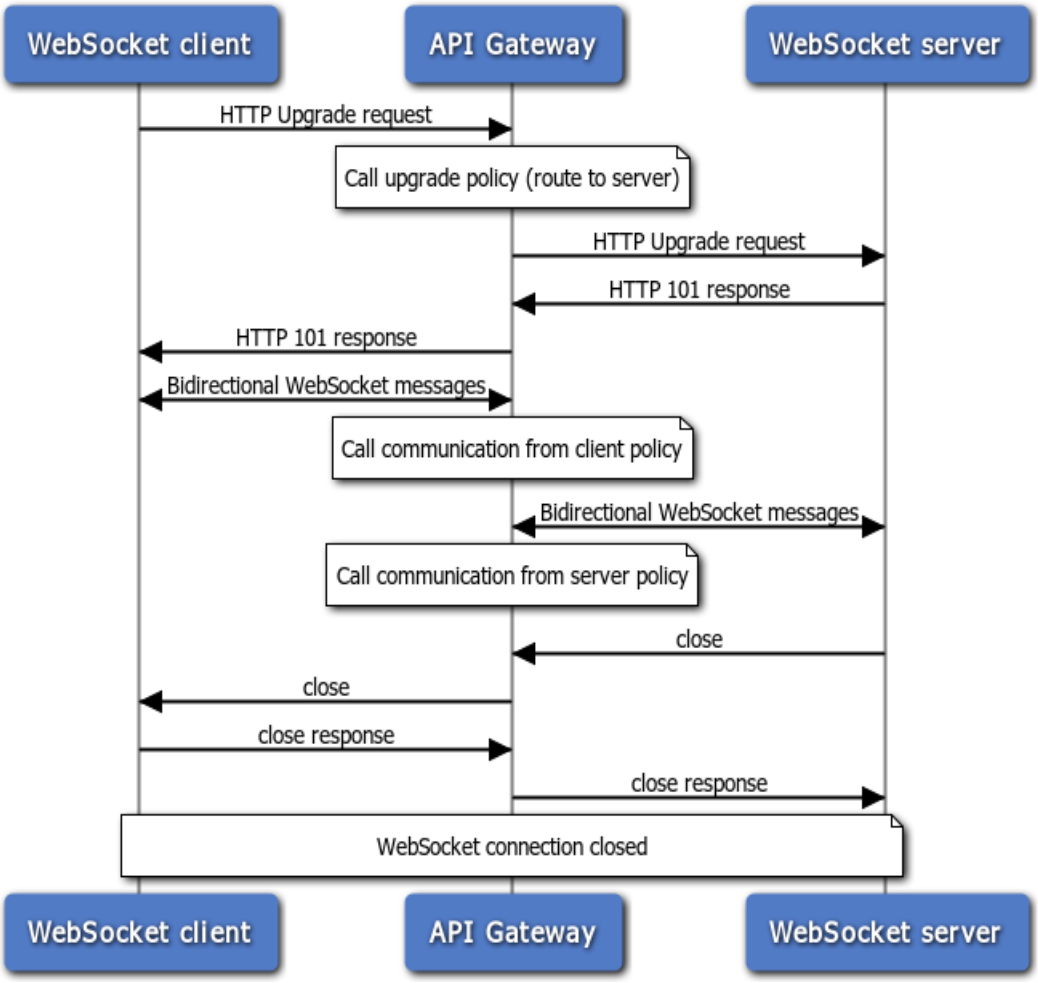

Figure 3.1: Sequence diagram showing flow of messages for WebSocket communication

Once a connection to the server has been established the client sends an opening handshake to the server. The WebSocket client's handshake is an HTTP upgrade request. Assuming the handshake succeeds the TCP socket underlying the HTTP upgrade request remains open and both client and server can start communication.

The client asks the server a protocol upgrade by sending a key in the Sec-WebSocket-Key header which is base64 encoded. For the server to form a response, it will take this and append the magic string 258EAFA5-E914-47DA-95CAC5AB0DC85B11 to it, and then calculate the SHA-1 hash of this string. Then it will encode that hash value to base64, and that will be the sec-WebSocket-Accept header in the server's response. Once the connection is established between the two parties, a full duplex communication can begin.

The server of this work is implemented using the basic four main methods of communication in the Java WebSocket includes:

@onOpen: It is used to decorate a method which is called once new connection is established.

