American Journal of Mobile Systems, Applications and Services
Vol. 1, No. 3, 2015, pp. 174-181
http://www.aiscience.org/journal/ajmsas

Home Automation Based on an Android and a Web Application Using Raspberry Pi

Dimitris Kehagias[*], Dorian Nini

Department of Informatics, Technological Educational Institution of Athens, Athens, Greece

Abstract

The aim of this paper is to present the development of a flexible, low cost system to remote control of home appliances. The control is accomplished either from an Android or a Web application. Both applications communicate with a common server, which provides status data in real time, both to Browser and Android clients. The system is based on Raspberry Pi embedded system.

Keywords

Home Automation, Raspberry Pi, Android, Web

Received: August 31, 2015 / Accepted: September 16, 2015 / Published online: October 16, 2015

1. 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, the use of such suitable devices in a home connected to a local network creates a "Smart home".

A "Smart home" uses microcontrollers that automate the monitoring of conditions of the house and change the condition of other devices (Home automation). Specifically, microcontrollers can measure parameters associated with a living area such as temperature or humidity of a room or outdoor in order to control and alter the status of other devices, that is, microcontrollers can automatically enable or disable an electronic device. Having the proper equipment, a house enables the user to know at any moment the situation on site and to control any device connected to the network.

This capability of the user to constantly monitor his house first aims to facilitate his living and achieve the lowest possible power consumption. This can be done either entirely within the user's discretion, i.e. to manually alter the status of the devices, or by setting specific threshold values that are maintained by automatically controlling the desired devices.

Such a home may be accomplished by either ready embedded systems or by using a platform that allows the connection of other devices and components therein. One such platform is the Raspberry pi [1] used in this work.

The purpose of this work is to develop an easily configured and intuitive interface, yet powerful and reliable to provide its user with full monitoring and controlling capabilities over his home and home devices. This is achieved either from an Android or a Web application. The challenge we faced was to find a way in which the two applications will communicate with a common server, which will provide status data in real time not only in the Android application but also to the Web application while in the meantime being able to control the connected devices. In addition, we had to keep the web technologies HTML, PHP and JavaScript without the need to create a Java Applet.

We have addressed this challenge by using the WebSockets [2], [3] communication protocol, which is designed to be implemented in web browsers and web servers, but can be used by any client/server application. But the real importance

* Corresponding author
E-mail address: dkehayas@teiath.gr (D. Kehagias)

is that WebSockets provide a way to build scalable, real-time web applications. So, with the help of the “Java Web Sockets JSR-356”standard we implemented a WebSocket server, which is able to serve simultaneously in real time both the Web application and the Android one, controlling the connected devices using the PI4J library. The appropriate server, that is running our application, is Glassfish 4.1.

This paper is organized as follows: the following section depicts a brief overview of the aspects “Internet of Things”, “Home automation” and “Related work”. Section 3 exhibits the issues we had to consider in designing the proposed system. Next, the hardware and software implementation of the system is presented. Then, we conclude in Section 5.

2. Smart Home

The popularity of remote intelligent home system has been increasing greatly in recent years due to much higher affordability and simplicity through internet connectivity. The concept of "Internet of Things" is closely associated with the commercialization of Domestic/Industrial automation.

2.1. Internet of Things (IoT)

IoTs can be described as connecting everyday objects like smart phones, internet televisions, sensors and actuators to the internet where the devices are intelligently linked together to enable new forms of communication amongst people and themselves [4].

A thing, in the Internet of Things, can be a person with a heart monitor implant, a farm animal with a biochip transponder, an automobile that has built-in sensors to alert the driver when tire pressure is low - or any other natural or man-made object that can be assigned an IP address and provided with the ability to transfer data over a network. So far, the Internet of Things has been most closely associated with machine-to-machine (M2M) communication in manufacturing and power, oil and gas utilities. Products built with M2M communication capabilities are often referred to as being smart [5]. Such smart devices are widely used in home automation.

