INTERNATIONAL
HELLENIC
UNIVERSITY

Security and privacy in Internet of Things

Student Name: Aikaterini Kokoliou

SID: 3301150003

SCHOOL OF SCIENCE & TECHNOLOGY

A thesis submitted for the degree of

Master of Science (MSc) in Mobile and Web Computing

DECEMBER 2017

THESSALONIKI – GREECE

Security and privacy in Internet of Things

Student Name: Aikaterini Kokoliou

SID: 3301150003

Supervisor:	Prof. Kalloniatis
Supervising Committee Members:	Prof. Kalloniatis
	Prof. Katsikas
	Dr. Baltatzis

SCHOOL OF SCIENCE & TECHNOLOGY

A thesis submitted for the degree of

Master of Science (MSc) in Mobile and Web Computing

DECEMBER 2017

THESSALONIKI – GREECE

Abstract

This dissertation was written as a part of the MSc in Mobile and Web Computing at the International Hellenic University.

The Internet of Things (IoT) is taking advantage of the Internet and other wireless technologies to make physical objects interact online. It is a promising technology and receives research and public attention in the recent years.

IoT is still very young and despite the inevitable progress in the field, it contains privacy and security risks. Researches and applications of the IoT are in an early stage and to preserve peoples' privacy, the security of IoT must be strengthen.

Following this path, this thesis proposes the use of a Raspberry pi as a security countermeasure. The Raspberry pi acting as a server, an access point and a role base model secures the communication between an android application and the sensors attached to it. Giving the opportunity to a mobile application to handle any kind of device attached. To demonstrate the proposed technique, a mobile application will be developed, a software component on the Raspberry pi to support the aforementioned functionalities and also hand-made printed circuits boards to simulate simple sensor functionalities.

At this point I would like to thank my supervisor, Dr. Christos Kalloniatis, for the encouragement, motivation, enthusiasm, guidance and advice he has provided in all time of research and writing of this thesis. I have been extremely lucky to have a supervisor who cared so much about my work, and who responded so promptly to all my questions and queries.

Aikaterini Kokoliou

17/12/2017

Contents

Table of Contents

Table of Figures

Table of Tables

1 Introduction

The motivation behind this dissertation was to study the available IoT devices along with the security and privacy risks and try to build from scratch a complete IoT solution. This solution will include the physical hardware devices, the front-end and communication software for the remote handling of the devices and the back-end software to support the whole solution.

1.1 Background

The background of this dissertation includes the study of the key characteristics and the architecture of IoT devices, the key characteristics of security and privacy in IoT and, also, the study of available IoT solutions.

1.2 Problem

The problem needs to be tackled is the security and privacy in IoT, since, despite the numerous research around this problem, it is not yet solved. For the solution proposed in this thesis, the security risks of the devices used will be handled by the Raspberry pi. Since the main idea is to connect all sensors and devices used to a Raspberry pi and then interact with them via the Raspberry pi. As a result, the problem will need to be overcome is to secure the communication between the mobile application and the Raspberry pi.

1.3 Goals

The first step of the dissertation is to design and produce a prototype of IoT devices that could be wired to a Raspberry pi. Secondly, the design of a software component that will command the connected devices and manage the communication with the mobile application. Furthermore, the introduction of a role-access model in the aforementioned software to control the users that will interact with the devices attached. Finally, the design of the mobile application already mentioned.

1.4 Conclusion

As a conclusion, the security and privacy enhancements introduced in the solution will be presented at the latest chapter along with future thoughts for improvements.

2 IoT Characteristics & Architecture

The study of the IoT characteristics and architectures reveal the building blocks of IoT and, the main facts that should be taken under consideration when building an IoT application.

2.1 Characteristics

The Internet of Things (IoT) is a network where different objects can sense, communicate and share information [1]. These objects are uniquely identified, embedded with electronics that have sensing, actuation and programming capabilities. They are also able to interoperate within the existing Infrastructure under specific communication protocols [2]. The state of these devices can change anytime from anywhere and by anything while any kind of data may be collected from them as well.

With IoT, Internet is not just a network of computers but a network of heterogeneous physical objects like vehicles, smart phones, toys, home appliances, medical instruments, buildings, wearables or even industrial systems all connected to communicate, share [3] and collect valuable information aiming to upgrade peoples' lives.

IoT does not target only at sophisticated electronic devices but everyday objects like clothes, food, animals that would be recognizable, addressable and controllable via communications means. These objects may obtain enough intelligence through their sensing and computational capabilities and may be able to operate with or without external control [4].

The goal of Internet of Things is to enable any device related to any service or any business, to be connected anytime, with anything and anyone ideally using any network and any path providing information or services of any kind [5].

The main IoT key characteristics that will be further analyzed below:

1. Connectivity
2. Interconnectivity

3. Heterogeneity
4. Dynamic changes
5. Enormous scale
6. Safety

2.1.1 Connectivity

Connectivity hides a dual explanation meaning. All the devices should be connected on a network and at the same time being able to communicate in order to receive and transmit data.

2.1.2 Interconnectivity

Apart from network connection, devices should also be able to interoperate in order to exchange information under a common data and communication infrastructure.

2.1.3 Heterogeneity

The devices that built up a small collaborative network are heterogeneous. That means that they have different hardware, and software, they serve different needs, they may be able to communicate using different means, but at the end all these should be able to interoperate under a unique umbrella.

2.1.4 Dynamic changes

The IoT should be able to handle dynamic changes as the state of those devices alters dynamically. The devices may be active or sleeping, connected or disconnected. Their location, speed or access point may change along with e.g. vehicles' movement, or even the surrounding conditions, affecting their performance and characteristics.

2.1.5 Enormous scale

All the different "things" coexisting under an IoT solution, need to exchange an enormous amount of information every second especially in safety-critical environments. As a result, the management of this amount of information is of a great importance and a major outcome of IoT revolution.

2.1.6 Safety

The benefits that IoT offers to the society and to everyday life are obvious. A great load of everyday tasks is handled more easily, upgrading peoples' communication, transportation and entertainment. No matter the benefits though, safety is a key issue that should be tackled since IoT has invaded into peoples' houses via multiple smart devices and in many cases even critical personal data are exchanged. Securing the endpoints, the networks and the data exchanged across, is of a great importance.

2.2 Architecture

The IoT definition and presentation of its characteristics give an idea of what IoT is about. To go deeper into this subject, it is important to analyze the main IoT architecture [6], which consists mainly of four (4) basic layers [7]:

1. Sensor Layer – Smart Devices (lower level)
2. Gateways and Networks
3. Management Service Layer
4. Applications Layer

Given the name of each layer an initial insight is illustrated regarding the building blocks of the IoT architecture. At the following paragraphs, an analysis of each layer will be given for a thorough understanding.

2.2.1 Sensor Layer – Smart Devices

The sensor layer is the lower level of the IoT stack (architecture). It mainly consists of sensors and smart devices (hardware and software) which represent the edges of the IoT. Nowadays there is a plethora of chipsets and platforms to choose from, that evolve everyday by integrating sophisticated software, allowing themselves to take decisions [8]. Examples of such devices are the Arduino platform, the Raspberry Pi, ARM and microchip micro-processors that are embedded to different electronic devices and many more.

Apart from complete devices with integrated software, a large variety of sensors allows us to sense almost anything by using them. For example, on the market are available temperature and humidity sensors that can communicated with a smart phone or be directly connected to a micro-processor, CO_2 detectors, flood or smoke detectors, motion and presence detectors, accelerometers GPS and many more.

The goal of IoT is the successful combination and collaboration of different sensors and devices to gather the maximum related information possible. To do so, apart from the hardware needed it is also important to manage connectivity. Connectivity is managed by the second layer of the IoT architecture stack, analyzed below.

2.2.2 Gateways and Networks

Massive volume of data will be produced by the sensors and the devices included in the lower level, so the key point is to establish and maintain a robust and high-performance network infrastructure, as a mean of transport. Current networks work under different protocols to support machine to machine communications and manage their applications. A lot of research is needed to unify the different communication protocols allowing any sensor and device combination.

To achieve the interoperability of the devices and the Internet connection, different gateways (microcontrollers or microprocessors) and various gateway networks can be used (WI-FI, ethernet, GSM etc.). Connecting any device to the network is an important step, followed by the secure management of information exchanged through analytics or process modeling.

2.2.3 Management Service Layer

The management service layer abstracts the lower levels in order to provide in the final layer (the application) only the necessary data, excluding unrelated information, unformulated data or sensitive personal information. It performs a three-dimensional filtering [7]:

- Service or business-related filtering,
- Data analytics, providing statistical analysis and real-time decisions
- Data filtering, reducing the risk of privacy disclosure of the data source.

Security must be enforced not only in the data manipulation but across all the dimensions of IoT architecture. Security of the system prevents system hacking and compromises by unauthorized personnel, reducing the possibility of risks.

2.2.4 Applications Layer

At the upper layer exists the application. It is the higher level of the IoT stack and is the point where user is allowed to interact with the system. The applications layer includes

different application platforms like mobile applications, websites or other online applications for personal or industry purposes. IoT applications have already captured multiple domains [9] such as:

- Transportation
- Culture and Tourism
- City
- Retail and Supply chain
- User Interaction
- Healthcare
- Environment and Energy and,
- Lifestyle,

making smarter and easier peoples' daily routine and working environment.

3 Devices and Applications of IoT

As already mentioned in the previous chapter, IoT applications prevail in multiple domains. In this chapter, available software and hardware regarding IoT will be presented [10]. Examples of webpages, mobile applications, IoT devices like smart watches, or smart electronic devices and even IoT developing boards and platforms will be analyzed to provide the basic idea of IoT solutions [11].

3.1 Smart Home

The smart home is likely the most popular IoT application since it is an affordable solution and already available to consumers [12]. One example is "the student thermostat" of the company Nest, purchased by Google, which is a copy of a common device, a thermostat, embedded with Internet access. It is more than just a thermostat; it knows if you are home or not to save your energy bills. It is connected to all heating devices reminding you even when you should change your air filter based on how many hours your AC has run. The mobile application is shown at Figure 1.

Figure 1: Nest – A smart Thermostat project [13]

In the same category, Apple has launched a series of devices, examples of them are presented in Figure 2, that support HomeKit. HomeKit sets strict standards for how the smart home devices will interact with each other and with a personal assistant - Siri.

Figure 2: HomeKit – Smart Home by Apple [14]

There are sensors that measures temperature, humidity or in general air quality and through HomeKit software can adjust the living parameters for you. At the same time Schlage Sense Bluetooth Deadbolt Siri can control door locks if you simply “talk” to him. Diving deeper into the field of Artificial Intelligence more examples arise. Like Cubic, a voice activated personal assistant which can sync with lots of smart systems in your home and your Android or Apple phone. The name of this device clearly corresponds to its shape as it is shown at Figure 3.

Figure 3: Cubic – A personal assistant [15]

3.2 Wearables

Watches are no longer just telling time. The Apple Watch, as shown at Figure 4, and other smartwatches on the market have turned our wrists into smartphone holsters by enabling text messaging and more.

Figure 4: Smart watch by Apple [18]

Apart from watches another example of a wearable IoT device is the "Jawbone Up" which is a complete fitness tracking kit. It may look like a simple bracelet, as shown at Figure 5,but it is loaded with a variety of different sensors. It helps you measure steps, distance, calories, heart-rate, sleep duration and allows food and drink loading. It comes with a smart coach which can help you review performance. Finally, being connected to the Internet allows community-based data sharing [19].

Figure 5: Jawbone up – An activity tracker [20]

A different device that can help you reduce your stress levels using meditation is the "Muse Headband", presented at Figure 6. The brain sensing headband helps you get the

most out of your meditation practice by giving real time biofeedback of what is going on in your mind. When you put the Muse headset on, you are asked to complete breathing exercising according to the sounds you hear which indicate how focused and calm you are. If your mind is too active, the Muse gives you feedback to clear your thoughts.

Figure 6: Muse headband – A meditation option [21]

3.3 Smart Cities

The IoT has the potential to transform entire cities by solving problems that citizen face daily like traffic congestions, available parking slots, reduce noise, or pollution. Example given, the "Luminext" [22] a smart street lighting solution. Luminext offers sustainable solutions for conventional, static dimmable and dynamic outdoor lighting keeping the users of what is going on in their area. A different example is a smart parking solution, the "Streetline" [23]. Streetline utilizes hardware data with software analysis to power smart parking applications with a flow of occupancy data for effective parking policy decisions and parking guidance. A great example of a smart city solution is the "Enevo" [24] which deals with Waste management. Enevo's waste analytics solution allows you to track your waste generation and get detailed information on how much you are diverting from the landfill by recycling.

3.4 Prototyping Platforms

Apart from the available solutions on the market, there are a lot of development boards and platforms available, upon which programmers can build applications taking advantage of their features. Most of those platforms faced rapid development the latest years as the demand for smart solution increases. An example of such a solution is the Arduino Uno. It remains at the top of the list for both beginners and experts. It is one of the first microcontroller-based development boards. The Arduino Uno R3, shown at Figure 7, is the simplest yet the most powerful prototyping environment. It has digital input/output pins and six analog ones giving the opportunity for connections with multiple sensors and other digital devices. It also comes with a 32KB of Flash memory accommodating code for complex operations.

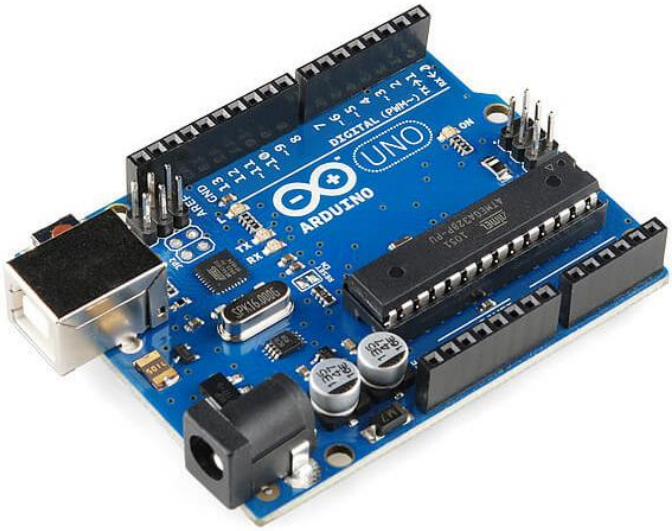


Figure 7: Arduino Uno [25]

Apart from Arduino, Raspberry Pi is another popular platform [26] used for educational purposes and by non-technical or skilled users. The latest model is presented in Figure 8 and is called “Raspberry Pi 3”. It includes a built-in Wi-Fi and Bluetooth module, making it the most compact and standalone computer. It is a powerful platform considering its characteristics and has forty GPIO pins where different sensors, peripheral and accessories can be connected. Raspberry Pi runs on a customized Debian Linux called Raspbian, which provides an excellent user experience. For developers and hackers, it offers a powerful environment to install a variety of packages including Node.js, the LAMP stack, Java, Python and many more.

Figure 8: Raspberry Pi 3 [27]

Finally, and for more advanced IoT projects, another option is “Intel Edison” powered by Intel [28], a screenshot of this device is given at Figure 9. It is also integrated with a Wi-Fi and Bluetooth module, it has twenty digital input/output pins, six analog inputs supporting UART and I2C protocol while it is further compatible with Arduino Uno.

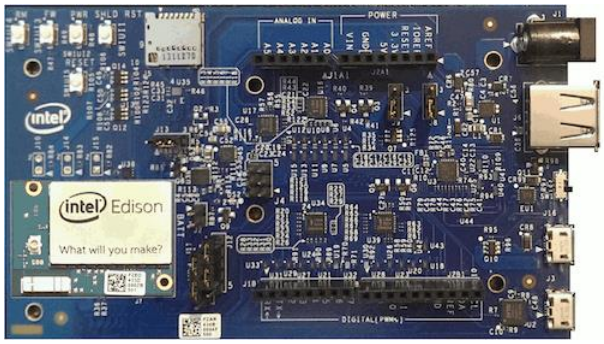


Figure 9: Intel Edison [28]

The above mentioned IoT applications, devices and platforms are just a few examples of the available solutions on the market that worth to be mentioned.

4 Security and Privacy in IoT

In this chapter the main characteristics of security and privacy will be analyzed along with possible threats specifically in IoT area [10]. Security and privacy are two concepts closely related, but in fact there are important differences between them. Privacy relates to the persons. Assuring privacy gives persons the opportunity to keep full control of personal information they disclose to an application [11]. While, security refers to data and the effort to guarantee them from multiple aspects that will be presented in the following chapter.

4.1 Security characteristics

The aspects that complete the security of data or else the key security characteristics existing in any field are:

- Confidentiality
- Integrity
- Availability
- Information Security
- Non-repudiation
- Reliability
- Authenticity
- Access Control

For the area of IoT, the aforementioned characteristics could be grouped into two major security aspects, the flaws of sensors, RFID cards or other small IoT devices in general and the security in the propagation area.

In the effort to reduce the cost of sensors and RFID or NFC tags that are used in IoT projects, it is difficult to integrate adequate security abilities. Sophisticated encryption could raise their price making them unaffordable especially when included in applications used in peoples' everyday life. Most IoT projects are based in the use of any kind of sensors in order to gather information. This fact complicates the process of securing IoT applications, since, those devices may be prone to different security risks. Apart from the

physical insecurity of those devices, the risk rises when any application or other device needs to access them to collect information. In any wireless sensor network, exposure of the wireless signal leaves an open door for illegal listening.

Leaving aside the security of the sensors and their accessing method, the security in information propagation network is also an important aspect. Large amount of data generated from the sensors needs to be propagated, processed and controlled in the back-end. Consequently, apart from securing the sensors, the security of the back-end system also needs to be re-assured using authentication and encryption techniques.

To conclude, security in IoT is a complicated issue since it has many different aspects to take into consideration: the number of different devices, different technologies and different communication means that need to co-operate. Securing each device, each kind of communication and the information processed requires a system that possesses all the key characteristics mentioned, and further analyzed [29] in the following section.

4.1.1 Confidentiality

Information exchanged between any device of an IoT system or stored in any device is kept secret or private from unauthorized users. The confidentiality in other words ensures that the exchanged data during a communication are kept confidential apart from authorized entities. Confidentiality is generally ensured through encryption of the communication line where data are being exchanged.

4.1.2 Integrity

The state of maintaining internal accuracy and consistency or lack of corruption regarding the data saved, displayed or exchanged under the umbrella of an IoT system. Integrity ensures that the exchanged data are not altered by unauthorized entities during a communication procedure. Message Authentication Code can be used to provide this property.