@onClose: It is used to decorate a method which is called once the connection is being closed.

@onError: It is used to decorate a method which is called once Exception is being thrown by any method annotated with @OnOpen, @OnMessage and @OnClose.

@onMessage: It is used to decorate a method which is called once new message is received.

It also includes the @Inject method, whereby the session handler class is introduced. In which includes all functions that take place during the client-server communication.

The server has the ability to control the connected devices, this was achieved by using the library Pi4J.

The application runs on Glassfish 4.1 server [13].

3.2 Web Application

For the development of the Web application, as regards the client's part, HTML5 and JavaScript technologies were used. On the server side, we have the PHP pages that read static data from JSON files. Data that inform about the GPIO ports that are in use or the devices connected

to each room. For the implementation of the Web application we used the Adobe Dreamweaver [14].

The following use case diagram shows the different types of roles in a Web application and how those roles interact with it.

Figure 3.2: *use case diagram for Web application.*

First, the user needs to login, after a successful entry of a user, the selection room will appear, then, user selects a room of interest. At the end, comes up the view of a room, the corresponding connected devices and the status of each device.

The Web application allows control of remote devices via a website. This website has 3 main interfaces:

- Login Form:

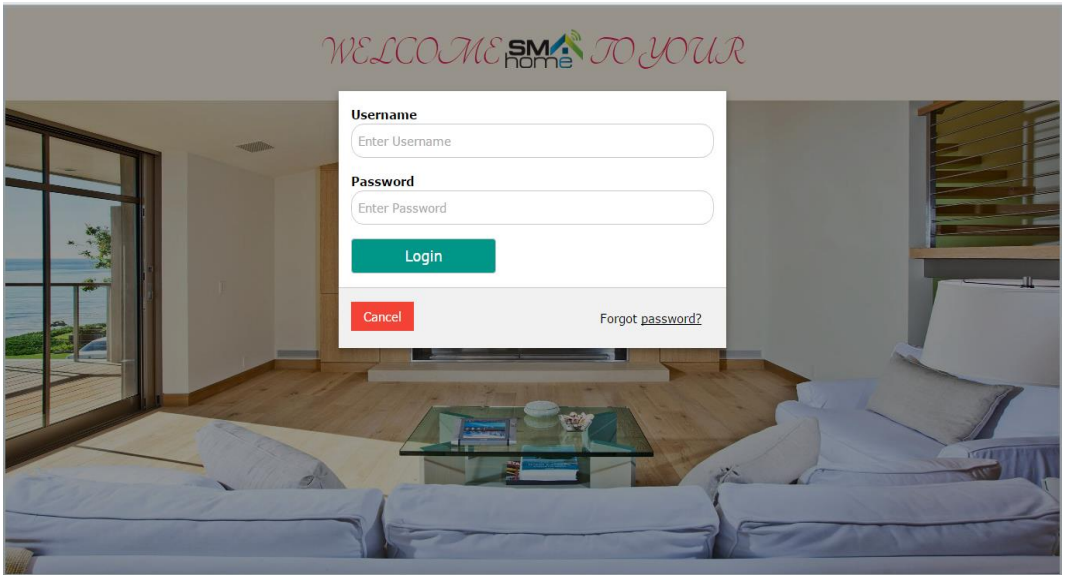

Figure 3.3: *Login screenshot.*

- Room selection:

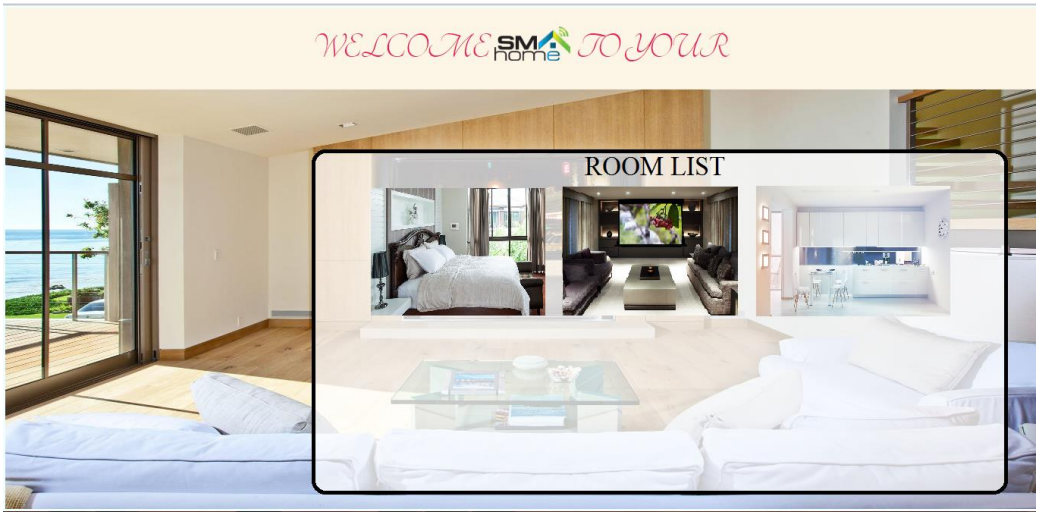

Figure 3.4: *Room selection screenshot.*

- Room view:

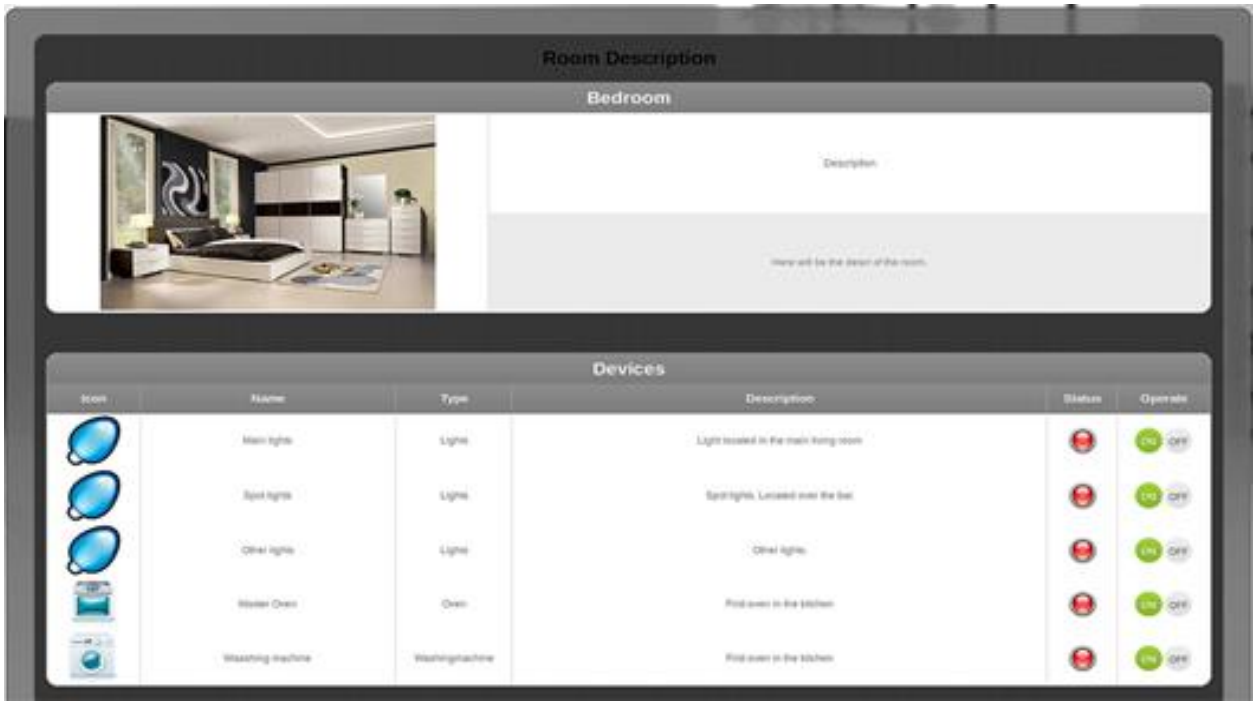

Figure 3.5: *Selected room screenshot.*

- Login Form (Figure 3.3). The Login interface is a simple PHP script which in turn is connected to a SQLITE database. Upon successful entry of a user, comes up the room selection screen (Figure 3.4).