2.2. Home Automation

Home automation or Smart Homes can be described as introduction of technology within the home environment to provide convenience, comfort, security and energy efficiency to its occupants. Adding intelligence to home environment can provide increased quality of life. With the introduction of the Internet of Things, the implementation of home automation is getting more popular.

The components of a home automation system can be: sensors (such as temperature, daylight, or motion detection sensors), controllers (such as a dedicated automation controller), activators (such as light switches and appliances) and channels (wired or wireless).

For such a system it is required one or more “man to machine” and/or M2M interfaces, so that the inhabitants of the house to be able to interact with the system for monitoring and control. This may consist of a dedicated terminal or nowadays it may be an application that runs on a smart phone or a tablet or even on one web page. The devices can communicate via individual wiring or by connecting them all to a wired network, or wirelessly, using one or more protocols. So we can use a central controller as well as we can individual handling each device at home.

This field has many applications. As a security element it can be oriented toward personal security or material objects. As an energy efficiency element, light intensity sensors can be used to keep the light level within a range controlling shutters to reduce the power consumption. Combining light sensors with presence sensors lights in bedroom can be turned off when not required significantly reducing the power consumption. As a comfort element a reasonable temperature can be maintained in different rooms with changing outdoor conditions [6].

Communication protocols are one of the most serious issues for home automation due to the fact that the devices installed in the house do not support common protocols making the communication between them hard to implement. Thus, we should ensure that there is compatibility between smart devices we choose. The most common protocols that support wireless communication are the Z-Wave and ZigBee [7].

2.3. Related Work

The idea of home automation using the Raspberry Pi platform has been around almost since it launched in 2012. First concepts for home automation based on the aforementioned platform were a little crude, but nonetheless they opened the road to the creation of new technologies and richer APIs.

Plethora of approaches exists that use the Raspberry Pi as a base for home automation. During our research and comparison of different works related to the subject, we found out that some aspects that were considered very important for us were missing such as ability to control devices both from a Web frontend and Android app, independence from other frameworks, a monolithic approach, real-time data and others.

Soundhar Ganesh et al. [8] proposed a system that operates with a built in display as a central control panel and different sensors and relays perform the desired operations. The remote operation of the devices is achieved by reading the title of an e-mail. Numerous other schemes have been proposed but only with a single control point, either it being a Web frontend [9], or either an Android application [10], [11], [12]. We propose a system that gives the user more freedom. We created two remote applications, a Web frontend and an Android application. In case the user doesn't have an Android device he can still use the Web frontend in any HTML5 compliant browser. The challenging issue was to make the system communicate at the same time to the Android and Web frontend, since they use different technologies. We used WebSockets technology. WebSockets not only allow us to have real-time data communication (it's basically a TCP/IP connection) but can also communicate with different programming languages as long as they have an implementation for it like Lua, NodeJS, Python, Java, Javascript and others. The aspect of WebSockets also allows for expansion of our implementation through add-ons that can be created using one of the aforementioned languages.

Other architectures rely on the use of frameworks [13]. Albeit our application is not as complete as IBM's Bluemix for example, we wanted to have a "from scratch" solution to our problems. This ideology also avoids being depended from other companies where you might have to pay for loyalties.

Furthermore we are using Android and WebSockets. This is fairly new and not many implementations exist using this kind of technology. This was possible with the use of the Tyrus 1.10 implementation of the Java JSR-356 specification.

Although we are not the first to use WebSockets for this kind of projects, we can say that the strongest point of our proposal is the combination and integration of as many features as possible to a home automation system. Also the use of WebSockets in combination with the Android is quite a new option for making real time communications between Web, Android, Java and any other language that has an implementation for WebSockets.

3. System Overview

The architecture of the proposed system is illustrated in Figure 1. As stated before, our 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.