4.1.3 Availability

Ensures that data are always accessible and available upon demand by an authorized user. That suggests that the communication system has to remain accessible and functional despite failures or attacks as for example the Denial of Service attack. Backup systems and/or secondary uninterruptable power units can be used as an asset to ensure availability.

4.1.4 Information Security

The preservation of the above three mentioned characteristics: confidentiality, integrity and availability for any essential information exchanged or used in an application. From the three characteristics above we conclude that a set of different measures are necessary to assure information security.

4.1.5 Non-repudiation

The ability to prove the occurrence of a claimed event or action and its originating entities. In other words, non-repudiation ensures whether an entity has actually taken part in an exchange of information which means that the entity has actually send or receive the information.

4.1.6 Reliability

Property of consistent intended behavior and results. The whole IoT system should be reliable, meaning that it should behave as expected, manage all requests on time and respond to them avoiding any information leakage.

4.1.7 Access Control

A composite characteristic that ensures any entity taking part in the system is authorized to be part of it, and that any protected information is accessed only by authorized users of the system. This complex characteristic can be ensured through three individual simple characteristics. One of these characteristics is the Identification which is the proof that each entity is who or what it claims to be. The second one is the Authentication which is the procedure through which this claim is verified. And the third, the Authorization which allows to determine which information can be accessed, with what action and by which entities. For example, it is important that a mobile application user is a real human being using his smart phone to interact with a web-server and not a robot wishing to intrude into the system. Or that the server which replies to a mobile application is the designed one and not an attacker in the middle. A user access table could be used as a role access model to assign access rights to entities of a system.

4.2 Privacy Characteristics

Security is necessary to ensure privacy but not sufficient, therefore privacy is a key point in IoT as well. Even a small RFID tag embedded in smart card could be tracked inside a mall for example, making possible for an owner to track peoples' habits and preferences according to their position. Also, multiple IoT applications mentioned in the earlier chapter measure and store personal information like the heartbeat, the blood type, personal habits, credit card details etc. So, security measures in such cases are necessary to assure the privacy of sensitive personal data.

In the following subsection the key characteristic of privacy [30] will be presented to highlight the points that need to be taken under consideration when building an application.

4.2.1 Anonymity

The situation where a subject (for example a user of an application) stays anonymous with respect to a possible attacker. It is important that the real identity of the entities involved in the IoT will be kept secret.

4.2.2 Unlikability

The situation where a possible attacker cannot distinguish whether two parts of a system are related or not. Meaning that if a possible attacker enters the system, he will not be able to combine the actions taken with the entities actual identities. So even if the attacker manages to capture an action he will not be able to identify the source or the destination entity or in other words the attacker will not be able to group actions taken by the same user.

4.2.3 Undetectability

The situation where a possible attacker cannot distinguish whether a subject exists or not. Or else the state of data being exchanged (messages) in a system cannot be discernible from no message in the communication line of the system.

4.2.4 Unobservability

Unobservability ensures that when a user may use a service, others especially third parties are not able to understand that this resource or service is being used. Unobservability

ensures that any subject attacker or even entity of the system cannot distinguish whether an operation is being performed.

4.2.5 Identity management

The situation where each subject has different access rights in a system based on how the person was authenticated by the system.

In the following chapters the developed IoT solution will be presented, the architectures and technologies used and the security and privacy enhancements added to secure the solution to the maximum extent possible.

5 Developed Project

In this chapter all the aspects of the developed project will be presented starting from the motivation behind the project and finally reaching the presentation of the developed application in pictures that reflect the latest developed version of it and the presentation of a real-life picture that reflects a possible state of the all the systems components.

5.1 Motivation

The motivation behind this dissertation was the development of a complete IoT solution. A solution that will cover all levels of the IoT architecture as they have already been demonstrated in the previous chapter.

To be more precise, one of the goals was the design and production of the devices that will act as sensors and actuators. Studying the IoT devices that exist in the market, the main conclusion was that it is difficult to increase their safety and privacy issues trying at the same time to keep their cost low. Because the cost increases along with the effort to embed encryption techniques and sophisticated software in a device. As a result, the solution proposed is to design simple sensors and actuators that could be easily attached to a Raspberry Pi. The Raspberry pi would be responsible to handle the security threats while all the other devices will be attached to it using wires. It constitutes the ideal solution since it has a number of pins where the designed devices could be connected directly and also offers the possibility of installing almost any operating system that allows the installation of the software required for the co-ordination and the security of the final solution.

This idea reduces the cost of the IoT devices and, also, gathers the insecure edges of our system (different devices) in one, the Raspberry Pi. The Raspberry Pi is a low cost IoT development board that supports numerous of software packages. Another advantage of this approach is that, not only the security edges are reduced but also, the effort to communicate with all the different devices.

The disadvantage though is that only sensors that are already compatible with the Raspberry Pi could be used or sensors that will be designed to be compatible. Fortunately, the

market has a variety of sensors available and, also, the design and production of such small and simple devices is relatively low. Within the framework of this dissertation small PCBs will be designed having LEDs and switches as a proof of concept for the proposed solution.

Designing and producing the sensors was the first step towards the development of the final IoT solution. The next one was to decide, design and build the remote-control system of these devices. The Raspberry Pi, as already explained, should be able to handle the connected IoT device. At the same time, it should offer the possibility of remote control since the Raspberry will be installed somewhere near to the aforementioned devices.

A widespread tool available for remote control of the Raspberry Pi pins is the Apache server. The Apache server is a package available for Raspberry Pi 3 that allows remote control via a PHP script that can be accessed by the server any time. Through the PHP saved on the Raspberry Pi, Apache server is possible to handle remote requests and, also, handle the GPIO pins of the Raspberry. That is the main software component on the Raspberry Pi that should be responsible for handling the connected devices.

The last component necessary to complete the solution was the development of an application that is able to interact with the server mentioned above, gaining access on the IoT devices. For this purpose, a mobile application was developed in Android while for the communication between the Raspberry Pi and the mobile Application HTTP POST requests were exclusively used.

The final objective of the dissertation was the use of more than one Raspberry Pis to offer the option of extra connected devices with multiple different purposes and, also, the security and privacy enhancement in all the aforementioned components to the maximum possible extent. In the context of this dissertation, two different Raspberry Pis 3 were used and two different boards with LEDs and switches were produced as a proof of concept. The key enhancements towards security which will be further analyzed in consequent chapters are:

- Wire Connection of the devices
- Access Point that limits the users willing to interact with the Raspberry Pi
- Meaningless messages for the communication between the mobile application and the Raspberry Pi using HTTP POST over HTTP GET method
- A Role Access Model (MySQL Database) to handle the user access rights
- Error connection logging in the Raspberry Pi

- No tracking of user actions

5.2 Basic Functionality

The project aims at developing a secure IoT solution for handling sensors and accessories remotely via a mobile application, named “RaspIoT”, an abstraction of the whole solution is given at Figure 10. The mobile application will be developed in Android for the initial project while a future target is to make the solution platform-independent. The IoT solution could be used practically for any kind of sensor, switch or accessory that is compatible with the Raspberry Pi 3, and as a proof of concept the connected devices will be a set of LEDs and switches. The sensors will be attached to the Raspberry pi and their state will be updated or read via application driven events. These events will be captured by the Apache web server running on the Raspberry pi that will judge whether an action should be executed or not according to the access rights of each mobile app user. The access rights of all users will be saved in a database managed by the Raspberry pi web server and handled by an admin user.

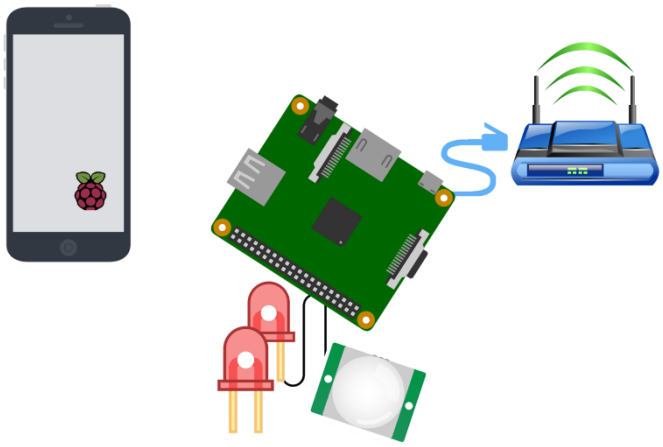


Figure 10: IoT Solution

The Raspberry Pi will be transformed, using the necessary libraries, into an access point in order to provide Internet Access to the Smart Phone users.

At the same time, the mobile application provided together with the Raspberry Pi, as a single package, will include all the necessary settings to allow remote connection with the designed devices. The user access model mentioned before, and other developed security mechanisms will handle undesirable connections and malicious requests to assure,

to the maximum extent possible, the security of the developed solution and the privacy of the users' data.

5.2.1 Application Mock up screens

The figures presented below represent the application screens as shown on the mobile phone. The xml code that correspond to each figure from Figure 12 to 15 can be found in the Appendices chapter [B] along with the Java code [A]. The screenshots are presented in sequence in order to produce a use case scenario where an enrolled user changes the state of two LEDs attached to Raspberry1:

1. The user selects the developed Application called "RaspIoT".

Figure 11: User selects the RaspIoT app icon

2. The user selects the desired Access Point (Raspberry1) by clicking the preferred List view item.

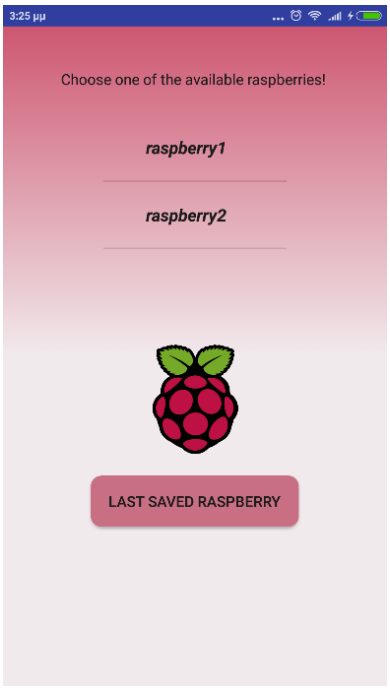


Figure 12: Initial Screen

3. Clicking “connect” the application automatically disconnects the mobile phone from the connected wi-fi and connects it to the selected network.
4. The next step is that the user presses the “connection setup” which actually saves the IP that corresponds to the selected Raspberry Pi and takes the user to the next screen.

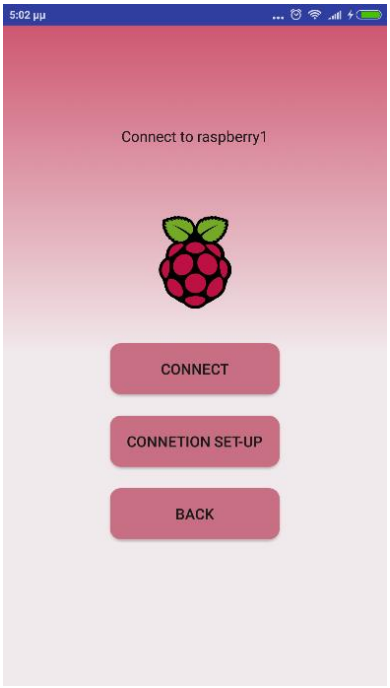


Figure 13: Raspberry1 was selected

5. The user may now enroll himself to the Database of the IoT system hosted on the main Raspberry Pi server. In order to enroll, the user types in an email and a password.

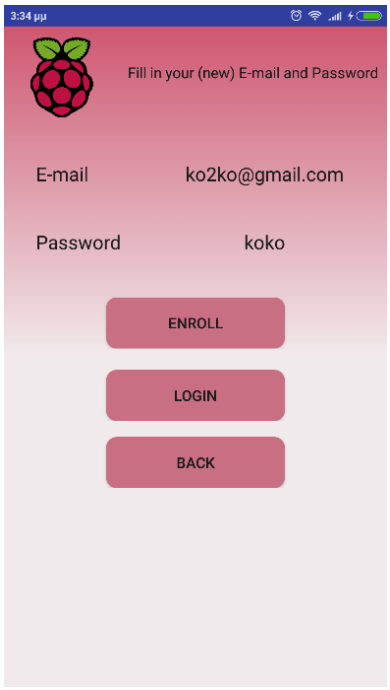


Figure 14: Set-up was completed, user may login

Initially the access rights of the user are limited to only one LED in order to provide an optical verification that the procedure was followed correctly. When the administration of the Database will grant full access rights to the user, he will be able to handle all the devices attached to the Raspberry pi.

6. At this point, assuming the user has login and has full access, all interactions with the Raspberry Pi's connected devices are available.

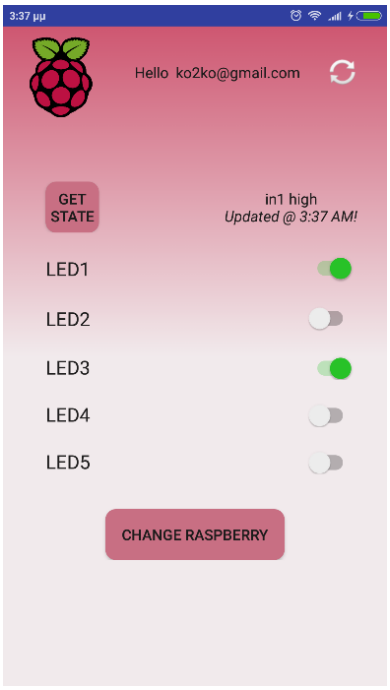


Figure 15: Last Screen (Basic Functionality)

7. Supposing the user changed the state of LED1 and LED3, as indicated at Figure 15, the effect on the connected devices of the Raspberry1 (Main Raspberry) is shown in the Figure 16 bellow.

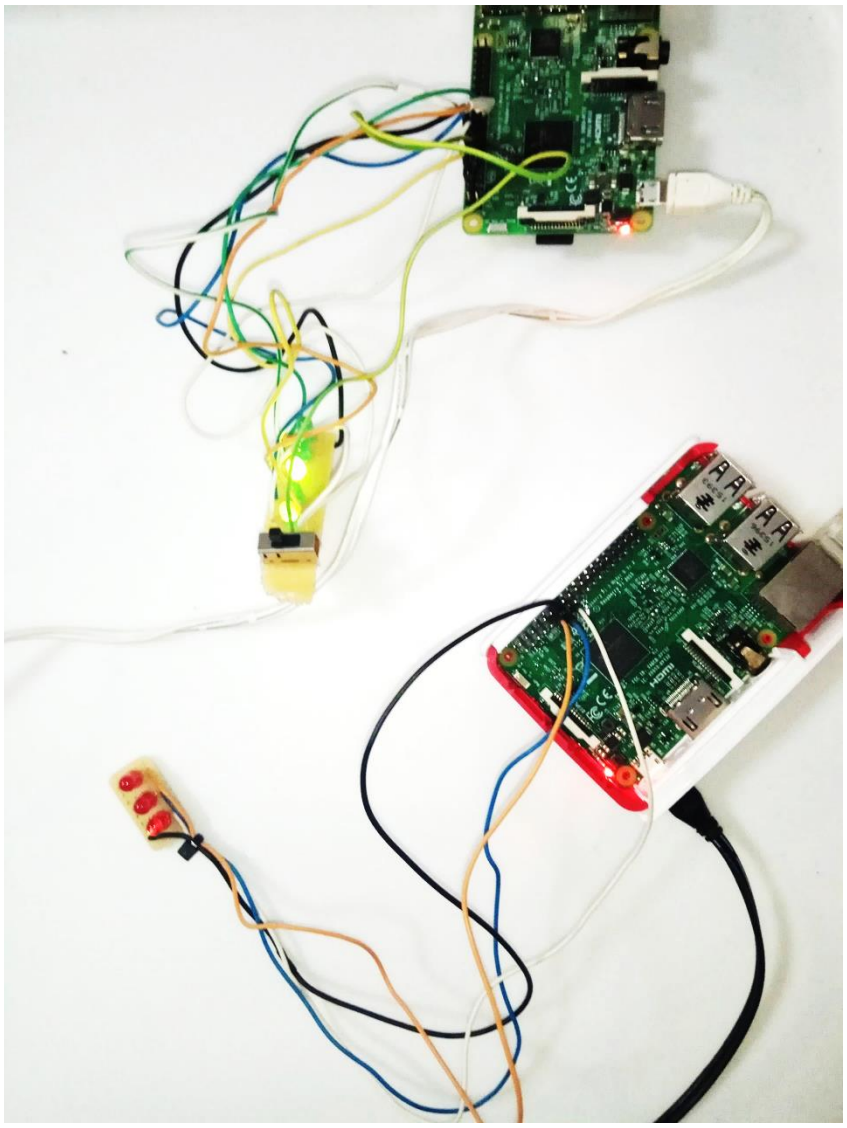


Figure 16: Connected LED 1 and 3 to Raspberry1 are turned on.

8. The user has also the option to switch Raspberry Pi in case more than one are available on the local network.

5.3 Architecture & Technologies Used

To describe the proposed IoT solution, a component diagram will be presented along with a description of key-points of the internal architecture.

5.3.1 Component Diagram & Electronic Designs

The component diagram presented at Figure 17 is a visualization of the whole IoT solution which is composed of three main parts:

- Mobile Application
- Raspberry Pi, and the
- Designed Boards

To describe the whole architecture, the end-points will be described first. The custom sensors are designed using an online tool the CircuitLab and then their prototypes are soldered by hand.