- Screen for selecting a room (Figure 3.4). On this screen a user selects a room of interest. Rooms are stored in a JSON file. The JSON file contains the name of a room, its description and its type (i.e. if it is living room, bedroom, etc.). For each room there is a JSON object.

- View of a room and the corresponding connected devices (Figure3.5). This screen shows the name of a selected room, its description and all connected devices that belong to this room, grouped on a board. For each device on the board there is a representative icon (according to the type of the device), its name, its type, its description, and finally a status button that indicates the status of the device (open or close) and two operational buttons (ON, OFF). All this information, for each device, is stored in JSON files.

At first all the status buttons are disabled and are coloured gray. This indicates that the status of each device has not yet been received from the server. Once the server obtains status data, the

indicator lights will change colour and the corresponding status buttons will be activated. If for example a device is opened, then the corresponding status button will turn green and its operational OFF button will be activated.

On the other hand, if a device is closed, then the corresponding status button will turn red and its operational ON button will be activated. The information about the status of each device is transferred, in real time, from a Java Script associated with the server.

3.3 Android Application

Although for control via mobile phone could be used its embedded browser, but we developed a separate application.

This application is able to run on any Android device and have an internet connection either Wi-Fi or from a mobile provider.

In order to be able to use the same server, the one that serves the Web application. To do that the most suitable library for this purpose is Tyrus 1.10 [15]. Tyrus is actually an API, which implements the Java JSR-356 with methods that make its use particularly easy. Just because Tyrus is an implementation of the JSR-356, communication between the mobile application and the server was from the beginning without any problem. The mobile application runs on Android 4.4 and for development we used the Android Studio [16].

The Android application works with the same logic as the Web application and provides the user with four activities (Android Activities) that lead him to the management of the desired device. The user has as final activity the “Operate Device” which provides information on a device, and a button for control it.

The route throw the application is diagrammatically: main screen (Figure 3.6) >Screen for selecting a room (Figure 3.7) > Screen for selecting a device (Figure 3.8) > Control screen (Figure 3.9).

- main screen:

Figure 3.6: *Main screen.*

- Screen for selecting a room:

Figure 3.7: *Room list.*

- Screen for selecting a device:

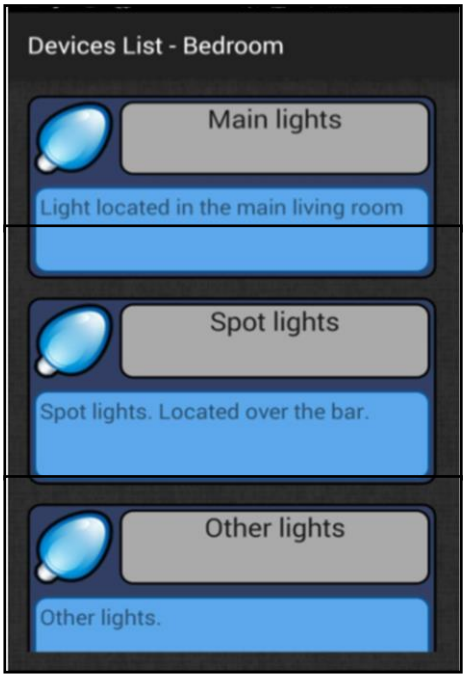

Figure 3.8: *Device selection.*

- Control screen:

Figure 3.9: *Device control.*

The Android application works with the same logic as the Web application and provides the user with four activities (Android Activities) that lead him to the management of the desired device. The user has as final activity the “Operate Device” (Figure 9) which provides information on a device, and a button for control it.

The transition from one activity to another is accomplished by calling an intent. The intent contains additional information, in the form of a JSON, suitable for any activity. All four activities include the same function through which they are connected to the WebSocket server. This function transfers the necessary data.

conclusion

General Conclusion

In this highly technological environment, by it's over clocking and future expansion capabilities, Raspberry Pi proved to be easy, economic and efficient platform for implementing the home automation system. The Raspberry Pi ensure the intelligent control of a house upon user authentication. We accomplished this through the use of low cost devices and the development of two user friendly interfaces for a Web and an Android application.

A WebSocket server, running on a Raspberry Pi card, is able to serve simultaneously in real time both the Web application and the Android one, controlling the connected devices.