3.1. Designing Issues

The challenge we faced from the outset was to find a way in which the two applications will communicate with a common server, which will provide status data *in real time* not only in the Android application but also to Web application. In addition, we had to keep the web technologies HTML, PHP and JavaScript without the need to create a Java Applet.

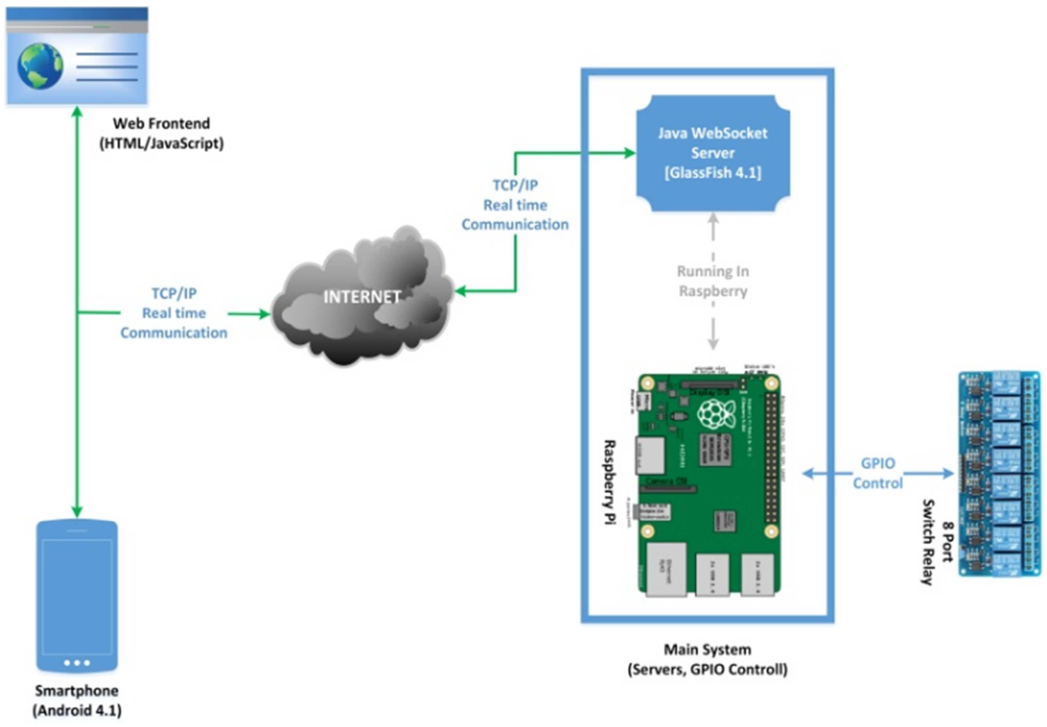


Figure 1. System's architecture.

The problem arises from the fact that we have 2 applications based on completely different technologies. One technology should be implemented in PHP-HMTL5, and the other with Android-Java. The communication of each individual application with the main server is done without any particular problem:

a) The Web part will simply have some PHP Scripts which would call Python scripts, which in turn would control the state of the GPIO ports. The Web application would get the status data with a Long Poll Ajax request, *but not in real time*.

b) The Android part would contact a Java TCP/IP server (the same that informs him about the data) and thus controlled the GPIO. A Java TCP/IP server would inform the Android application for any change of state in real time.

We can see some of the problems that arise:

- We need to separately address each communication.
- We need to ask the server with Long Poll Ajax requests, consuming unnecessary data.
- It is not achieved 100% real-time updating of data. If for example an Ajax request has 5” data application time, then data updated every 5” rather than when the updating is really possible.
- Running conflict: What happens when the Android application instructed to open a device and the Web application has not yet seen the change of status?
- Possible overload of Apache, and consequently the Raspberry (because of limited capacity), by Ajax requests.