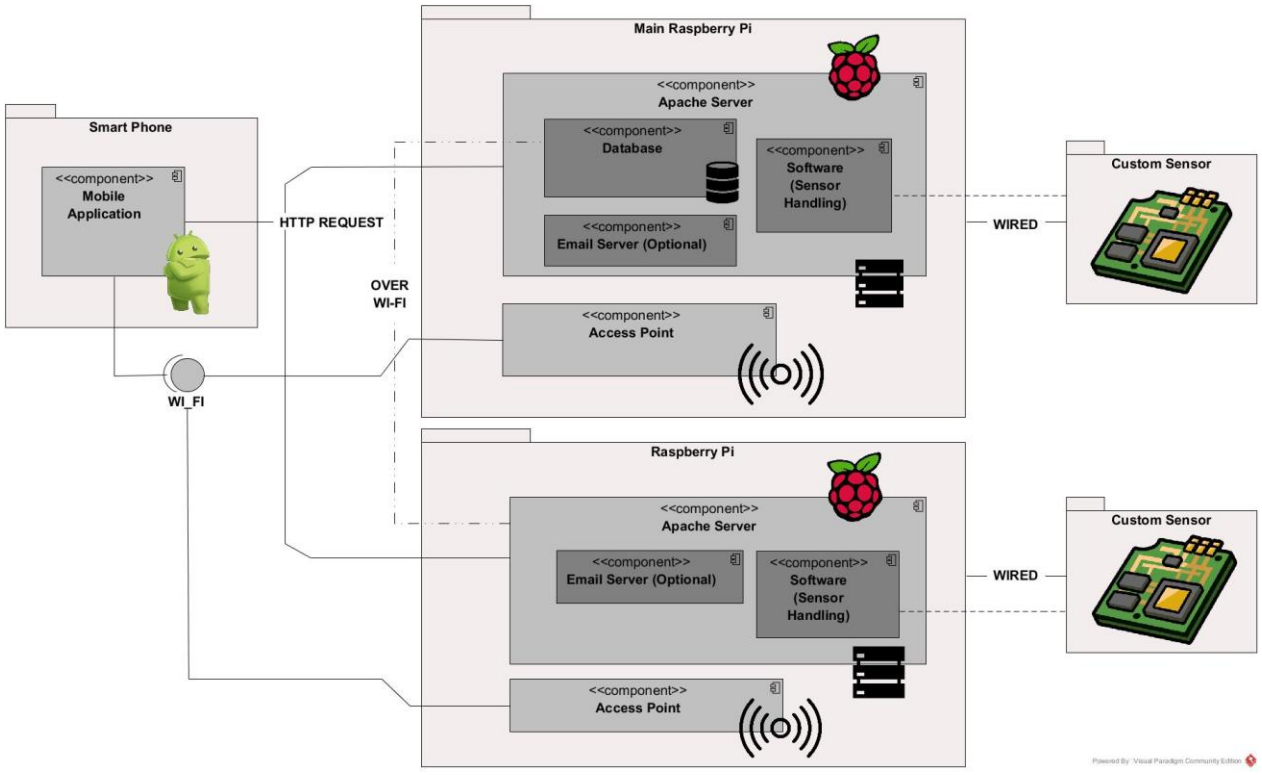


Figure 17: Component Diagram produced on Visual Paradigm

Two different boards were created (one for each Raspberry Pi), since each Raspberry can act upon the devices attached to its pins. The main Raspberry was attached to CustomBoard1 (Figure 18) and the other on the CustomBoard2 (Figure 19). As shown at the figures below, the CustomBoard1 has 5 LEDs and 1 switch, while the second one has

only 3 LEDs. Each LED represents an output of the Raspberry Pi and the switch a digital input. The information gathered from the input of the Raspberry works as input for the mobile application while the outputs are driven by commands also initiated by the mobile application. The mobile application drives remotely the state of the LEDs while it can also ask to be informed about the state of the switch.

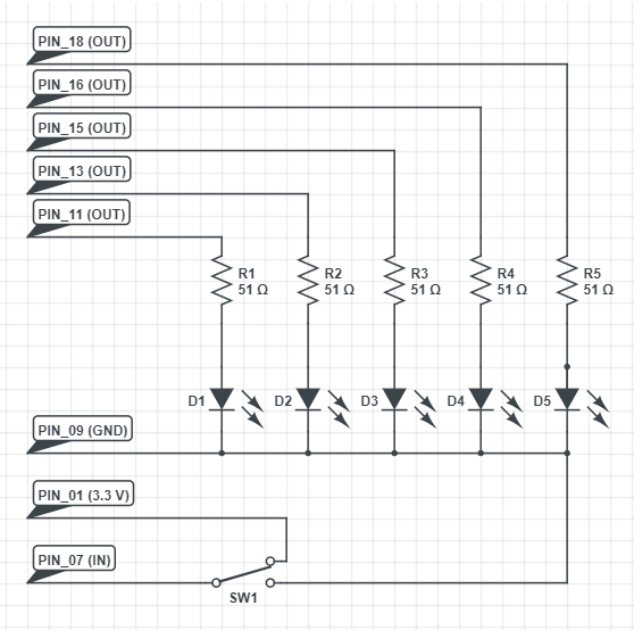


Figure 18: CustomBoard1 (Main Raspberry)

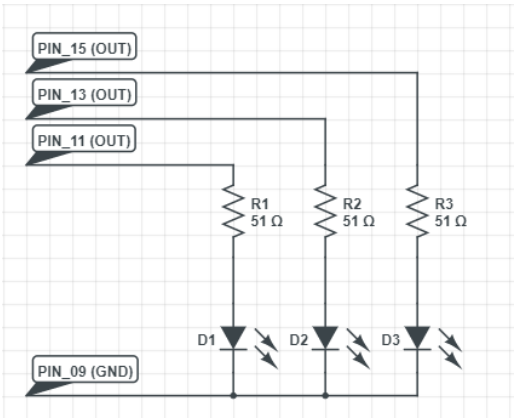


Figure 19: CustomBoard2 (Second Raspberry)

The labels on the above two designs name the pin numbers of the Raspberry pi 3 pinout. On the following figure, Figure 20, the pinout is presented giving the idea of how the devices are connected to the Raspberry pins.

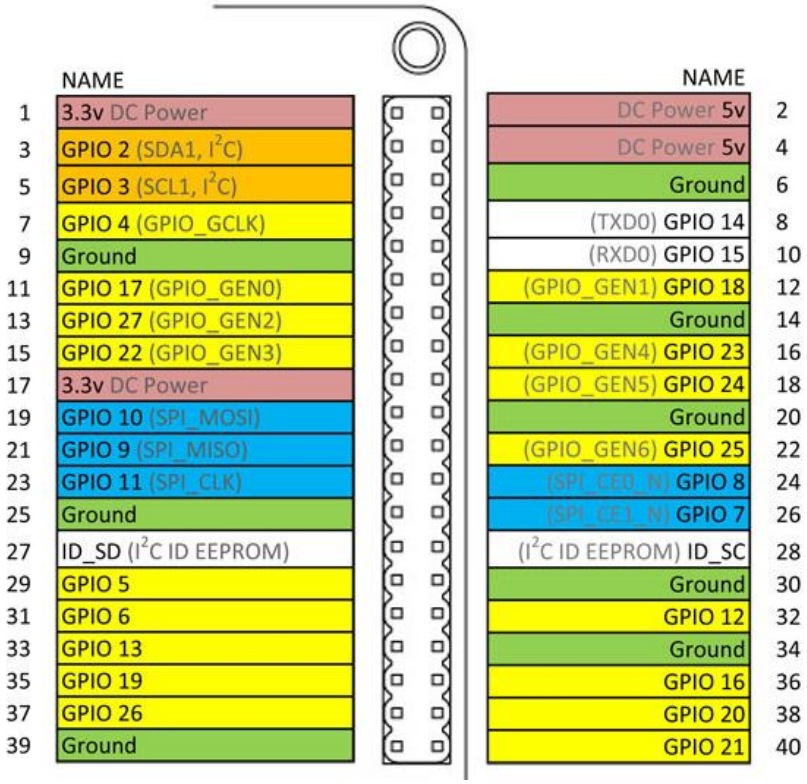


Figure 20: Raspberry Pi 3 Pinout

As already mentioned in the functionality of the solution, the devices are remotely driven by the application and this is possible at the last screen of the mobile application (Figure 21).

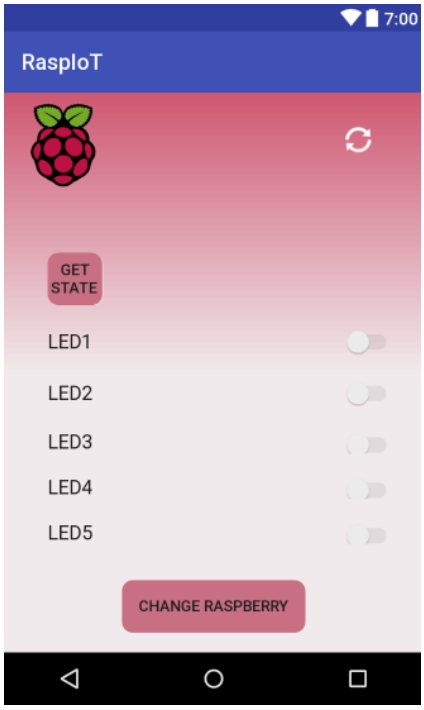


Figure 21: Application Last Screen

The GET_STATE button informs the user about the state of the switch which could be ON or OFF since the switch belongs to digital two-states inputs. The switch-button on the left of each LED tag control the state of the LEDs shown at the aforementioned electric designs. Changing the state of a switch, the state of the LED changes as well. The LED and the switch are two typical digital I/O that work as a proof of concept for this dissertation. Any other digital device could be used in their position but also any other type of device compatible with the Raspberry pinout. Figure 22 below gives a more realistic view of the connection between the Raspberry Pi and the devices used.

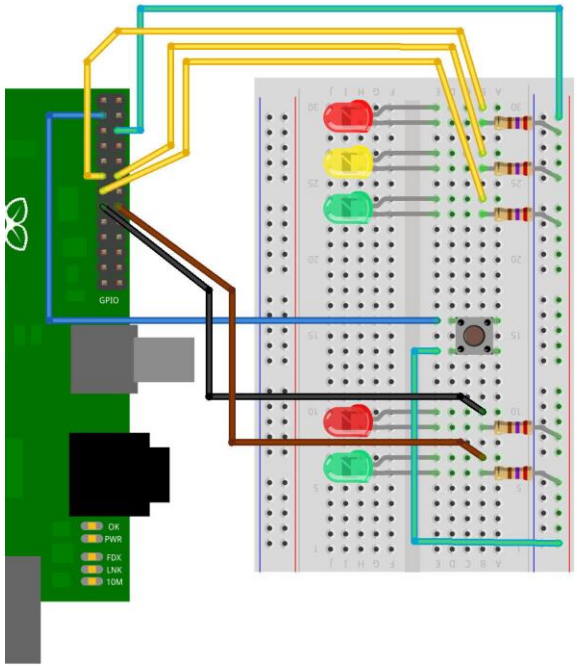


Figure 22: Initial testing on breadboard (before soldering)

To allow the mobile user act upon the connected devices it was necessary to design and develop the communication component that will handle the messages along the mobile application and the Raspberry. This component is the Apache Server and will be further analyzed along with all the features used for the proposed IoT solution.

Apache is a web server software developed by Apache Software Foundation and is an open source software available for free. It can be highly customized to meet the needs of many different environments by using the available extension and modules. The main web server functionality is to check for the web page requested and fetch it. It also serves the fetched web page, handling the communication with the website and the requests made, while it is also responsible to handle and clean the memory used (cache, modules).

So basically, a web server is the software that receives the request to access a web page. It runs a few security checks on the HTTP request and brings the web page asked. Then depending on the page asked, the page may ask the server for extra modules while generating the requested document, and then it serves the document initially requested.

In this dissertation, the mobile application sends HTTP POST request to a web page in php, stored in the Raspberry pi apache directory. The php script includes an initial check to identify whether the requester is connected to the provided Access Point. Each Raspberry Pi emits a different access point that is active to a limited radius around the Raspberry pi. In this way we ensure that the users of our system are close to the Raspberry pi (inside a house, a room etc.). The password of the Raspberry pi is not available to the public, it is encrypted inside the code of the mobile application. The password is also stored in the Raspberry pi in a non-text file. It is difficult for an attacker to identify the password even if he gains access to the Raspberry pi because all files in the Raspberry including essential passwords are being protected with a different password. At the same time, inside the code of the Android application the password is divided in two parts and then both these parts are encrypted using shift Cipher a type of the Caesar Cipher. The time needed to crack the password may not worth the gain, since knowing this password does not assure to the attacker the use of the final system.

Apart from this first security enhancement, the web server further checks the access rights of the user. It necessary for a requester to enroll in a database hosted in the server. At the time of the first enrollment the access rights are by default equal to zero. Then the administrator of the system should log-in to the database using the php-my-admin tool. To access this site, the admin should be using a computer connected in the local network and use the password and username especially created for this purpose. It is not possible for any attacker outside the local network to connect to the database and alter the access rights. This database works as a role base access model for our IoT solution. At this point, two level of access rights exist: the default one, or else the zero level which is limited to the one LED, and the full access, or else the level one which gives access to all the devices attached to the Raspberry Pis.

While in each Raspberry pi there is a unique Apache web server that handles the HTTP, requests arriving for the devices attached to it, there is one database hosted only on the main Raspberry Pi and both the servers check the access rights of the users enrolled to this database, having a unique point of reference. The database saves the email of the

requesters, their password, their access rights and it assigns to each enrollment a unique id. When a user enrolls or logins, the mobile application sends the email and the password via a POST request, at the reply the requester get its unique number which is therefore used in all the subsequent requests. In this way, it is avoiding the use of essential information in POST requests. The user is also prompted to use an email that is not connected to any other service and a password never used before.

It is worth to mention at this point that the HTTP requests used for the proposed solution are only HTTP POST. The POST requests are more secure than GET requests as it will be described at the security enhancement chapter that follows.

Another module used for the purposes of the IoT solution developed is an email server established on the Raspberry pi. For this purpose, an email account was created, and its credentials were saved inside the php script. Whenever a new user enrolls in the system, the server informs the admin by sending an email to his account. The admin can therefore log in to the database and change the access rights if needed. In case the user is not known to the admin, he may as well delete the entry keeping the solution secure. In case the user does not want to have an email account for this purpose, then this option can be disabled.

The last component that completed the IoT solution architecture was the mobile application. The application for the time being is developed in Android and a future goal is to develop the application for all the different platforms. The developed Android application follows the MVC (Model-View-Controller) framework, a visualization of this model is given at the Figure 23. The architecture choice is important in order to ensure the quality of implementation in the long term and reduce development effort. The MVC is a standard software framework that separates the user interface (View) and the business rules and data (Model) using a mediator (Controller) to connect the model to the view [31].

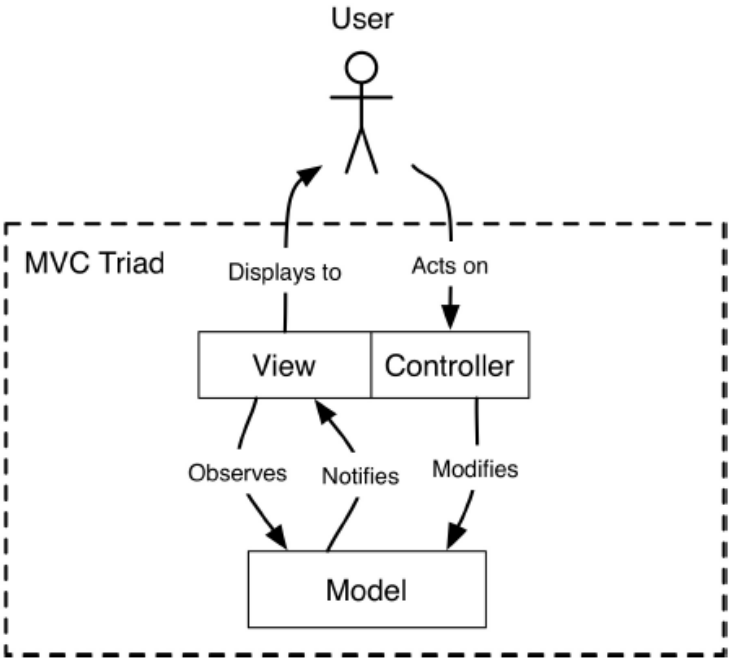


Figure 23: MVC model [31]

The main benefit is the separation of concerns since each part takes care of its own work. The view takes care of the user interface, the controller takes care of the data and the controller sends messages between both. The controller provides the data model to view and any changes to the controller are transparent to the view while UI changes will not affect the business logic and vice versa. Android uses an MVC pattern having XML files acting already as a view. In the previous chapter all the xml files were presented in figures showing the UI of the application. However, the use of the XML file is not enough, the real separation is that a user interaction on the screen initializes a method that then handles the data. There is a method related to each button and/or switch on the interface separating it from the related data (Model).

The most important library used was the OKHTTP, a library necessary to prepare and send the HTTP POST requests to the Raspberry server. The POST requests are created in an unconceivable way: random numbers, letters and expressions are used in order to make impossible for an attacker to understand the meaning and the purpose of these requests. The code written for the development of the mobile application can be found in the Appendices [A , B and C]. Then in the server side, the php script de-compose these requests and update the state of the connected devices or informs the android user about their state. Those scripts could be found also in the Appendices [D].

On the following chapter, the set-up procedure for all the different components of the IoT solution will be explained in detail.

5.4 Project set-up

In this chapter all the necessary set-up procedures for the integration of the external libraries and packages will be analyzed. Firstly, the back-end will be presented (Raspberry – Pi set up) and then the front end (Android Mobile Application) while at the end all the security features and mechanisms will be highlighted as well.

5.4.1 Raspberry Pi set-up

The Raspberry Pi, as already presented in the previous paragraphs, is a small computer almost the same in size with a credit card. The Raspberry Pi 3 which is the latest model available in the market, is able to run Linux or a limited Windows edition but it also has its own operating system, Raspbian. There are other options as well, but they are mostly used and made for specific projects.

Raspberry Pi is not a ready-to-go platform, the image of the chosen operating system need to be copied in an SD card that will be fitted afterwards into the Raspberry Pi before a user is able to use it as a common Desktop. The procedure followed for the Raspbian installation will be provided in single steps below.

There is always the option to use a pre-installed SD card that has already installed the necessary NOOBS software, alternatively NOOBS is available for download at the official Raspberry Website [29].

NOOBS Installation

To set up the initially blank SD card with NOOBS the following steps were needed:

1. Format of the SD card in another PC
    a. WINODWS: Format was handled using the Association's Formatting Tools, which can be downloaded from [30]
2. NOOBS zip file downloaded from the official Raspberry site with the option "offline and network install".
3. All the files from the zip file where extracted files onto the SD card.

Currently the following OSes are included in NOOBS:

a. Raspbian

b. Pidora
c. LibreELEC
d. OSMC
e. RISC OS
f. Arch Linux
g. Windows 10 IoT Core

4. The first boot always takes a while until SD format and setup is completed.
5. From the following menu (Figure 24), that was loaded right after the initial boot, Raspbian OS was chosen.