Very soon in near future, the traditional grids of today will evolve into a robust, effective, environment friendly and energy efficient system known as the Smart Grid. Even our home will undergo its own transformation towards the smart homes that will be in constant interaction with the grid in an effort for better energy management and full home automation to ensure comfort, security and privacy. Present report sought to design a smart home using various sensors to be controlled and monitored by the Raspberry Pi via the IoT.

Home automation included a smart door bell, humidity and temperature control via a fan or air-conditioning system of the home. The system is also equipped with automated lights and virtual switches for controlling lights and appliances in the home remotely using external and / or internal networking with the Raspberry Pi via Web or Android application. Full functionality of prototype indicates that devices like Raspberry Pi can play very important role in designing smart home of the future at very low cost. An energy aware smart home can be developed using Raspberry Pi and other sensors.

This work can be enhanced for future applications in Building Automation Systems like power grid control and protection, surveillance, power monitoring, fault monitoring, security etc.

Building automation is an example of a distributed control system – the computer networking of electronic devices designed to monitor and control the mechanical, security, fire and flood safety, lighting (especially emergency lighting), HVAC and humidity control and ventilation systems in a building.

Conclusion

In the era of digital revolution, where technologies and devices are entering every aspect of life, the term “smart” creates in peoples' minds great expectations on the potentialities of the future buildings. It stimulates imagination even beyond the real possibilities. This increases the discussion on the topic creating two effects. On the one side, it becomes more difficult to get a clear vision of the state of the art and distinguish between what is real and what is imagination, on the other hand, visions and dreams are a stimulation to push technologies and reality beyond.

In some cases, visions and dreams on the future reality of homes and buildings started well before the digital revolution. These influenced and shaped culture and people’s perception about future buildings and, consequently, the technologies that have then been developed that are now appearing. Such authors like Asimov, cult movies and cartoons like Star Wars or the Jetson family created imaginaries that are now affecting our society. It is indeed that such references have a strong power and impact on the development of the new actual technologies.

At the same time, these technologies have their own impact on society. In particular, people dream and imagine a futuristic world but then they are not so available to get reached by it. This resilience to the new is particularly strong in the field of buildings for several historic and again cultural reasons. In fact, many of the proposed smart technologies are “hidden”, in the sense that they act tending not to change people habits but just optimize the existing building systems.

References:

[1] https://en.wikipedia.org/wiki/ [Online], accessed on Mai 1st, 2016. Available on: https://en.wikipedia.org/wiki/Home_automation

[2] Cisco Validated Design, " Building Automation System over IP (BAS/IP) Design and Implementation Guide " Available on:

http://www.cisco.com/c/dam/en_us/solutions/industries/docs/trec/jControls_DIG.pdf

[3] https://www.raspberrypi.org/ [Online], accessed on Mai 1st, 2016.

[4] www.sainsmart.com [Online], accessed on April 19th, 2016. Available on: http://www.sainsmart.com/8-channel-dc-5v-relay-module-for-arduino-picarmdsp-avr-msp430-ttl-logic.html.

[5] https://en.wikipedia.org/ [Online], accessed on April 15th, 2016. Available on: https://en.wikipedia.org/wiki/Sensor

[6] Jaosn Lengstorf, Phil Leggetter, Book: "Realtime Web Apps with HTML5 WebSocket, PHP, and jQuery", 2013.

[7] https://www.android.com/ [Online], accessed on Mai 15th, 2016.

[8] http://www.putty.org/ [Online], accessed on March 7th, 2016.

[9] https://sourceforge.net [Online], accessed on March 7th, 2016. Available on: https://sourceforge.net/projects/xming/

[10] http://pi4j.com/ [Online], accessed on March 7th, 2016.

[11] http://www.outbackpower.com [Online], accessed on Mai 2nd, 2016. Available on:

http://www.outbackpower.com/downloads/documents/flexcoupled/app_note_acc_new.pdf

[12] https://tools.ietf.org/html/rfc6455 [Online], accessed on March 17th, 2016.

[13] https://glassfish.java.net/ [Online], accessed on March 27th, 2016.

[14] http://www.w3schools.com/ [Online], accessed on March 7th, 2016.

[15] https://tyrus.java.net/ [Online], accessed on March 7th, 2016. Available on: https://tyrus.java.net/documentation/1.12/user-guide.html.

[16] https://developer.android.com/ [Online], accessed on Mai 16th, 2016. Available on: https://developer.android.com/studio/index.html.