3.2. Our Approach

We faced the above issues by using WebSockets. The WebSockets is a protocol which provides full-duplex communication over a single TCP connection. It is relatively new technology that became a proposed IETF (Internet Engineering Task Force) specification In December 2011 [3].

Technically, WebSockets allow a long-held single TCP socket connection to be established between the client and the server, removing the need to poll the server and allowing messages to be sent back and forth while keeping the connection open. But the real importance is that WebSockets provide a way to build scalable, real-time web 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 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.

4. System Implementation

This section describes the hardware and software implementation of the system.

4.1. Hardware Implementation

The core of the home automation system consists of Raspberry-pi Model B+ board. Figure 2 shows the integration of electronic components into the Raspberry-pi. The electronic components that we have used are:

Raspberry Pi Model B+, power supply for Raspberry Pi, T-Cobbler Plus-Breakout Board, DC/DC YwRobot Power MB V2 power supply, 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.

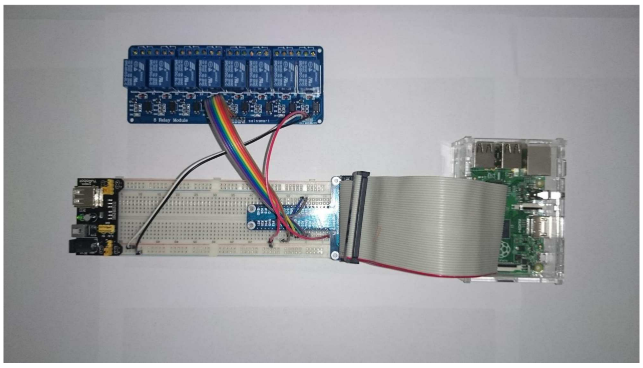


Figure 2. Hardware implementation.

4.2. Software Implementation

The software that we developed/used in this work concerns the Web and Android applications, as well as the main server.

As operating system we chose Raspbian which is based upon the Debian Wheezy Linux operating system and has been optimized for use with Raspberry Pi.

4.2.1. 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 CS6.

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

- Login Form (Figure 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 4).
- Screen for selecting a room (Figure 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 (Figure5). 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.

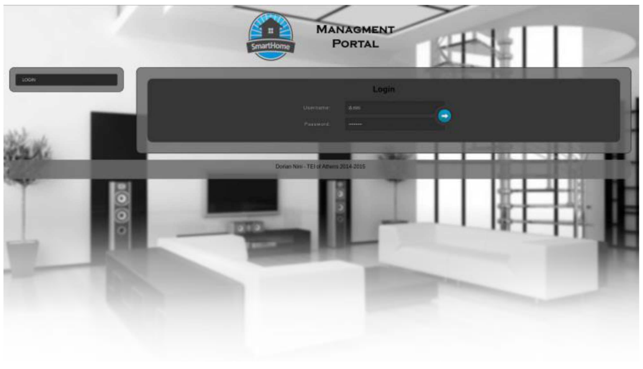


Figure 3. Login screenshot.