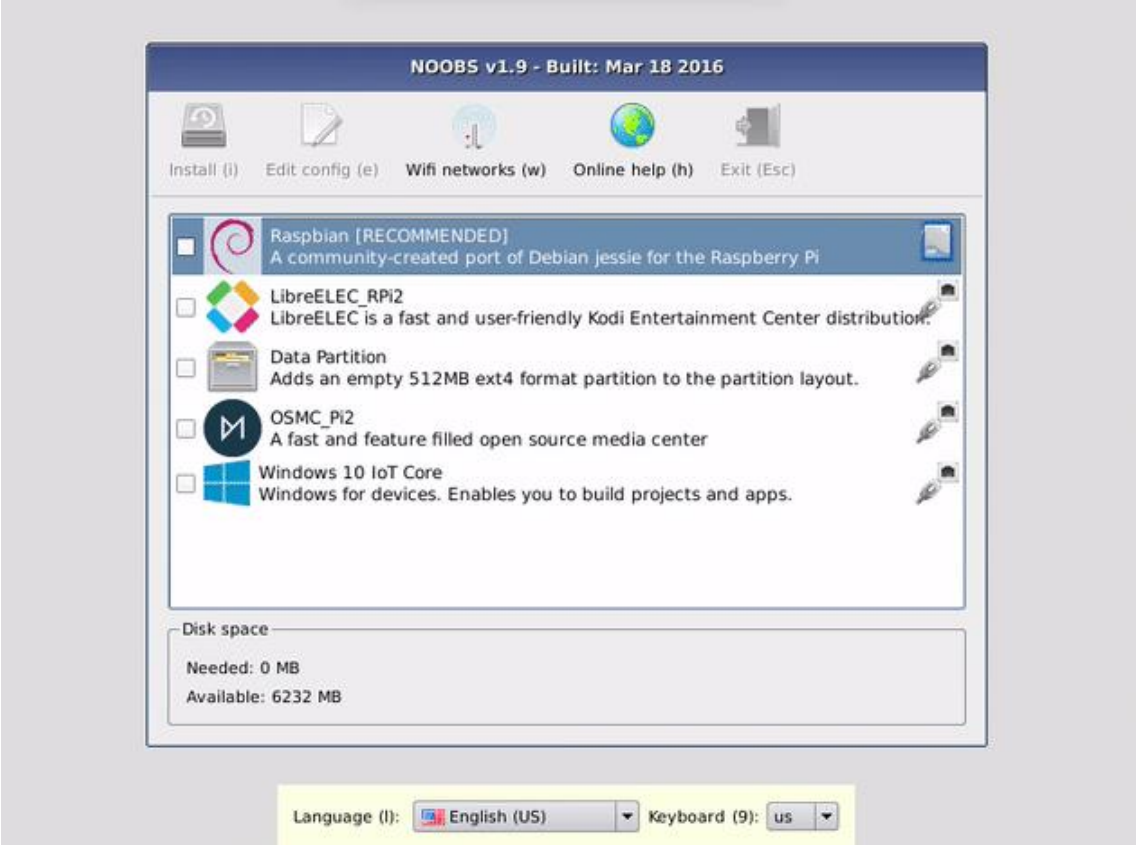


Figure 24: Noobs Installation on Raspberry Pi [31]

6. Raspbian installation takes up to 20 minutes during which additional settings were handled like keyboard language etc.

7. When everything was set-up the Raspbian Desktop finally appeared (Figure 25)

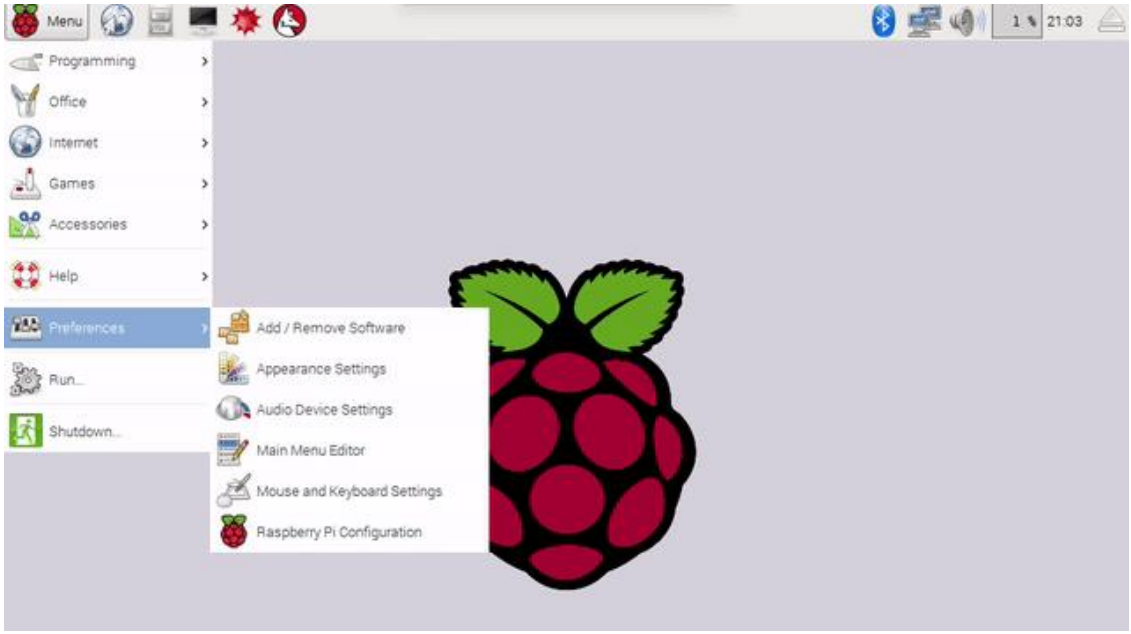


Figure 25: Desktop Screen – Raspberry Pi [31]

Now that the OS was installed in the Raspberry Pi, the next steps were the setup of the Access Point and the setup of the Web Server. To set up the Raspberry Pi as a wireless Access Point the following steps were needed.

Access Point Setup

Nowadays all routers provide wireless access, so maybe an extra Access Point seems redundant. Though, in this case the Access Point is being used as an extra security measure. Precisely, any user that is not connected to the Access Point produced by the Raspberry Pi will not be able to interact afterwards with the Web Server and consequently with the devices attached to the Raspberry Pi.

As a result, users that do not have access to the credentials of the Access Point will not be able to interact at all with the Web Server. These credentials are stored inside the Raspberry Pi and, also, in the code of the Mobile Application. Countermeasures developed to handle the risks emerged, will be analyzed in subsequent chapter.

Also, the specificity of this setup is that the Raspberry is not actually a router but a bridge. DHCP is thus delegated to the main ADSL router and all devices connected to the AP will appear on the same network than other devices.

To set-up the Access Point the following steps were necessary:

1. Download and installation of the software that will act as the “host AP” meaning host access point. To do so, the following commands were written at the Terminal of the Raspberry Pi:
    a. sudo apt-get update
    b. sudo apt-get install -y bridge-utils hostap
2. To create a bridge, ip_forward was enabled in the kernel, for that, it was necessary to edit the file “/etc/sysctl.conf ”, removing the comment (#) from the following line :
    a. net.ipv4.ip_forward=1

3. Then the network was configured to create a bridge with eth0 [31]. Actually, the following lines were added in the file “/etc/network/interfaces”, in order to bridge the “incoming” ethernet with the “outcoming” wireless Internet connection:
    a. auto lo br0
    b. iface lo inet loopback
    c. # Disable eth0 / wlan0 config, handled by bridge
    d. auto eth0
    e. iface eth0 inet manual
    f. allow-hotplug wlan0
    g. iface wlan0 inet manual
    h. # Create a bridge with static IP
    i. auto br0
    j. iface br0 inet static
    k. bridge_ports eth0
    l. address 192.168.1.7
    m. broadcast 192.168.1.255
    n. netmask 255.255.255.0
    o. gateway 192.168.1.1
    p. # Or use dhcp client on bridge
    q. #iface br0 inet dhcp

4. wlan0 is not YET configured in bridge br0. “Hostapd” package will handle this once the access point is ready [32]. So, the next step was to configure the access point. For this, edit of the file “/etc/hostapd/hostapd.conf” was necessary:
    a. # First part is about configuring the access point and is copied from reference 1
    b. interface=wlan0
    c. driver=nl80211
    d. hw_mode=g
    e. channel=6
    f. ieee80211n=1
    g. wmm_enabled=1
    h. ht_capab=[HT40][SHORT-GI-20][DSSS_CCK-40]
    i. macaddr_acl=0
    j. auth_algs=1
    k. ignore_broadcast_ssid=0
    l. wpa=2
    m. wpa_key_mgmt=WPA-PSK
    n. rsn_pairwise=CCMP
    o. # This part is about setting SSID and WPA2 password
    p. ssid=wifi_ssid #Raspberrypi1, for the current project!
    q. wpa_passphrase=wifi_password
    r. # This line ask hostapd to add wlan0 to the bridge br0
    s. bridge=br0
5. The last step needed was to allow the hostapd to use the new configuration. For that, at the end of the file “/etc/default/hostapd” and the following lone was added:
        i. DAEMON_CONF="/etc/hostapd/hostapd.conf

! To save any of the open files pressing “Control” and “X” was necessary followed by “yes” and “Enter”.

Apache Server Setup

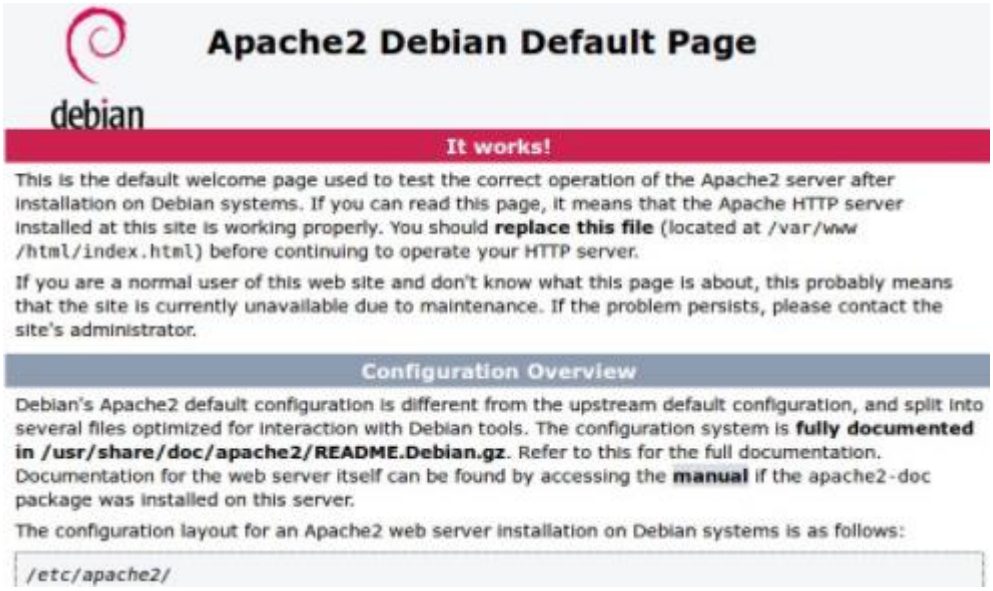
Apache is a well-known web server application that can be installed on a Raspberry Pi. It can server both HTML files over HTTP and even dynamic web pages using PHP. In the current project php will be used as the scripting language in order to:

1. Manage communication with the Android Application (get and post http requests)
2. Store data in a Database to simulate a role-based management system (inline sql commands)
3. Execute terminal commands to check whether a user is logged in the host access point (inline terminal commands)
4. Execute terminal commands to handle all the connected devices (inline terminal commands)

The code of the main php file is provided in the Appendices while some worth-mentioned parts are presented inside this chapter when needed [33].

To set-up the Apache Web Server the following steps were necessary:

1. Installation of the "apache 2" package via the terminal
    a. sudo apt-get install apache2 -y
2. Automatically after the installation, Apache placed a test HTML file in the web folder. This default web page was then browsed at "http://RASPBERRY_PI_IP", in our case "http://192.168.1.7" from another computer on the local network, to prove the configuration up to this point (Figure 26).
3. The default web page is just a HTML file on the filesystem. It is located at /var/www/html/index.html.
4. In order to edit this file, it was necessary to change the ownership from the root user to the pi user typing the command:



    a. sudo chown pi: index.html

Figure 26: Apache 2 default page [33]

5. In order to allow Apache server to process PHP files, it was necessary to install PHP5 module as well. At the terminal to install the new package the following command was typed in:
    a. sudo apt-get install php5 libapache2-mod-php5 -y
6. It worth's to mention that in order to be able to change any file in the directory "/var/www/" the following commands were additionally needed
    a. sudo chown www-data:www-data /var/www
    b. sudo chmod 775 /var/www
    c. sudo usermod -a -G www-data pi
    d. sudo reboot
7. Replacement of the old index.html file with the new index.php file was necessary to test the php configuration:
    a. sudo rm index.html
    b. sudo nano index.php
    c. <?php echo "hello world"; ?>
8. The file was saved and browsed at "http://192.168.1.7/index.php"

From this point more than one files can be written inside the directory "/var/www/html" and can be accessed from any browser in the local network using the url: "http://192.168.1.7/FILENAME.php". The php file is not shown like an HTML file only lines starting with "echo" can be visualized in the web browser. Also, in case an error exists in the php file from a wrong command to a small typo, browsers will deny uploading the page, so it is necessary to verify that the content of the php is correct before trying to upload anything.

MySQL Installation

Another necessary package for the project is MySQL [34]. To install the necessary package the following steps were needed:

1. apt-get install php5-mysql
2. apt-get install mysql-server mysql-client
    a. During this installation step, the root password should be configured.
    b. A good practice would be to assure the password in a safe place since it is difficult to recover
3. apt-get install phpmyadmin

a. Apache2 should be chosen as a Web Server when asked
b. During the Installation the MySQL root password set before was asked

4. A few lines needed to be changed at the file: "/etc/apache2/apache2.conf"
    a. Using the terminal of the Raspberry for editing the file is necessary since all this configuration files are not allowed to be altered by any user apart from the root. This is why the word "sudo" is needed in front of any edit command:
        i. sudo nano /etc/apache2/apache2.conf
    b. The following line was added at the end of the file, and then it was saved:
        i. Include /etc/phpmyadmin/apache.conf
    c. Restart of the server was necessary, and for this reason the following command was needed:
        i. /etc/init.d/apache2 restart
5. MYSQL offers to Raspberry pi users the option to handle databases form an online platform "PhPMyadmin". To access the phpMyAdmin page, it is necessary to know the IP of the Raspberry pi in order to type at a browser the following url: "http://192.168.1.7/phpmyadmin"
6. The above-mentioned webpage is a quite user friendly GUI where the administrator using the "root" credentials for the username and the "MySQL root password" as a password, is able to edit all databases.

In order to allow the php to interact with the databases available it is necessary to create a second user that will have specific access rights on the databases which php wishes to access. To make this possible a few more steps were needed [35]:

7. Creation of an additional user apart from the root, typing on the terminal
    a. mysql>CREATE USER 'USERNAME'@'localhost' IDENTIFIED BY 'begoode';
8. Granting to "USERNAME" all privileges
    a. mysql>GRANT ALL PRIVILEGES ON *.* TO 'USERNAME'@'localhost' -> WITH GRANT OPTION;
    b. The terminal will reply with "Query OK, 0 rows affected"

At this point, all created tables can be shown using the command:

c. mysql>show tables;

9. In order to connect from a php file stored in "var/www/html" directory to one of the created databases it is necessary to include in the beginning of the php file the credentials of the new database user created before:
    a. <?php
    b. //Filename: Index.php
    c. interface Index
    d. {
    e. const HOST ="localhost";
    f. const UNAME ="USERNAME";
    g. const PW ="yourPassword";
    h. const DBNAME = "DATABASE NAME";
    i. public static function doConnect();
    j. }
    k. ?>
10. If everything is set up correctly up to this point then the url: "http://localhost/index.php" will return "Successful connection to MySQL"
11. Apart from connection to the desired database, all the typical SQL commands should be included inside the php file in order to be able to write or retrieve information from it. A more detailed description of the commands used will be given at the end of this sub-section.

Mail Server

Completing the previous set-ups, the Raspberry Pi server was up and running serving the HTTP requests [36], when the user asking was already logged in the AP and had the necessary access rights. The step to be made, was the creation of a mechanism that would notify the administrator that a new user wishes to acquire access to the system.

For this reason, a mail server was needed in order to send an email to a specific destination (the administrator) and notify him accordingly. Instead of setting up a new mail server, SMTP was used, an easy solution to send emails from the command line and consequently immediately from the php.

Unfortunately, it is required to hard code the password of a google email account on the Raspberry pi so a new email account was created for this purpose. Also, as it will be mentioned at the security countermeasures that SD card will be encrypted so possible

intruders will not have easy access on any file. Nevertheless, the account used here, will be exclusively used for this purpose only, reducing the security risks.

To set the SMTP the following step were taken:

1. At the Raspberry terminal the next lines were used:
    a. sudo apt-get install ssmtp
    b. sudo apt-get install mailutils
2. The ssmtp file was configured as shown below
    a. sudo nano /etc/ssmtp/ssmtp.conf
    b. root=postmaster
    c. mailhub=smtp.gmail.com:587
    d. hostname=Raspberrypi
    e. AuthUser=AGmailUserName@gmail.com
    f. AuthPass=TheGmailPassword
    g. FromLineOverride=YES
    h. UseSTARTTLS=YES
3. To send an email only the next command was needed:
    a. echo "Hello world email body" | mail -s "Test Subject" recipient-name@domain.com

That is the last requisite package for the current project. Having the full set-up of the Raspberry Pi completed the last step is the copy of the image to two other SD-card, one will serve as a back-up and the second for the second Raspberry pi used in the context of the dissertation.

5.4.2 Android Studio Setup

For the development of the mobile application Android Studio was used. The minimum SDK version supported is fifteen (15) while the current target version is twenty-six (26). The mobile application was developed to handle remotely a group of sensors compatible with Raspberry Pi, as already mentioned. Its main task is the wireless communication with the web server in order to read information from the sensors and change the state of devices attached to the Raspberry Pi. To accomplish these tasks, four (4) main activities were developed:

1. Main activity
2. Login activity

3. Settings activity and the
4. Raspberry handler activity

For each of the activities a Java file was created to support the functions and the actions to be taken and also an xml file to produce the graphic environment. All these files can be found in the Appendices chapter as well [A, B]. Also, other xml files were produced as well, to save code names for features like colors, background styling or even labels of xml features used inside the application.

Apart from the basic libraries necessary to produce the graphical environments other libraries where need to:

- Save the value of variables that were used across different activities
- Produce and handle HTTP requests
- Handle the wireless connection of the smart phone and
- Produce animation to occupy the users while an action is executed in the background.

For each one of the aforementioned activities a Java and a xml files were created. To handle the communication with the server, the dependencies of the application need to be updated to include the OK HTTP library. This library allows the application to send get and post http requests to the server. Since OK HTTP library it is not included in the basic Android package the "build.gradle" file need to be updated in order to include the aforementioned library. As a result, the following dependency was added at the end of grandle file:

- "compile 'com.squareup.okhttp3:okhttp:3.8.1'"

The whole build gradle file can be found in the Appendices [C].

Other needed libraries like the Shared Preferences Editor or the Graphics that were needed for the enhancement of the graphical environment and the memory management were just imported at the beginning of any Java file needed.