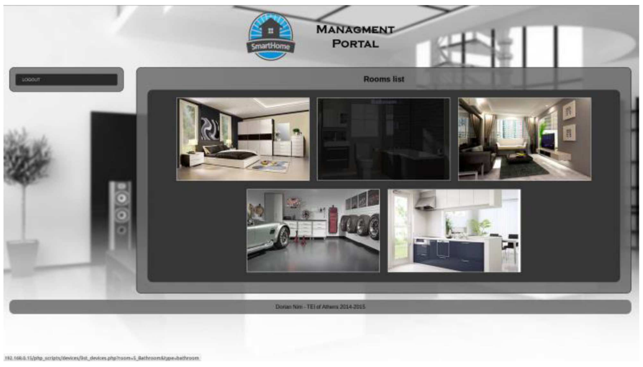
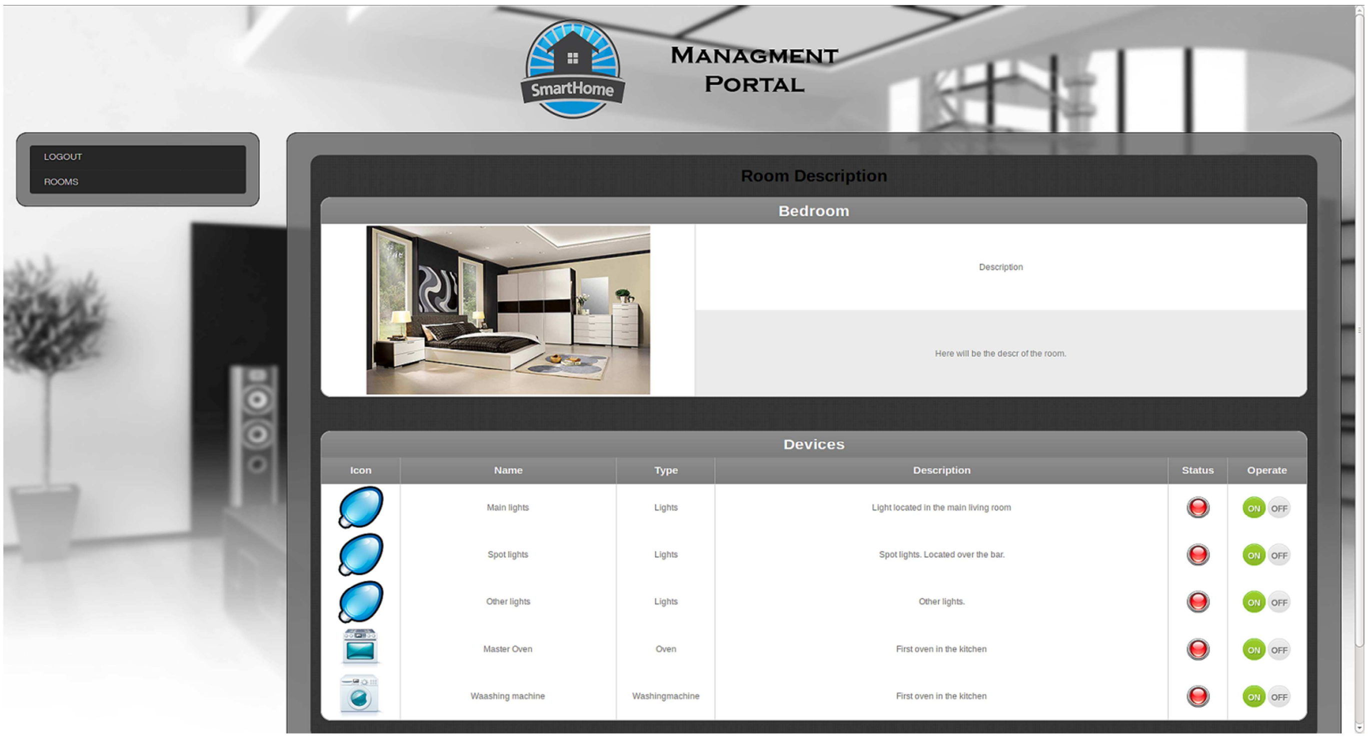


Figure 4. Room selection screenshot.



Room Description

Bedroom

Description

Here will be the descr of the room.

Devices

Icon	Name	Type	Description	Status	Operate
	Main lights	Lights	Light located in the main living room	🔴	ON OFF
	Spot lights	Lights	Spot lights. Located over the bar.	🔴	ON OFF
	Other lights	Lights	Other lights.	🔴	ON OFF
	Master Oven	Oven	First oven in the kitchen	🔴	ON OFF
	Waashing machine	Washingmachine	First oven in the kitchen	🔴	ON OFF

Figure 5. Selected room screenshot.

4.2.2. Android Application

Although for control via mobile phone could be used its embedded browser, we decided to develop a separate application because we wanted to show:

- The Flexibility of WebSockets
- The convenience that provides a standalone application without having to rely on a browser

This application is able to run on any Android device with a minimum OS version of 4.1(currently covering around 80% of Android devices) and have an internet connection either WiFi or from a mobile provider. In order to be able to use the same server, the one that serves the Web application, we had to find a suitable library for WebSockets that could function on Android environment as well. Although the existing Java implementation works fine, its use in Android environment is somewhat difficult. The most suitable library for this purpose is Tyrus 1.10 [14]. Tyrus is actually an API, which implements the JavaJSR-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 4.4.4 and for development we used the Android Studio 1.1.0. [15]

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 route throw the application is diagrammatically: main screen (Figure 6) >Screen for selecting a room (Figure 7) > Screen for selecting a device(Figure 8) > Control screen (Figure 9). 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.

4.2.3. Main Server

As already mentioned for the implementation of the server we used the Java WebSockets JSR-356 standard.

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-95CA-C5AB0DC85B11 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.

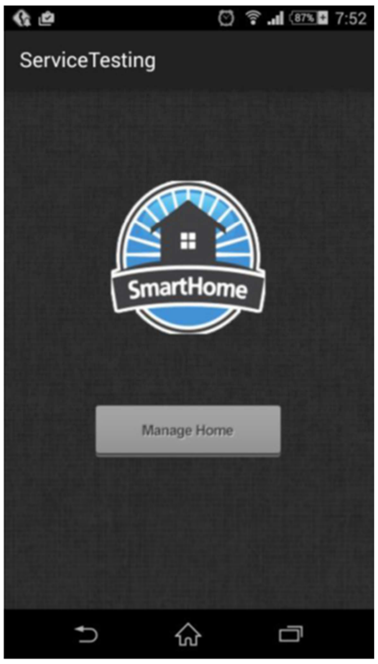


Figure 6. Start.

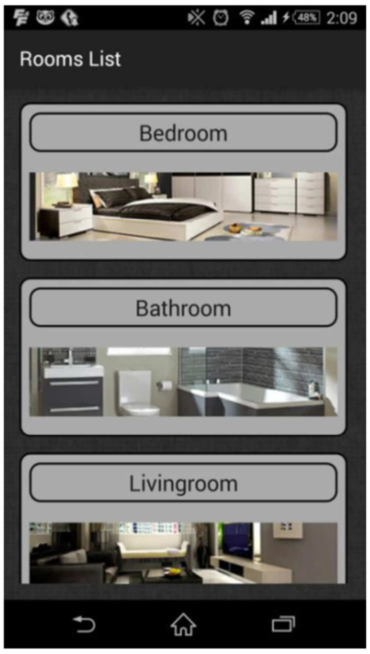


Figure 7. Rooms list.

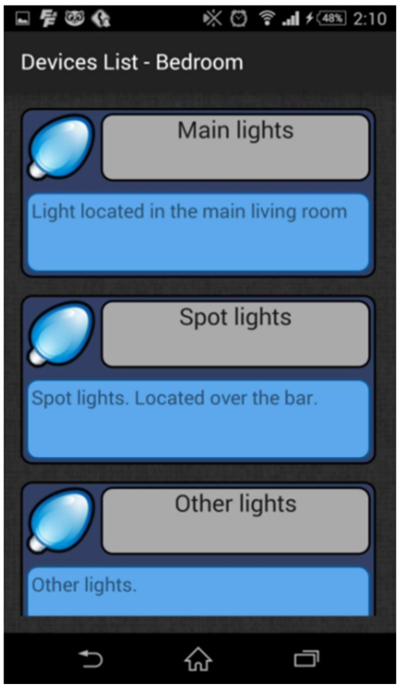


Figure 8. Device selection.