Apart for the setup needed for the development of the application, extra steps were needed to assure the debugging of the application.

The Android Studio offers two ways of debugging. The first is the live emulator which emulates a real device visualizing the produced code or the second ways which allows debugging on real devices connected via a USB at the PC running the Android Studio. For this project the use of the emulator was not possible since the emulator is always

offline and therefore cannot interact with any web server. As a result, a real mobile phone was used instead.

In order to use a smart phone for debugging, it is necessary to turn on the developer mode on the device. Each device needs different preparation in order to allow USB debugging, in most of the cases though the following procedure explained is enough (Figure 27 - Figure 32):

1. Open the “Settings” application to bring the list of settings. At the end there is the option “About device”

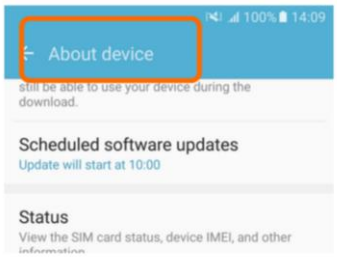


Figure 27: Enable USB Debugging (Step 1) [38]

2. On the About Device page, tap on the Software Info option.

Figure 28: Enable USB Debugging (Step 2) [38]

3. At the Software Info page, tap on the Build Number for seven (7) times, or more. Build Number is non-clickable, so nothing is noticeable on the first taps.

4. Then on the following screen, multiple clicking on the “Build Number” will unlock the developer mode.

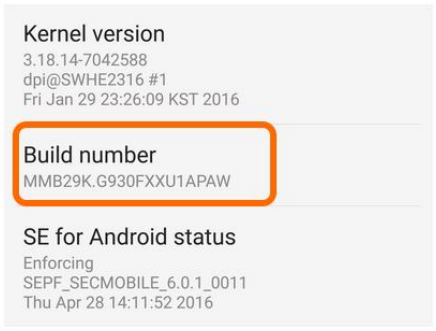


Figure 29: Enable USB Debugging (Step 4) [38]

5. After a few taps the following countdown message will appear.

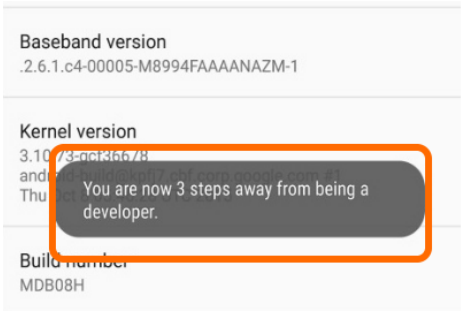


Figure 30: Enable USB Debugging (Step 5) [38]

6. When taps are enough the following message will be displayed

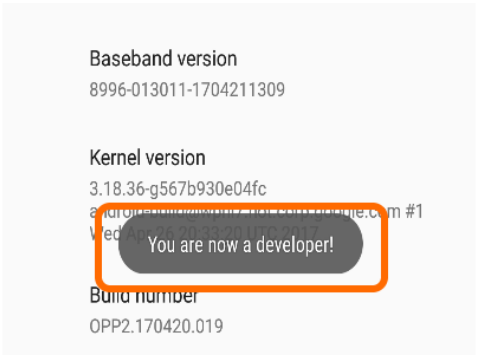


Figure 31: Enable USB Debugging (Step 6) [38]

7. Back to initial setting tab “Developer Option” will be available
8. Inside this tab, “USB Debugging” should be enabled.

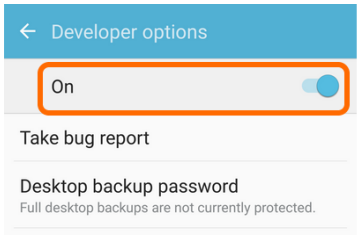


Figure 32: Enable USB Debugging (Step 8) [38]

When this setup is completed the phone used will become visible in the list of the debuggers inside the Android Studio, as shown at Figure 33.

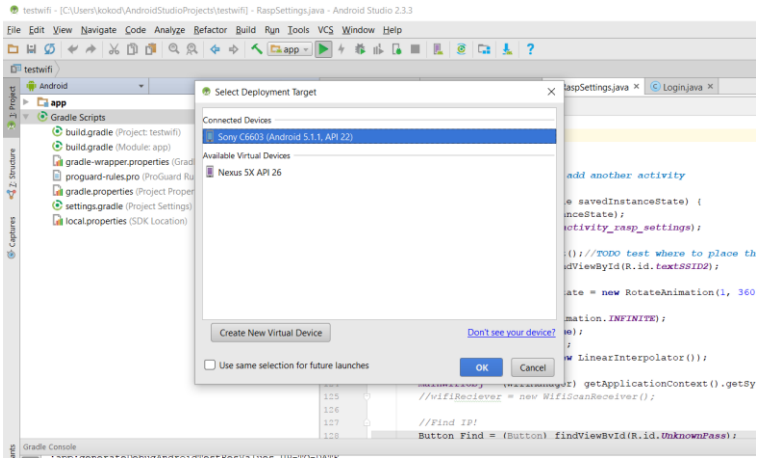


Figure 33: Android Studio Environment

The developed project alters the wi-fi condition of the smart phone used to connect it to the Access Point, whether it was or was not connected earlier to this network. As a consequence, specific permissions need to be granted to allow the application to handle the internet connection of the smart phone. To provide extra permissions in an application build in Android Studio, it is necessary to alter the "AndroidManifest.xml" file in the manifests folder. For this project the following permission were added:

- INTERNET
- ACCESS_WIFI_STATE
- ACCESS_NETWORK_STATE
- CHANGE_WIFI_STATE
- CHANGE_NETWORK_STATE

With the aforementioned alterations all the settings for the development and the debugging of the front-end part of the project are completed.

5.5 Security and Privacy Countermeasures

In order to develop a secure application, it is necessary to assure the developed project from many different aspects. For example, the physical insecurity of the Raspberry Pi and any other device used should be considered. The information stored in the Raspberry Pi or in the application itself. The privacy of the information measured by the sensors and further processed by the mobile application. The safety of the credentials used to login to the access point or used to enroll and login to the database where the access rights are managed. The identity management of the requests made to the server. All these issues are important and need to be taken under great consideration. A gap in any of them can put the whole project at risk, therefore it is necessary to secure the project from all the mentioned risks. In Table 1 the security and privacy Countermeasures are briefly presented and then analyzed further in the subsequent paragraphs.

Table 1: Security and Privacy Countermeasures

	Security	**Privacy**
Raspberry Pi serves as a Security barrier for any sensor used (wired communication)	✓	
No information regarding the **user habits or actions** is stored in the Sensors or in the Raspberry Pi		✓
Immediate **access to the Raspberry Pi files** is not possible. Raspberry Pi is not connected to any device that allows human interaction and SSH feature is disabled.	✓	
Communication between Raspberry Pi and the Android Application is handled using **HTTP POST** requests over HTTTP GET and the data exchanged are **inconsistent**.	✓	
Access Point limits the users that can interact with the system (Each Raspberry creates its own AP)	✓	
An **access Role Model** is developed to handle the rights of the users (Database build in the main Raspberry Pi). Access to the Database is password protected.	✓	
AP Password saved in the Raspberry Pi is **encrypted** (Caesar cipher way)	✓	

The most difficult part of an IoT project is the security of the sensors used. As already mentioned few chapters earlier, trying to keep the cost of such devices low makes difficult to upgrade them using encryption algorithms or other sophisticated security hardware and software modules. To overcome this obstacle, instead of using ready to go solutions on the market, sensors that can be embedded in a PCB are used in the project. Thereby, these sensors are connected to the Raspberry Pi using wires and not wirelessly, risking the

whole project. As a result, the risk of the communication between the sensors and the processing point of information is eliminated.

The aforementioned solution does not secure the project from physical insecurities like theft etc. For this risk, a future thought is to develop protective anti-theft packages to add an extra level of security to the physical risks. For the moment being, the physical insecurity is hidden under the umbrella of the hosting environment (the house that the project will be used). It is important to mention that the sensors do not have any storing capabilities so even in the case of theft no information could be extracted from them and as a result no additional risk remains for the application or its users apart from the physical loss.

Apart from the sensors, the same risks apply for the Raspberry Pi as well. In this case, things are more complicated since a group of important information are stored in the Raspberry Pi. The Raspberry Pi is not like a laptop, meaning that anyone wishing to access the files should connect it to external devices: a monitor, a keyboard and a mouse. In this project the Raspberry Pi is not connected to any device that allows human interaction and the feature for remote usage via SSH is disabled. That means that it should be transferred to another location where it could be connected to the devices needed for it to be manageable. Even if an intruder manages to work with the Raspberry Pi, no much information will be available to him since the project does not store valuable information, it basically acts as a real-time server to transfer information when needed. The most valuable things stored in the Raspberry Pi are:

- the password of the Access Point which is meaningless out of the range of the LAN network,
- and the credentials of the email account used to notify the administrator.

Both the files that store this information are password protected, adding an extra level of security to the project. Finally, the email used for this purpose is uniquely created for the developed project and it is not related to any other account or activity. It is also worth to mention that e-mail notification is an additional feature, in case the administrator is willing to check upon the enrolled users often enough, the e-mail notification system is not necessary. Example given, in case the project is intended for home usage, anyone in the house willing to access the server could notify the person in the house who plays the administrator role to grant him access. So, for this feature it is upon the user to decide to which extend he wishes to secure his system.

The access point password is an information that is saved in two places, not only in the Raspberry but in the mobile application as well. The executable code of the mobile application is not readable. An intruder should use specific tools and techniques to retrieve the code behind an executable file. Though, since those tools exists, it was necessary to add an extra level of protection. Thereby, the password is not saved as a complete readable word, it is split up in two parts and each one is shifted different times, using different functions, making it a little more difficult for the intruder to make out the password. In case the intruder manages to follow the procedure and possesses the tools needed to break the Java code, he will finally manage to acquire the password. But, as already mentioned, the password is not of great importance out of the signal range of the AP. No other valuable information is stored inside the application, all measurements from the sensors are viewed instantaneously, and the users' actions are not saved.

The previous analysis makes quite clear that the end points of the project are secured to the maximum extent possible, but still not all aspects are covered. The most difficult part was the security of the communication between the mobile application and the server. To assure the security of the communication, the following countermeasures were taken:

- Project-specific Access Point
- A Role Access Model
- HTTP POST request with meaningless context

The Access Point offered by the Raspberry Pi provides a bridge between the ethernet port of the Raspberry Pi and the wi-fi. Requests to the server coming from devices that have never been connected to the network provided by the Raspberry Pi are not allowed. The server checks to which network the requester is connected and then reads the context of the request made. Therefore, any intruder that does not acquire the necessary password will not manage to use the sensors or the devices attached.

Apart from the network verification, the server further verifies the access rights of the requester. Each user is enrolled in the system using a valid e-mail address and a password to login in the future. At the initial enrollment, the user has restricted access. To be more precise, the user may only interact with one LED attached to the Raspberry Pi as a proof of connectivity. When the administrator grants him full access, changing the necessary field in the database used for this purpose, he makes the user able to control everything connected to the Raspberry Pi. To avoid creating sessions and storing extra information for the users, each time the application sends a request for a device it includes inside the

POST request an id number which reflects the id of the user credentials inside the database mentioned earlier. In case the credentials and the access rights of the requester are verified, the server sends back the state of the device or any other information needed.

Finally, instead of using simple GET requests, the information send is hidden inside POST requests. POST unlike GET requests, security-wise, are never cached, do not remain in the browser history or in web server logs and cannot be bookmarked. While technically-wise, have no restrictions on data length or on the characters that are allowed. The POST request is a safer option but to add an extra block of security, the information that is transferred is not concise, meaning that no full sentences or meaningful words are used. The whole procedure is handled with numbers and letters that represent devices and their state, making the information exchanged meaningless to a possible intruder. The only critical information exchanged is the email, the password and the id which, as explained. The user should be responsible enough to use a password that it not related to e-mail he uses, a password exclusively to be used in the IoT solution.

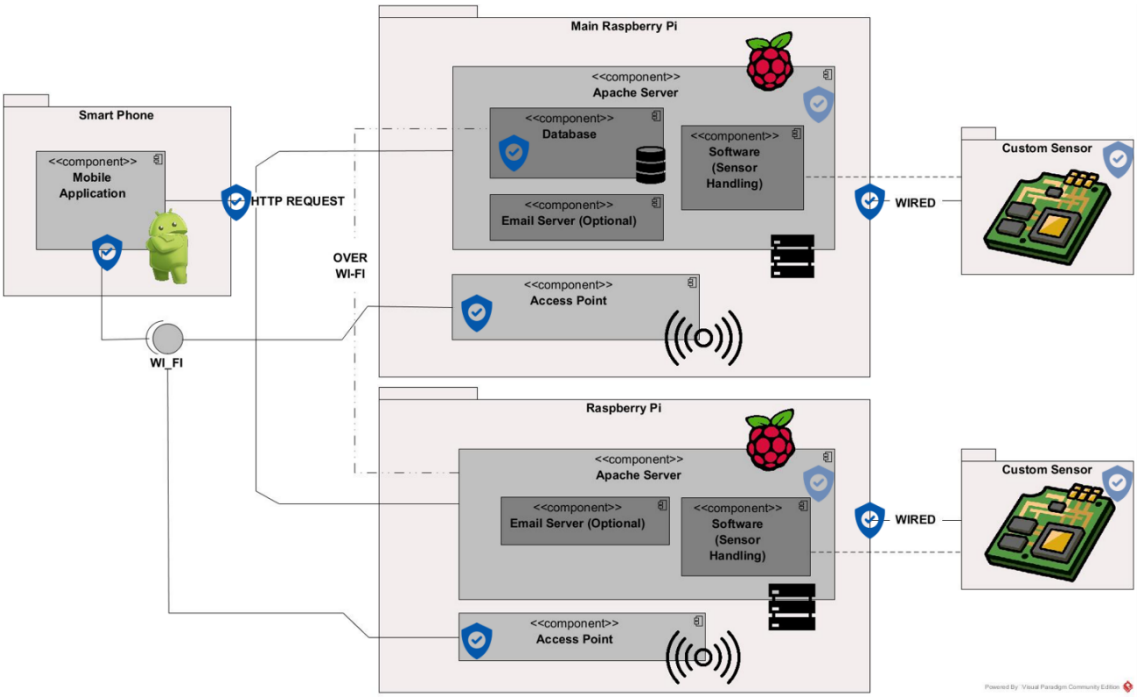


Figure 34: Security and Privacy Countermeasures on the Component Diagram

The mentioned countermeasures are a combination of techniques and ideas used to assure to the maximum extent possible the security and privacy of the whole project. The effort made can be visualized in the Component Diagram presented in Figure 34, where the dark blue icons reflect the security enhancements, and the light blue refer to privacy. It is true

that in case the project will be upgraded and used outside the range of a local network, extra countermeasures are going to be needed to re-assure the security levels.

In the last chapter, conclusions, the goals achieved, and the future thoughts will be presented to summarize and evaluate the developed solution.

6 Conclusions

In the above chapters all the aspects of the designed IoT solution were presented and also the background research on IoT architecture, technologies security and privacy. The solution proposed was built from scratch, including both the software components and, also, parts of hardware that were used. The boards used as sensors were designed and built by hand, the two Raspberry Pis were set-up using the necessary libraries while two different developed servers handle the devices connected to each one. Finally, the mobile application, "RaspIoT", was developed to handle all the devices attached to all different pins in each Raspberry Pi.

The security and privacy enhancements secured the IoT system that is used inside a local network. Extra work is needed to be able to use safely the system from anywhere. The server of the Raspberry Pi could be accessible from anywhere using the Port Forwarding technique, but extra security measures need to be taken across Denial of Service attacks or the Person in the middle attack. The use in the Local network reduces those risks since it restricts any attacker that resides away from the system's location.

Another thought is the introduction of more sensors of different kind to make the solution compatible with more real-life activities like temperature monitoring, presence of action monitoring, locking of doors, automated door opening, TV handling etc. For the time being, adding up a few sensors to the system architecture could make the solution directly suitable to be used as a smart room or as a smart house application or even inside a museum e.g. as a way of interaction between the visitors and the exhibits.

The Raspberry Pi gives us the opportunity to introduce into the system an endless group of sensors and devices that may be connected to it using wires making it easy to introduce more functionality to the designed IoT solution.

A further future development will be the design of a Web Interface that will let the final user alter the connected devices on the Raspberry Pis on their wish. And to allow the easy integration of further devices and packages for those and for the Raspberry Pis should be produced as well to make the designed IoT solution a consumable product even for people with no developing and/or electronic capabilities.

To conclude, in the framework of this thesis, the solution designed proposes the use of a Raspberry Pi as a countermeasure for the security of the sensors used, allowing at the same time the developed mobile application to handle them remotely in a secure way, inside a local network.

Bibliography

[1] S. L. &. L. D. X. &. S. Zhao, "The internet of things: a survey," Springer Science+Business Media New York 2014, 26 April 2014.

[2] F. F. G.-N. B. L. F.-R. L. B. a. M. A. A.-V. Jorge E. Ibarra-Esquer, "Tracking the Evolution of the Internet of Things Concept Across Different Application Domains," *Sensors*, vol. 17, no. 6, 2017 .

[3] Dr. Jenifer Sunrise Winter, Dr. Ryota Ono, in *The Future Internet*, Springer, 2015, p. 127.

[4] David Gil,Antonio Ferrández, Higinio Mora-Mora, and Jesús Peral, "Internet of Things: A Review of Surveys Based on Context Aware Intelligent Services," *Sensor*, vol. 16, no. 7, 2016.

[5] Charith Perera, Arkady Zaslavsky, Peter Christen, Dimitrios Georgakopoulos, "Context-awareness and model driven engineering: Illustration by an E-commerce application scenario," in *Digital Information Management 2008. ICDIM 2008. Third International Conference on, pp. 864-869*, 2008.

[6] F. C. (UniS), "The Internet of Things – Architecture," 2013.

[7] S. Mandal, "Internet of Things (IoT) - Part 2 (Building Blocks & Architecture)," 21 08 2015. [Online]. Available: http://www.c-sharpcorner.com/UploadFile/f88748/internet-of-things-part-2/.

[8] J. Fuller, "The 4 stages of an IoT architecture," HPE, 26 05 2016. [Online]. Available: https://techbeacon.com/4-stages-iot-architecture.

[9] Pallavi Sethi and Smruti R. Sarangi, "Internet of Things: Architectures, Protocols, and Applications," *Journal of Electrical and Computer Engineering*, vol. 2017, p. 25, 2017.

[10] C. Maple, "Security and privacy in the internet of things," *Journal of Cyber Policy*, pp. 155-184, 2017.