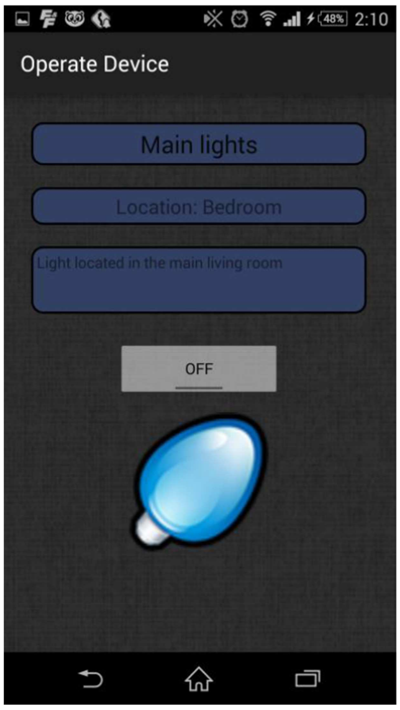


Figure 9. Device control.

The server of this work is implemented with two classes: “DeviceWebSocketServer.java” and “DeviceSessionHandler.java”. The first class includes the basic four main methods of communication using the Java WebSocket annotations @onClose, @onOpen, @onErrorand @onMessage. It also includes the @Inject method, whereby the second class is introduced. The second class includes all functions that take place during the client-server communication.

Control of the connected devices on GPIO ports of Raspberry Pi platform achieved through PI4J library [16]. 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 application runs on GlassFish 4.1 server. For the whole development phase was used the environment NetbNetbeans IDE 8.0.1.

5. Conclusions

In this paper we have proposed and implemented a low cost system that will ensure the intelligent control of a house upon user authentication. We accomplished this through the utilization 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.

References

[1] http://www.raspberrypi.org/.

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

[3] http://tools.ietf.org/html/rfc6455.

[4] G. Kortuem, F. Kawsar, D. Fitton, and V. Sundramoorthy, "Smart objects as building blocks for the internet of things," *Internet Computing, IEEE,* vol. 14, pp. 44-51, 2010.

[5] http://whatis.techtarget.com/definition/Internet-of-Things.

[6] Daniel Castellano, Jose Maria Canas, “Home automation system with web interface in the JdeRobot framework”, XV Workshop of physical agents, June 2014, Leon Spain.

[7] http://en.wikipedia.org/wiki/Home_automation.

[8] Soundhar Ganesh, Venkatash, Vidhyasagar, Maragatharaj, “Raspberry Pi Based Interactive Home Automation System through Internet of Things”, International Journal for Research in Applied Science & Engineering Technology (IJRASET), Vol: 3, Issue 3, March - 2015, pp: 809-814.

[9] Bharath Lohray, “Home automatation using Raspberry Pi”,http://www.linuxjournal.com/content/home-automation-raspberry-pi?page=0,2.

[10] Dr. S. Kanaga Suba Raja, C. Viswatnathan, Dr. D. Sivakumar, M. Vivekanandan, “Secured Smart Home Energy System (SSHEMS) Using Raspberry Pi”, Journal of Theoretical and Applied Information Technology, 10 th August 2014. Vol: 66 No.1, pp: 305-314.

[11] P Bhaskar Rao, S. K. Uma, “Raspberry P I Home Automation with Wireless Sensors Using Smart Phone”, International Journal of Computer Science and Mobile Computing, Vol.4 Issue.5, May- 2015, pp. 797-803.

[12] Syed Anwaarullah, S. V. Altaf, “RTOS based Home Automation System using Android”, International Journal of Advanced Trends in Computer Science and Engineering, Vol: 2 No.1, 2013, pp: 480 – 484.

[13] DIY Hacking, “IOT Based Raspberry Pi Home Automation using IBM Bluemix”, http://diyhacking.com/raspberry-pi-home-automation-ibm-bluemix/.

[14] https://tyrus.java.net/documentation/1.10/user-guide.html.

[15] http://developer.android.com.

[16] http://pi4j.com/.