[11] A. Meola, "Internet of Things devices, applications & examples," Business Insider, 19 12 2016. [Online]. Available: http://www.businessinsider.com/internet-of-things-devices-applications-examples-2016-8.

[12] "Smart City Applications," Postscapes, [Online]. Available: https://www.postscapes.com/internet-of-things-award/smart-city-application/.

[13] M. Rogers, "Nest Mobile Android app & iPhone app updates," Nest, 10 02 2012. [Online]. Available: https://nest.com/blog/2011/12/14/nest-mobile-android-app-iphone-app-updates/.

[14] J. Axworthy, "The best Apple HomeKit compatible products for your smart home," Wearble , 06 09 2017. [Online]. Available: https://www.wareable.com/smart-home/best-apple-homekit-devices-7868.

[15] " Voice controlled gadgets for modern smart home," DrPrem, [Online]. Available: https://drprem.com/home/voice-controlled-gadgets-for-modern-smart-home/.

[16] "Watch," Apple, [Online]. Available: https://www.apple.com/lae/watch/.

[17] "7 Internet of Things (IoT) Device Examples," Plasma, 16 07 2015. [Online]. Available: http://www.plasmacomp.com/blogs/internet-of-things-iot-device-examples.

[18] Jawbone, 2017. [Online]. Available: https://jawbone.com/up.

[19] D. B. Mesko, "Top 10 Healthcare Wearables For A Healthy Lifestyle," TMF, 21 07 2016. [Online]. Available: http://medicalfuturist.com/top-healthcare-wearables/.

[20] Luminext, 03 08 2017. [Online]. Available: https://www.luminext.eu/index.php/en/.

[21] "The Streetline Solution," Streetline, 21 07 2016. [Online]. Available: https://www.streetline.com/how-it-works/.

[22] "Simplify Your Waste," enevo, [Online]. Available: https://www.enevo.com/.

[23] B. Bourque, "ARDUINO VS. RASPBERRY PI: MORTAL ENEMIES, OR BEST FRIENDS?," Digital Trends, 08 03 2015. [Online]. Available: https://www.digitaltrends.com/computing/arduino-vs-raspberry-pi/.

[24] "12 Best Development Boards for DIY Projects," mepits, 07 01 2015. [Online]. Available: https://www.mepits.com/project/236/DIY-Projects/12-Best-Development-Boards-for-DIY-Projects.

[25] "Raspberry Pi 3," Raspberry Pi , [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/.

[26] J. MSV, "10 DIY Development Boards for IoT Prototyping," The new stack, 07 05 2016. [Online]. Available: https://thenewstack.io/10-diy-development-boards-iot-prototyping/.

[27] "The Internet of Things: An Overview," *Internet Society*, pp. 20-46, 2015.

[28] P. A. P. o. C. S. a. I. George Corser, "INTERNET OF THINGS (IOT) SECURITY," *IEEE Internet Technology Policy Community White Paper*, 2017.

[29] M. R. ABDMEZIEM, "Data Confidentiality in the Internet of Things," 2016.

[30] M. H. (. K. Andreas Pfitzmann (TU Dresden), "Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management –A Consolidated Proposal for Terminolog," Reasearch Gate, 2008.

[31] S. Borini, "Traditional MVC," in *Understanding Model-View-Controller*, GitBook, 2017.

[32] "Raspberry Official website," [Online]. Available: https://www.raspberrypi.org/downloads.

[33] "Association's Formatting Tool," [Online]. Available: https://www.sdcard.org.

[34] [Online]. Available: https://wiki.debian.org/BridgeNetworkConnections#Configuring_bridging_in_.2Fetc.2Fnetwork.2Finterfaces.

[35] [Online]. Available: http://blog.ithasu.org/2016/10/using-a-raspberry-pi-3-as-a-wifi-access-point-and-bridge/.

[36] [Online]. Available: https://www.raspberrypi.org/documentation/remote-access/web-server/apache.md.

[37] [Online]. Available: http://lastbug.net/blog/2016/01/23/install-apache-php-and-mysql-and-phpmyadmin-on-raspberry-pi/.

[38] [Online]. Available: http://www.php5dp.com/get-mysql-and-php-to-work-together-in-raspberry-pi/.

[39] [Online]. Available: http://www.raspberry-projects.com/pi/software_utilities/email/ssmtp-to-send-emails.

[40] "Guide to set-up your raspberry," [Online]. Available: https://lifehacker.com/the-always-up-to-date-guide-to-setting-up-your-raspberr-1781419054.

[41] [Online]. Available: http://www.tech-recipes.com/rx/62547/how-to-unlock-developer-mode-on-android/.

Appendix

A. Android Application Source Code

The Java code that is responsible for all actions taken in the framework of the Android Application

Main Activity (Initial Screen)

```
package kokoliou_katerina.testwifi;
import android.app.ListActivity;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.SharedPreferences;
import android.net.wifi.ScanResult;
import android.net.wifi.WifiManager;
import android.os.Bundle;
import android.preference.PreferenceManager;
import android.view.View;
import android.view.animation.Animation;
import android.view.animation.LinearInterpolator;
import android.view.animation.RotateAnimation;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;
import java.util.List;

public class MainActivity extends ListActivity {
   //WiFi Manager
   WifiManager mainWifiObj;
   WifiScanReceiver wifiReciever;

   // Create SharedPreferences and Editor to save values between different activities
   SharedPreferences sharedPref;
   String savedValueSt;

   //The list view that contains all the available wi-fis
   ListView list;
   String wifis[];

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

      //Rotate animation of an image to keep the user-eye occupied while the available wi-fi are scanned
      RotateAnimation rotate = new RotateAnimation(0, 360, Animation.RELATIVE_TO_SELF, 0.5f,
Animation.RELATIVE_TO_SELF, 0.5f);
      rotate.setDuration(3000);
      rotate.setRepeatCount(1);
```

```
        rotate.setInterpolator(new LinearInterpolator());
        ImageView rasp = (ImageView) findViewById(R.id.imageView);

        //Text View to inform the user about the action he needs to perform
        TextView MAtitle = (TextView) findViewById(R.id.textMainA);
        MAtitle.setText("Choose one of the available raspberries! ");

        //Button that lead to the set-up the communication with the rasp-pi
        Button Raspberry2 = (Button) findViewById(R.id.Raspberry2);

        //The wifi manager helps to get the list with the available wu-fi
        list = getListView();
        mainWifiObj = (WifiManager) getApplicationContext().getSystemService(Context.WIFI_SER-
VICE);
        wifiReciever = new WifiScanReceiver();

        //Function that inititates the scanning
        mainWifiObj.startScan();

        //Start the animation
        rasp.startAnimation(rotate);

        //Listening to single list item on click
        list.setOnItemClickListener(new AdapterView.OnItemClickListener() {
          public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
            // selected item
            String ssid = ((TextView) view).getText().toString();
            SaveString("SSID",ssid);
            Toast.makeText(MainActivity.this, ssid, Toast.LENGTH_SHORT).show();

            //Move the user to the set-up activity and end the active one
            startActivity(new Intent(MainActivity.this, RaspSettings.class));
            finish();
          }
        });

        Raspberry2.setOnClickListener(new View.OnClickListener() {
          @Override
          public void onClick(View v) {
            //Move the user to the handling activity  (the last chosen Raspberry-pi can be controlled)
            startActivity(new Intent(MainActivity.this, RaspberryHandler.class));
            /* if you want to finish the first activity then just call*/
            finish();
          }
        });

    }

    //Function to save string and use them between different activities
    public void SaveString(String key, String value){
        sharedPref = PreferenceManager.getDefaultSharedPreferences(getApplicationContext());
        SharedPreferences.Editor editor = sharedPref.edit();
        editor.putString(key, value);
        editor.apply();
    }

    //Function to load string from a different activitie
    public void LoadString(String key){
        sharedPref = PreferenceManager.getDefaultSharedPreferences(getApplicationContext());
        savedValueSt = sharedPref.getString(key,"");
```

```
    }

    //In case the phone is shaken, or the app restarted -> re scan is necessary
    protected void onResume() {
        registerReceiver(wifiReciever, new IntentFilter(WifiManager.SCAN_RESULTS_AVAILA-
BLE_ACTION));
        super.onResume();
    }

    class WifiScanReceiver extends BroadcastReceiver {
      // @SuppressLint("UseValueOf")
        public void onReceive(Context c, Intent intent) {
            List<ScanResult> wifiScanList = mainWifiObj.getScanResults();
            wifis = new String[wifiScanList.size()];

            for (int i = 0; i < wifiScanList.size(); i++) {
                wifis[i] = ((wifiScanList.get(i)).toString());
            }

            //Procedure to filter only the access point produced by a Raspberry
            String filtered[] = new String[wifiScanList.size()];
            int counter = 0;
            for (String eachWifi : wifis) {
                String[] temp = eachWifi.split(",");

                filtered[counter] = temp[0].substring(5).trim();
                if (filtered[counter].startsWith("Raspberry")) {
                    counter++;
                }

            }
            String filteredFinal[] = new String[counter];
            for (int i=0;i<counter;i++)
            {
                filteredFinal[i] =filtered[i];

            }

            list.setAdapter(new ArrayAdapter<String>(getApplicationContext(), R.layout.listitem, R.id.label,
filteredFinal));

            //Extra styling for the list view
            list.setFooterDividersEnabled(true);
            list.setDividerHeight(5);

        }
    }//Broadcast Receiver

}//Main Activity
```

Raspberry Settings (Second screen)

```
package kokoliou_katerina.testwifi;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.net.wifi.WifiConfiguration;
```

```
import android.net.wifi.WifiInfo;
import android.net.wifi.WifiManager;
import android.os.AsyncTask;
import android.preference.PreferenceManager;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;
import android.widget.TextView;
import android.widget.ImageView;
import java.io.IOException;
import java.net.URL;
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.Response;

public class RaspSettings extends Activity {

    //HTTP Client
    OkHttpClient client;

    //WiFi Manager
    WifiConfiguration wifiConfig;
    WifiManager mainWifiObj;

    // Create SharedPreferences and Editor
    SharedPreferences sharedPref;
    String savedValueSt;

    TextView textSSID;
    String parameter;
    String wifiSSID;

    String test = "";
    int check = 0;
    String htres = "a";

    //Ip of any Raspberry connected.
    String ip = "";

    //Settings for 3 raspberries
    int cur_rasp = 0;
    String ip1 = "";
    String ip2 = "";
    String ip3 = "";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_rasp_settings);

        //The Ok HTTP client should
        client = new OkHttpClient();
        textSSID = (TextView) findViewById(R.id.textSSID2);

        wifiConfig = new WifiConfiguration();
        mainWifiObj = (WifiManager) getApplicationContext().getSystemService(Context.WIFI_SER-
VICE);
```

```
        // The IP of the Raspberry Pi are static now so Find actually choose the correct IP for each Raspberry AP name
        Button Find = (Button) findViewById(R.id.UnknownPass);

        //Close leads back to the previous screen
        Button Back = (Button) findViewById(R.id.Close);

        //Connect to the selected Access Point
        Button Connect = (Button) findViewById(R.id.okButton);

        //Image (decoration)
        final ImageView rasp1 = (ImageView) findViewById(R.id.imageView);

        //To check if connected
        final WifiInfo wifiInfo = mainWifiObj.getConnectionInfo();
        //Load Value
        LoadString("SSID");
        wifiSSID = savedValueSt;

        //To check if IP is already set
        if (wifiSSID.equals("Raspberry1")) {
            ip = "192.168.1.7";
            SaveString("savedip",ip);
            ip1 = ip;
            SaveString("savedip1",ip1);
            cur_rasp = 1;
        } else if (wifiSSID.equals("Raspberry2")) {
            ip = "192.168.1.8";
            SaveString("savedip",ip);
            ip2 = ip;
            SaveString("savedip2",ip2);
            cur_rasp = 2;
        } else if (wifiSSID.equals(String.format("\"Raspberry3\""))) {
            ip = "192.168.1.9";
            SaveString("savedip",ip);
            ip3 = ip;
            SaveString("savedip2",ip3);
            cur_rasp = 3;
        }

        if (wifiInfo.getSSID().equals(String.format("\"%s\"", wifiSSID))) {
            textSSID.setText(" All settings for " + wifiSSID + "are made");
            //Go to the Login screen
            startActivity(new Intent(RaspSettings.this, Login.class));
            finish();
        } else {
            textSSID.setText("Connect to " + wifiSSID);
        }

        //Function connect to the network;
        Connect.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //Verify if connected
                if (wifiInfo.getSSID().equals(String.format("\"%s\"", wifiSSID))) {
                    Toast.makeText(RaspSettings.this, " You are already connected!",
Toast.LENGTH_LONG).show();
                    //If IP is already set the Dialog will not be displayed at all
                    textSSID.setText("You are already connected to " + wifiSSID + "! Please set-up the
IP");
```

```
            } else {
                textSSID.setText(" Wait until WI-FI is set!");
                String start1;
                start1 = start() + end();
                finallyConnect(start1, wifiSSID);

                textSSID.setText("Wait until you have full WI-FI signal and then set the IP!");
            }
        }
    });

    Back.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            startActivity(new Intent(RaspSettings.this, MainActivity.class));
        }
    });

    Find.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            check = 0;
            if (cur_rasp == 1){
                ip = "192.168.1.7";
                SaveString("savedip",ip);
                ip1 = ip;
                SaveString("savedip1",ip1);
                textSSID.setText(" IP is set! ");
            }
            else if(cur_rasp == 2){
                ip = "192.168.1.8";
                SaveString("savedip",ip);
                ip2 = ip;
                SaveString("savedip2",ip2);
                textSSID.setText(" IP is set! ");
            }
            else
            {
                //nothing
            }
            startActivity(new Intent(RaspSettings.this, Login.class));
            finish();

        }
    });

}

public String start(){
    String start1;
    char[] start = {'b', 'l', 's', 'q', 'j'};
    for (int j = 0; j < start.length; j++) {
        char temp;
        temp = (char) ((int) start[j] - 1);
        start[j] = temp;
    }
    start1 = new String(start);
    return start1;
}
```

```
    public String end(){
        String end1;
        char[] end = {'3', '9', '3', '4'};
        for (int j = 0; j < end.length; j++) {
            int temp;
            temp = (int) end[j] - 2;
            end[j] = (char) temp;
        }
        end1 = new String(end);
        return end1;
    }

    public void SaveString(String key, String value) {
        sharedPref = PreferenceManager.getDefaultSharedPreferences(getApplicationContext());
        SharedPreferences.Editor editor = sharedPref.edit();
        editor.putString(key, value);
        editor.apply();
    }

    public void LoadString(String key) {
        sharedPref = PreferenceManager.getDefaultSharedPreferences(getApplicationContext());
        savedValueSt = sharedPref.getString(key, "");
    }

    private void finallyConnect(String networkPass, String networkSSID) {

        wifiConfig.SSID = String.format("\"%s\"", networkSSID);
        wifiConfig.preSharedKey = String.format("\"%s\"", networkPass);

        //Remember id
        int netId = mainWifiObj.addNetwork(wifiConfig);
        mainWifiObj.disconnect();
        mainWifiObj.enableNetwork(netId, true);
        mainWifiObj.reconnect();

    }

}//RaspSettings
```

Login Activity (Third Screen)

```
package kokoliou_katerina.testwifi;

import android.app.Activity;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.AsyncTask;
import android.preference.PreferenceManager;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import android.widget.TextView;
import java.io.IOException;
import java.util.concurrent.ExecutionException;
import okhttp3.FormBody;
import okhttp3.MediaType;
import okhttp3.OkHttpClient;
```

```
import okhttp3.Request;
import okhttp3.RequestBody;
import okhttp3.Response;

public class Login extends Activity {

    // Create SharedPreferences and Editor
    SharedPreferences sharedPref;
    String savedValueSt2;
    OkHttpClient client;
    TextView LoginTxt;
    EditText Name ;
    EditText Password ;
    String ip;
    String url,urlbase;
    String user;
    String password;
    String action;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);

        /* Define all the buttons (use final for Buttons used in different functions) */
        Button Back = (Button) findViewById(R.id.Back);
        final Button Login = (Button) findViewById(R.id.Login);
        final Button Enroll = (Button) findViewById(R.id.Enroll);

        client = new OkHttpClient();

        LoginTxt = (TextView) findViewById(R.id.result);
        Name = (EditText) findViewById(R.id.Name);
        Password = (EditText) findViewById(R.id.Password);

        LoadString("loginmail");
        user = savedValueSt2;
        Name.setText("" + user + "");

        LoadString("loginpass");
        password = savedValueSt2;
        Password.setText("" + password + "");

        LoadString("savedip");
        ip = savedValueSt2;
        urlbase = "http://" + ip;

        LoginTxt.setText("Fill in your (new) E-mail and Password");

        /*********************************************************/
         /*  Set an onclick/onchange listener for every button  */
        /*********************************************************/

        Enroll.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                  /* Get State */
                url = urlbase + "/index.php";

                action = "enroll";
```

```
            user = Name.getText().toString() + "";
            if (!isValidEmail(user)) {
                Name.setText("");
                Toast.makeText(Login.this, " The e-mail you entered is not valid!",
Toast.LENGTH_SHORT).show();
            } else{
                sendrequest();
            }
        }
    });

    //On successful login the user moves automatically to the last screen
    Login.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            /* Get State */
            url = urlbase + "/index.php";

            action = "login";
            user = Name.getText().toString();
            if (!isValidEmail(user)) {
                Name.setText("");
                Toast.makeText(Login.this, " The e-mail you entered is not valid!",
Toast.LENGTH_SHORT).show();
            } else{
                SaveString("LoginMail",user);
                sendrequest();
            }
        }
    });

    Back.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            /* Change Raspberry */
            startActivity(new Intent(Login.this, MainActivity.class));
            finish();
        }
    });

}
public void SaveString(String key, String value){
    sharedPref = PreferenceManager.getDefaultSharedPreferences(getApplicationContext());
    SharedPreferences.Editor editor = sharedPref.edit();
    editor.putString(key, value);
    editor.apply();
}

public void LoadString(String key){
    sharedPref = PreferenceManager.getDefaultSharedPreferences(getApplicationContext());
    savedValueSt2 = sharedPref.getString(key,"");
}

public void sendrequest(){
    password = Password.getText().toString();
    PostHandler handler = new PostHandler();
    String result = null;

    try {
        result = handler.execute(url).get();
```

```
        if (result == null)
            LoginTxt.setText("Null is returned");
            //DEBUG MSG
            //LoginTxt.setText("Null is returned" + user + password);
        else {
            LoginTxt.setText(result);
            //DEBUG MSG
            //LoginTxt.setText(result + " " + user + " " + ip );

            if ((action.equals("login")) && (result.startsWith("Succ"))){
                /*If login is succesful save credentials*/
                SaveString("loginmail",user);
                SaveString("loginpass",password);
                startActivity(new Intent(Login.this, RaspberryHandler.class));
            }
        }
    } catch (InterruptedException e) {
        e.printStackTrace();
    } catch (ExecutionException e) {
        e.printStackTrace();
    }
}

//Function to verify whether the user have enter an email
public static boolean isValidEmail(CharSequence target) {
    if (target == null) {
        return false;
    } else {
        return android.util.Patterns.EMAIL_ADDRESS.matcher(target).matches();
    }
}

public class PostHandler extends AsyncTask<String, Void, String> {
    OkHttpClient client = new OkHttpClient();
    public PostHandler(){
    }

    @Override
    protected String doInBackground(String... params) {
        RequestBody formBody = new FormBody.Builder()
                .add("Name", user)
                .add("Password", password)
                .add("Action", action)
                .build();

        Request request = new Request.Builder()
                .url(params[0]).post(formBody)
                .build();

        try {
            Response response = client.newCall(request).execute();
            if (!response.isSuccessful())
                throw new IOException("Unexpected code " + response.toString());
            return response.body().string();
        } catch (Exception e) {
        }

        return null;
    }
```

```
    //Send JSON to server
    protected String post(String url, String data) throws IOException {
        MediaType JSON = MediaType.parse("application/json; charset=utf-8");
        RequestBody body = RequestBody.create(JSON, data);
        Request request = new Request.Builder()
                .url(url)
                .post(body)
                .build();
        Response response = client.newCall(request).execute();
        return response.body().string();
    }//post
  }//post handler

} //Login
```

Raspberry Handle (Last Screen)

```
package kokoliou_katerina.testwifi;

import android.app.Activity;
import android.content.Intent;
import android.content.SharedPreferences;
import android.graphics.Typeface;
import android.os.AsyncTask;
import android.preference.PreferenceManager;
import android.os.Bundle;
import android.text.SpannableString;
import android.text.SpannableStringBuilder;
import android.text.Spanned;
import android.text.style.StyleSpan;
import android.view.View;
import android.widget.Button;
import android.widget.CompoundButton;
import android.widget.ImageButton;
import android.widget.Switch;
import android.widget.Toast;
import android.widget.TextView;
import java.io.IOException;
import java.util.concurrent.ExecutionException;
import java.util.Calendar;
import okhttp3.FormBody;
import okhttp3.MediaType;
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.RequestBody;
import okhttp3.Response;
import static android.R.drawable.ic_popup_sync;

public class RaspberryHandler extends Activity {

    // Create SharedPreferences and Editor
    SharedPreferences sharedPref;
    String savedValueSt1;

    OkHttpClient client;

    String fun_name,fun_value;

    Switch led1;
    Switch led2;
    Switch led3;
```

```
Switch led4;
Switch led5;

TextView result1 ;
TextView TxtIn1 ;

String ip;
String url,urlbase,user,password;

Calendar cal;
int minutes;
int hour;
int apm;

@Override
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_Raspberry_handler);

   led1 = (Switch) findViewById(R.id.Led1);
   led2 = (Switch) findViewById(R.id.Led2);
   led3 = (Switch) findViewById(R.id.Led3);
   led4 = (Switch) findViewById(R.id.Led4);
   led5 = (Switch) findViewById(R.id.Led5);

   Button Back = (Button) findViewById(R.id.ButtonBack);
   Button BtnIn1 = (Button) findViewById(R.id.BtnIn1);

   //Use Sync Button from Android Library
   ImageButton Sync = (ImageButton) findViewById(R.id.img_sync);
   Sync.setImageResource(ic_popup_sync);

   client = new OkHttpClient();

   result1 = (TextView) findViewById(R.id.result);
   TxtIn1 = (TextView) findViewById(R.id.In1);

   LoadString("savedip");
   ip = savedValueSt1;
   urlbase = "http://" + ip;

   LoadString("loginmail");
   user = savedValueSt1;
   LoadString("loginpass");
   password = savedValueSt1;
   url = urlbase + "/index.php";
   result1.setText("Welcome" + "  " + user);

   //To sync with what other user may change
   updateUI();

   /*********************************************************/
    /*  Set an onclick/onchange listener for every button  */
   /*********************************************************/

   led1.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
      public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
         if (isChecked) {
            fun_name = "led1";
            fun_value = "1";
```

```
            sendrequest();

        } else {
            fun_name = "led1";
            fun_value = "0";
            sendrequest();
        }
    }
});

led2.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
        if (isChecked) {
            fun_name = "led2";
            fun_value = "1";
            sendrequest();
        } else {
            fun_name = "led2";
            fun_value = "0";
            sendrequest();
        }
    }
});

led3.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
        if (isChecked) {
            fun_name = "led3";
            fun_value = "1";
            sendrequest();
        } else {
            fun_name = "led3";
            fun_value = "0";
            sendrequest();
        }
    }
});

led4.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
        if (isChecked) {
            fun_name = "led4";
            fun_value = "1";
            sendrequest();
        } else {
            fun_name = "led4";
            fun_value = "0";
            sendrequest();
        }
    }
});

led5.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
        if (isChecked) {
            fun_name = "led5";
            fun_value = "1";
            sendrequest();
        } else {
            fun_name = "led5";
```

```
                fun_value = "0";
                sendrequest();
            }
        }
    });

    BtnIn1.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            fun_name = "in1";
            fun_value = "0";
            sendrequest();
        }
    });

    Back.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            /* Back to main activity to change Raspberry */
            startActivity(new Intent(RaspberryHandler.this, MainActivity.class));
            finish();
        }
    });

    Sync.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            /* Update UI */
            updateUI();
        }
    });

}

public void LoadString(String key){
    sharedPref = PreferenceManager.getDefaultSharedPreferences(getApplicationContext());
    savedValueSt1 = sharedPref.getString(key,"");
}

private void updateUI(){
    //GET CURRENT STATE
    result1.setText("Hello" + "  " + ip);
    fun_name = "all";
    fun_value = "0";
    sendrequest();
}

public void sendrequest(){
    PostHandler handler = new PostHandler();
    String result = null;

    try {
        result = handler.execute(url).get();
        if (result == null)
            Toast.makeText(RaspberryHandler.this,"Null is returned" + fun_name + fun_value,
Toast.LENGTH_SHORT).show();
        else {
            //DEBUG MSG
            //Toast.makeText(RaspberryHandler.this, result, Toast.LENGTH_SHORT).show();
            //NEW set all answers
```

```
            result1.setText("Hello" + "   " + user);
            if(result.startsWith("state")){
               String intext;
               intext = "in1 low";

               if (result.charAt(5)=='A') {
                  result1.setText("Hello" + "   " + user + "\n" + "Only Led1 can be controlled!");
                  intext = "Limited Access";
                  if ((result.charAt(7) == '1') && (!led1.isChecked()))
                     led1.setChecked(true);
                  //Only led1 allowed
               } //Limited Access
               else {
                  result1.setText("Hello" + "   " + user);
                  if (result.charAt(5) == '1')
                     intext = "in1 high";
                  else if (result.charAt(5) == 'N')
                     intext = "Not in use";
                  //By default it starts as unchecked!
                  if ((result.charAt(7) == '1') && (!led1.isChecked()))
                     led1.setChecked(true);
                  if ((result.charAt(9) == '1') && (!led2.isChecked()))
                     led2.setChecked(true);
                  if ((result.charAt(11) == '1') && (!led3.isChecked()))
                     led3.setChecked(true);
                  if ((result.charAt(13) == '1') && (!led4.isChecked()))
                     led4.setChecked(true);
                  if ((result.charAt(15) == '1') && (!led5.isChecked()))
                     led5.setChecked(true);

               } //Full Access
               cal = Calendar.getInstance();
               minutes = cal.get(Calendar.MINUTE);
               hour = cal.get(Calendar.HOUR);
               apm = cal.get(Calendar.AM_PM);
               String a_pm ="";
               if (apm == '1'){
                  a_pm = "PM";
               }
               else
                  a_pm= "AM";

               SpannableString s0 = new SpannableString(intext + "\n");
               SpannableString s1 = new SpannableString("Updated @ " + hour + ":" + minutes+ " " +
a_pm + "!");
               int flag = Spanned.SPAN_EXCLUSIVE_EXCLUSIVE;
               s0.setSpan(new StyleSpan(Typeface.NORMAL), 0, s0.length(), flag);
               s1.setSpan(new StyleSpan(Typeface.ITALIC), 0, s1.length(), flag);

               SpannableStringBuilder builder = new SpannableStringBuilder();
               builder.append(s0);
               builder.append(s1);
               TxtIn1.setText(builder);

            }
            else if (result.startsWith("led")){
               Toast.makeText(RaspberryHandler.this, result, Toast.LENGTH_SHORT).show();
            }
            else  if (result.startsWith("in1")) {
                  cal = Calendar.getInstance();
```

```
                minutes = cal.get(Calendar.MINUTE);
                hour = cal.get(Calendar.HOUR);
                apm = cal.get(Calendar.AM_PM);
                String a_pm = "";
                if (apm == 1){
                   a_pm = "PM";
                }
                else
                   a_pm = "AM";

                SpannableString s0 = new SpannableString(result + "\n");
                SpannableString s1 = new SpannableString("Updated @ " + hour + ":" + minutes + " "
+ a_pm + "!");
                int flag = Spanned.SPAN_EXCLUSIVE_EXCLUSIVE;
                s0.setSpan(new StyleSpan(Typeface.NORMAL), 0, s0.length(), flag);
                s1.setSpan(new StyleSpan(Typeface.ITALIC), 0, s1.length(), flag);

                SpannableStringBuilder builder = new SpannableStringBuilder();
                builder.append(s0);
                builder.append(s1);
                TxtIn1.setText(builder);

             }
          else if(result.startsWith("NOT")) {
                Toast.makeText(RaspberryHandler.this, "Not in use!",
Toast.LENGTH_SHORT).show();
                switch (fun_name) {
                   case ("led1"):
                      led1.setChecked(false);
                      break;
                   case ("led2"):
                      led2.setChecked(false);
                      break;
                   case ("led3"):
                      led3.setChecked(false);
                      break;
                   case ("led4"):
                      led4.setChecked(false);
                      break;
                   case ("led5"):
                      led5.setChecked(false);
                      break;
                }
             }
          else if (result.startsWith("Access Not")){
             result1.setText("Contact your Administrator! \n" + result);
             switch (fun_name) {
                case ("led1"):
                   led1.setChecked(false);
                   break;
                case ("led2"):
                   led2.setChecked(false);
                   break;
                case ("led3"):
                   led3.setChecked(false);
                   break;
                case ("led4"):
                   led4.setChecked(false);
                   break;
                case ("led5"):
```

```
                led5.setChecked(false);
                break;
            case ("in1"):
                TxtIn1.setText("Access Not Granted!");
                break;
        }

      }
      else {
          result1.setText("Server busy,try again!" + result);
          //If this messages re-appears --> Comment for the developper
          //Check if gpio pin is configured as Input or Output
        }
      }

  } catch (InterruptedException e) {
    e.printStackTrace();
  } catch (ExecutionException e) {
    e.printStackTrace();
  }
}

public class PostHandler extends AsyncTask<String, Void, String> {
  OkHttpClient client = new OkHttpClient();

  public PostHandler(){
  }

  @Override
  protected String doInBackground(String... params) {
    RequestBody formBody = new FormBody.Builder()
        .add("Name", user)
        .add("Password", password)
        .add(fun_name, fun_value)
        .build();

    Request request = new Request.Builder()
        .url(params[0]).post(formBody)
        .build();

    try {
      Response response = client.newCall(request).execute();
      if (!response.isSuccessful())
        throw new IOException("Unexpected code " + response.toString());
      return response.body().string();
    } catch (Exception e) {
    }

    return null;
  }

  //Send request to server
  protected String post(String url, String data) throws IOException {
    MediaType JSON = MediaType.parse("application/json; charset=utf-8");
    RequestBody body = RequestBody.create(JSON, data);
    Request request = new Request.Builder()
        .url(url)
        .post(body)
        .build();
    Response response = client.newCall(request).execute();
```

```
            return response.body().string();
        }

    }
} //End of Activity
```

B. Application User Interface (XML files)

The XML files are responsible for the User Interface of the mobile application.

Main Activity

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/grad_back"
    tools:context="kokoliou_katerina.testwifi.MainActivity">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@null"
        android:orientation="vertical"
        android:weightSum="1"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true">

        <Space
            android:id="@+id/spaceM1"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="40dp"
            android:background="@null" />


        <TextView
            android:id="@+id/textMainA"
            android:layout_width="386dp"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:background="@null"
            android:textAlignment="center"
            android:textColor="@color/black" />

        <Space
            android:id="@+id/spaceM11"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="20dp"
            android:background="@null" />

        <ListView
            android:id="@android:id/list"
            android:layout_width="172dp"
```

```
        android:layout_height="192dp"
        android:layout_gravity="center_horizontal"
        android:layout_margin="4dip"
        android:background="@null"
        android:cacheColorHint="@color/white"
        android:drawSelectorOnTop="true"
        android:fadingEdge="none"
        android:scrollbarStyle="outsideOverlay"
        android:choiceMode="singleChoice"
        android:footerDividersEnabled="true"
        android:headerDividersEnabled="true"
        android:scrollbars="vertical"
        android:textAlignment="center"
        android:textColor="@color/white"
        android:visibility="visible" />

    <Space
        android:id="@+id/spaceM2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="20dp"
        android:background="@null" />

    <android.support.v7.widget.AppCompatImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:contentDescription="@string/app_name"
        android:layout_height="51dp"
        android:visibility="visible"
        app:srcCompat="@drawable/rasp1"
        android:layout_weight="0.26" />

    <Space
        android:id="@+id/spacemain1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="20dp"
        android:background="@null" />

    <Button
        android:id="@+id/Raspberry2"
        android:layout_width="195dp"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:background="@drawable/mybutton"
        android:ems="10"
        android:text="@string/toggle_rasp_ip"
        android:textColor="@color/black"/>

  </LinearLayout>

</RelativeLayout>
```

Raspberry Set-up

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
```

```
android:layout_width="match_parent"
tools:context="kokoliou_katerina.testwifi.RaspSettings"
android:layout_height="match_parent"
android:background="@drawable/grad_back">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="431dp"
        android:background="@null"
        android:orientation="vertical"
        android:weightSum="1"
        android:id="@+id/linearLayout"
        android:layout_centerVertical="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true">

        <TextView
            android:id="@+id/textSSID2"
            android:layout_width="285dp"
            android:layout_height="40dp"
            android:layout_gravity="center"
            android:layout_weight="0.22"
            android:background="@null"
            android:textAlignment="center"
            android:drawableBottom="@android:color/black"
            android:textColor="@color/black" />

        <android.support.v7.widget.AppCompatImageView
            android:id="@+id/imageView"
            android:layout_width="match_parent"
            android:layout_height="70dp"
            android:layout_alignParentEnd="true"
            android:layout_alignParentRight="true"
            android:layout_alignParentTop="true"
            android:layout_marginTop="24dp"
            android:layout_weight="0.19"
            android:contentDescription="@string/app_name"
            android:visibility="visible"
            app:srcCompat="@drawable/rasp1" />

        <Space
        android:id="@+id/space1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/textSSID2"
        android:layout_marginBottom="20dp"
        android:background="@null"
        android:layout_weight="0.13" />

        <Button
            android:id="@+id/okButton"
            android:layout_width="159dp"
            android:layout_height="wrap_content"
            android:layout_below="@id/space1"
            android:layout_gravity="center"
            android:background="@drawable/mybutton"
            android:text="Connect"
            android:textColor="@color/black" />

        <Space
```

```
        android:id="@+id/space2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/okButton"
        android:layout_marginBottom="20dp"
        android:background="@null"/>

    <Button
        android:id="@+id/UnknownPass"
        android:layout_width="159dp"
        android:layout_height="wrap_content"
        android:layout_below="@+id/space2"
        android:layout_gravity="center"
        android:background="@drawable/mybutton"
        android:text=" Connetion Set-up "
        android:textColor="@color/black" />

    <Space
        android:id="@+id/space3"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/okButton"
        android:layout_marginBottom="20dp"
        android:background="@null"/>

        <Button
            android:id="@+id/Close"
            android:layout_width="159dp"
            android:layout_height="wrap_content"
            android:layout_below="@+id/space3"
            android:layout_gravity="center"
            android:background="@drawable/mybutton"
            android:text=" Back "
            android:textColor="@color/black" />

</LinearLayout>

</RelativeLayout>
```

Login Activity

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/grad_back">

    <LinearLayout
        android:id="@+id/linearLayout2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:background="@null"
        android:orientation="vertical"
        android:weightSum="1">
```

```
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="horizontal">

    <android.support.v7.widget.AppCompatImageView
        android:id="@+id/appCompatImageView"
        android:layout_width="108dp"
        android:layout_height="76dp"
        android:layout_alignParentEnd="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentTop="true"
        android:layout_marginTop="10dp"
        android:contentDescription="@string/app_name"
        android:cropToPadding="false"
        android:visibility="visible"
        app:srcCompat="@drawable/rasp1" />

    <TextView
        android:id="@+id/result"
        android:layout_width="202dp"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:background="@null"
        android:drawableBottom="@android:color/black"
        android:textAlignment="center"
        android:textColor="@color/black"
        android:layout_weight="7.35" />

    </LinearLayout>

</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@null"
    android:orientation="vertical"
    android:weightSum="1"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_below="@+id/linearLayout2">


    <Space
        android:id="@+id/space"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentEnd="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentStart="true"
        android:layout_alignParentTop="true"
        android:layout_marginBottom="40dp" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@null"
```

```
    android:orientation="horizontal">

    <TextView
        android:id="@+id/textView"
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical"
        android:layout_marginLeft="30dp"
        android:layout_marginStart="30dp"
        android:background="@null"
        android:drawableBottom="@android:color/black"
        android:text="E-mail"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:textColor="@color/black"
        tools:ignore="HardcodedText,RtlHardcoded" />
    <!--"@string/led1_text"-->

    <FrameLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <EditText
            android:id="@+id/Name"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:background="@null"
            android:drawableBottom="@android:color/black"
            android:textAlignment="center"
            android:textColor="@color/black"
            android:layout_weight="7.35" />

    </FrameLayout>

</LinearLayout>

<Space
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="40dp" />

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@null"
    android:orientation="horizontal">

    <TextView
        android:id="@+id/led2_text"
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical"
        android:layout_marginLeft="30dp"
        android:layout_marginStart="30dp"
        android:background="@null"
        android:text="Password"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:textColor="@color/black"
        tools:ignore="HardcodedText" />
```

```
    <FrameLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <FrameLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content">

            <EditText
                android:id="@+id/Password"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_gravity="center"
                android:background="@null"
                android:drawableBottom="@android:color/black"
                android:textAlignment="center"
                android:textColor="@color/black"
                android:layout_weight="7.35" />

        </FrameLayout>
    </FrameLayout>
</LinearLayout>

<Space
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="20dp" />

<Space
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="20dp" />

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@null"
    android:orientation="vertical"
    android:weightSum="1">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@null"
        android:gravity="bottom"
        android:orientation="vertical">

        <Button
            android:id="@+id/Enroll"
            android:layout_width="168dp"
            android:layout_height="wrap_content"
            android:layout_gravity="bottom|center"
            android:background="@drawable/mybutton"
            android:gravity="center"
            android:text="  Enroll  "
            android:textColor="@color/black"
            tools:ignore="HardcodedText" />

    </LinearLayout>
```

```
</LinearLayout>

<Space
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:layout_marginBottom="20dp" />

<LinearLayout
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:background="@null"
   android:orientation="vertical"
   android:weightSum="1">

   <LinearLayout
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:background="@null"
      android:gravity="bottom"
      android:orientation="vertical">

      <Button
         android:id="@+id/Login"
         android:layout_width="168dp"
         android:layout_height="wrap_content"
         android:layout_gravity="bottom|center"
         android:background="@drawable/mybutton"
         android:gravity="center"
         android:text="  Login  "
         android:textColor="@color/black"
         tools:ignore="HardcodedText" />

   </LinearLayout>

</LinearLayout>

<LinearLayout
   android:layout_width="match_parent"
   android:layout_height="82dp"
   android:background="@null"
   android:orientation="vertical"
   android:weightSum="1">

   <LinearLayout
      android:layout_width="match_parent"
      android:layout_height="63dp"
      android:background="@null"
      android:gravity="bottom"
      android:orientation="vertical">

      <Button
         android:id="@+id/Back"
         android:layout_width="168dp"
         android:layout_height="wrap_content"
         android:layout_gravity="bottom|center"
         android:background="@drawable/mybutton"
         android:gravity="center"
         android:text="  Back  "
         android:textColor="@color/black"
         tools:ignore="HardcodedText" />
```

```
        </LinearLayout>

      </LinearLayout>

   </LinearLayout>

</RelativeLayout>
```

Raspberry Handler

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:background="@drawable/grad_back">

   <ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:orientation="vertical"
      android:padding="10dp"
      android:fillViewport="false"
      android:layout_alignParentBottom="true"
      android:layout_alignParentLeft="true"
      android:layout_alignParentStart="true"
      android:layout_below="@+id/linearLayout2">



   <LinearLayout
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:layout_alignParentLeft="true"
      android:layout_alignParentStart="true"
      android:background="@null"
      android:orientation="vertical"
      android:weightSum="1">

      <Space
         android:layout_width="match_parent"
         android:layout_height="wrap_content"
         android:layout_marginBottom="50dp"
         android:layout_alignParentLeft="true"
         android:layout_alignParentStart="true"
         android:id="@+id/space"
         android:layout_alignParentRight="true"
         android:layout_alignParentEnd="true"
         android:layout_alignParentTop="true" />

      <LinearLayout
         android:layout_width="match_parent"
         android:layout_height="wrap_content"
         android:background="@null"
         android:orientation="horizontal"
         android:id="@+id/linearLayout4"
         android:weightSum="1"
         android:layout_alignParentTop="true"
```

```
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true">

        <Button
            android:id="@+id/BtnIn1"
            android:layout_width="50dp"
            android:layout_height="wrap_content"
            android:layout_alignParentEnd="true"
            android:layout_alignParentRight="true"
            android:layout_alignParentTop="true"
            android:layout_marginLeft="30dp"
            android:background="@drawable/mybutton"
            android:gravity="center"
            android:text="Get State"
            android:textColor="@color/black"
            tools:ignore="HardcodedText" />

        <TextView
            android:id="@+id/In1"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_marginEnd="10dp"
            android:layout_marginLeft="10dp"
            android:layout_marginRight="110dp"
            android:layout_marginStart="110dp"
            android:background="@null"
            android:drawableBottom="@android:color/black"
            android:lines="2"
            android:textAlignment="center"
            android:textColor="@color/black" />

    </LinearLayout>

<Space
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="20dp" />

<LinearLayout
    android:orientation="horizontal"
    android:background="@null"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <TextView
        android:id="@+id/led1_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical"
        android:layout_marginLeft="30dp"
        android:layout_marginStart="30dp"
        android:background="@null"
        android:text="LED1"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:textColor="@color/black"
        tools:ignore="HardcodedText" />

    <FrameLayout
        android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content">

        <Switch
            android:id="@+id/Led1"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="end"
            android:layout_marginEnd="20dp"
            android:layout_marginRight="20dp"
            android:textColorLink="@android:color/black" />
    </FrameLayout>

</LinearLayout>

<Space
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="20dp" />

<LinearLayout
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:background="@null"
    android:layout_height="wrap_content">

    <TextView
        android:id="@+id/led2_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical"
        android:layout_marginLeft="30dp"
        android:layout_marginStart="30dp"
        android:background="@null"
        android:text="LED2"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:textColor="@color/black"
        tools:ignore="HardcodedText" />

    <FrameLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <Switch
            android:id="@+id/Led2"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="end"
            android:layout_marginEnd="20dp"
            android:layout_marginRight="20dp"
            android:textColorLink="@android:color/black" />

    </FrameLayout>

</LinearLayout>

<Space
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="20dp" />
```

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@null"
    android:orientation="horizontal">


    <TextView
        android:id="@+id/textView4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical"
        android:layout_marginLeft="30dp"
        android:layout_marginStart="30dp"
        android:background="@null"
        android:text="LED3"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:textColor="@color/black"
        tools:ignore="HardcodedText,RtlHardcoded" />
    <!--"@string/led1_text"-->

    <FrameLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <Switch
            android:id="@+id/Led3"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="end"
            android:layout_marginEnd="20dp"
            android:layout_marginStart="20dp"
            android:checked="false"
            android:textColorLink="@android:color/black" />
    </FrameLayout>

</LinearLayout>

<Space
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="20dp" />

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@null"
    android:orientation="horizontal">

    <TextView
        android:id="@+id/textView5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical"
        android:layout_marginLeft="30dp"
        android:layout_marginStart="30dp"
        android:background="@null"
        android:text="LED4"
        android:textAppearance="?android:attr/textAppearanceMedium"
```

```
        android:textColor="@color/black"
        tools:ignore="HardcodedText,RtlHardcoded" />
    <!--"@string/led1_text"-->

    <FrameLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <Switch
            android:id="@+id/Led4"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="end"
            android:layout_marginEnd="20dp"
            android:layout_marginStart="20dp"
            android:checked="false"
            android:textColorLink="@android:color/black" />
    </FrameLayout>

</LinearLayout>

<Space
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="20dp" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@null"
        android:orientation="horizontal">

        <TextView
            android:id="@+id/textView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center_vertical"
            android:layout_marginLeft="30dp"
            android:layout_marginStart="30dp"
            android:background="@null"
            android:text="LED5"
            android:textAppearance="?android:attr/textAppearanceMedium"
            android:textColor="@color/black"
            android:drawableBottom="@android:color/black"
            tools:ignore="HardcodedText,RtlHardcoded" />
        <!--"@string/led1_text"-->

        <FrameLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <Switch
                android:id="@+id/Led5"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_gravity="end"
                android:layout_marginEnd="20dp"
                android:layout_marginStart="20dp"
                android:textColorLink="@android:color/black" />
        </FrameLayout>
```

```
    </LinearLayout>

    <!--"@string/back forth"-->

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@null"
        android:orientation="vertical"
        android:weightSum="1">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="81dp"
            android:background="@null"
            android:gravity="bottom"
            android:orientation="vertical">

            <Button
                android:id="@+id/ButtonBack"
                android:layout_width="168dp"
                android:layout_height="wrap_content"
                android:layout_gravity="bottom|center"
                android:background="@drawable/mybutton"
                android:gravity="center"
                android:textColor="@color/black"
                android:text="  Change Raspberry  "
                tools:ignore="HardcodedText" />

        </LinearLayout>

        </LinearLayout>

        </LinearLayout>

    </ScrollView>

    <LinearLayout
        android:id="@+id/linearLayout2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@null"
        android:orientation="horizontal"
        android:weightSum="1"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true">

        <android.support.v7.widget.AppCompatImageView
            android:id="@+id/appCompatImageView"
            android:layout_width="108dp"
            android:layout_height="76dp"
            android:layout_alignParentEnd="true"
            android:layout_alignParentRight="true"
            android:layout_alignParentTop="true"
            android:layout_marginTop="10dp"
            android:contentDescription="@string/app_name"
            android:cropToPadding="false"
            android:visibility="visible"
```

```
        app:srcCompat="@drawable/rasp1" />

    <TextView
        android:id="@+id/result"
        android:layout_width="186dp"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_weight="0.01"
        android:background="@null"
        android:drawableBottom="@android:color/black"
        android:textAlignment="center"
        android:textColor="@color/black" />

    <ImageButton
        android:id="@+id/img_sync"
        style="@android:style/Widget.ImageButton"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_weight="0.48"
        android:background="@android:color/transparent"
        android:visibility="visible"
        app:srcCompat="@android:drawable/ic_popup_sync" />

  </LinearLayout>

</RelativeLayout>
```

Android Manifest (XML file)

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="kokoliou_katerina.testwifi">

  <uses-permission android:name="android.permission.INTERNET" />
  <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
  <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
  <uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
  <uses-permission android:name="android.permission.CHANGE_NETWORK_STATE" />
  <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
  <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />

  <application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
    <activity android:name=".MainActivity">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
    <activity android:name=".RaspberryHandler">
      <intent-filter>
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
```

```
        <activity android:name=".RaspSettings">
            <intent-filter>
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".Login">
            <intent-filter>
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

C. Module Gradle Script

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 26
    buildToolsVersion "26.0.1"
    defaultConfig {
        applicationId "kokoliou_katerina.testwifi"
        minSdkVersion 15
        targetSdkVersion 26
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
    useLibrary 'org.apache.http.legacy'
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
        exclude group: 'com.android.support', module: 'support-annotations'
    })
    compile 'com.android.support:appcompat-v7:26.+'
    compile 'com.android.support.constraint:constraint-layout:1.0.2'
    compile 'com.squareup.okhttp3:okhttp:3.8.1'
    testCompile 'junit:junit:4.12'
}
```

D. Raspberry Pi

GPIO and HTTP POST handling scripts in php.

Main Raspberry Pi

This Raspberry Pi is connected to five LEDs and a switch and is the main one that also serves the database with the access rights.

```
<?php
exec("gpio mode 0 out");
exec("gpio mode 2 out");
exec("gpio mode 3 out");
exec("gpio mode 4 out");
exec("gpio mode 5 out");
exec("gpio mode 7 in");

$response = array();
$ip = array();

exec("sudo arp -i br0 -a | cut -d' ' -f2 |tr -d '()' 2>&1", $output, $return);

if ( array_key_exists( $_SERVER['REMOTE_ADDR'] , array_flip($ip) ) !=" " ) {

if ( (isset($_POST['Name']) ) && (isset($_POST['Password']) )){

        $Name = $_POST['Name'];
        $Password = $_POST['Password'];
        $Access_r = "0";

        $username = "User";
        $password = "koko4444";
        $hostname = "192.168.1.7";
        $enroll = 0;

        $mail = $_POST['Name'];
        $pass = $_POST['Password'];

        $db = mysql_connect($hostname,$username,$password)
                or die("Unable to connect");

        $selected = mysql_select_db("Users",$db)
                or die("Could not select");

        //Check if it is a new Login or an Enrollment
        $exist = mysql_query("SELECT Name, Password, Access_r FROM Users");
        //Test Fail Login
        $fail_log ="0";
        $enroll = "0";

        while($row = mysql_fetch_array($exist)){
                if ( ($row{'Name'}) == $mail )  {//User Exists
                        if  (($row{'Password'}) == $pass){
                                $enroll = "0";
                                $rigth = $row{'Access_r'};
                                $fail_log = "3";
                        }
                        else{ //Fail Login
                                $enroll ="0";
                                $fail_log ="1";
                        }
                }
```

```
                        else {
                                $enroll = "1";//Enroll the new user
                        }
        }

        $print = "1";
    if($_POST['Action']=="login"){
            if($fail_log == "3")
                        echo "Succefull Login!";
                else if($fail_log =="1")
                        echo "Wrong Password!";
    }

        if (($enroll == "1") && ($_POST['Action']=="enroll") ) {
                $result = mysql_query("INSERT INTO Users(Name, Password, Access_r) VAL-
UES('$Name','$Password', '0')");
        }
        else{
                if($_POST['Action']=="enroll"){
                echo "User Exists! Try another email";
                }
                $print = "0";
        }

        if($result){
                echo "New User inserted succesfully";
        }
        else{//Try to enroll but failed
                if ($print !="0")
                        {echo "Insertion Failed";}
        }

        //Do not check acces rigth
        if ( isset($_POST['led1']) ) {
        if($_POST['led1'] == 1) {
        exec("gpio write 0 1");// pin 0 in wiring pi is gpio 17
                $led = exec("gpio read 0");
            if ($led == 1)
            echo "led1 on";
                else
                echo "Wrong Settings";
            } else {
            exec("gpio write 0 0");
        $led = exec("gpio read 0");//assure if command is exec.
                if ($led== 0)
                echo "led1 off";
                else
                echo "Wrong Settings";
                  }
        }
        //LED2
        if ( isset($_POST['led2']) ) {
                  if($rigth==1){
        if($_POST['led2'] == 1) {
        exec("gpio write 2 1");// pin 2 in wiring pi is gpio 22
                $led = exec("gpio read 2");
            if ($led == 1)
            echo "led2 on";
                else
                echo "Wrong Settings";
```

```
        } else {
        exec("gpio write 2 0");
$led = exec("gpio read 2");//assure if command is exec.
                if ($led== 0)
                echo "led2 off";
                else
                echo "Wrong Settings";
                                        }
                                        }//rigths
                  else
                                echo "Access Not Granted";
}

//LED3
if ( isset($_POST['led3']) ) {
if($rigth==1){
                                if($_POST['led3'] == 1) {
                                                exec("gpio write 3 1");// pin 3 is W.Pi 2?
                                                $led = exec("gpio read 3");
                                                if ($led == 1)
                                                                echo "led3 on";
                                                else
                                                                echo "Wrong Settings";
                                } else {
                                                exec("gpio write 3 0");
                                                $led = exec("gpio read 3");
                                                if ($led== 0)
                                                                echo "led3 off";
                                                else
                                                                echo "Wrong Settings";
                                }
                                }//rigths
else
                echo "Access Not Granted";
}

//LED4
if ( isset($_POST['led4']) ) {
if($rigth==1){
if($_POST['led4'] == 1) {
                                                exec("gpio write 4 1");// pin 4 is W.Pi 1?
                                                $led = exec("gpio read 4");
                                                if ($led == 1)
                                                                echo "led4 on";
                                                else
                                                                echo "Wrong Settings";
                                } else {
                                                exec("gpio write 4 0");
                                                $led = exec("gpio read 4");
                                                if ($led== 0)
                                                                echo "led4 off";
                                                else
                                                                echo "Wrong Settings";
                                }
                                }//rigths
                                else
                                                echo "Access Not Granted";

}
```

```
        //LED5
        if ( isset($_POST['led5']) ) {
                                if($rigth==1){
        if($_POST['led5'] == 1) {
                                                exec("gpio write 5 1");// pin 5 is W.Pi 1?
                                                $led = exec("gpio read 5");
                                                if ($led == 1)
                                                                echo "led5 on";
                                                else
                                                                echo "Wrong Settings";
                                } else {
                                                exec("gpio write 5 0");
                                                $led = exec("gpio read 5");//assu$
                                                if ($led== 0)
                                                                echo "led5 off";
                                                else
                                                                echo "Wrong Settings";
                                }
                                }//rigths
                                else
                                                echo "Access Not Granted";

        }

        if(isset($_POST['in1'])) {
                        if ($rigth==1){
                        $in1 = exec("gpio read 7");
                if($in1 == 1) {
                echo "in1 high ";
                } else {
                echo "in1 low";
                        }
        }//rigths
        else
                echo "Access Not Granted";

        }

        if(isset($_POST['all'])) {
        $in1 = exec("gpio read 7");
                        $led1 = exec("gpio read 0");
                        $led2 = exec("gpio read 2");
                        $led3 = exec("gpio read 3");
                        $led4 = exec("gpio read 4");
                        $led5 = exec("gpio read 5");
                        if ($rigth==1){
                        echo "state".$in1."".$led1."".$led2."".$led3."".$led4."".$led5;
                        }
                        else{
                        echo "state"."A"."".$led1."""0""""0""""0""""0";
                        }
        }

}else{ //Post req. incomplete
        echo "Required field(s) is missing";
}
}//Connected to AP
?>
```

Second Raspberry Pi

This Raspberry Pi is connected to three LEDs only and communicates with the main Raspberry to gain access over the access rights' database.

```
<?php

exec("gpio mode 0 out"); //PHYSICAL PIN 11
exec("gpio mode 2 out"); //PHYSICAL PIN 13
exec("gpio mode 3 out"); //PHYSICAL PIN 15
exec("gpio mode 4 out"); //PHYSICAL PIN 16
exec("gpio mode 5 out"); //PHYSICAL PIN 18
exec("gpio mode 7 out"); //PHYSICAL PIN 07

exec("gpio write 4 0");// ALWAYS OFF
exec("gpio write 5 0");// ALWAYS OFF
exec("gpio write 6 0");// ALWAYS OFF

$response = array();
$ip = array();

exec("sudo arp -i br0 -a | cut -d' ' -f2 |tr -d '()' 2>&1", $output, $return);

if ( array_key_exists( $_SERVER['REMOTE_ADDR'] , array_flip($ip) ) !=" " ) {



if ( (isset($_POST['Name']) ) && (isset($_POST['Password']) )){

        $Name = $_POST['Name'];
        $Password = $_POST['Password'];
        $Access_r = "0";

        $username = "User";
        $password = "koko4444";
        $hostname = "192.168.1.7";
        $enroll = 0;

        $mail = $_POST['Name'];
        $pass = $_POST['Password'];

        $db = mysql_connect($hostname,$username,$password)//new DB_CON-
NECT();//host="192.168.1.22/phpmyadmin/index.php",user="User",passwd="koko4444",db="Users");
                or die("Unable to connect");

        $selected = mysql_select_db("Users",$db)
                or die("Could not select");

        //Check if it is a new Login or an Enrollment
        $exist = mysql_query("SELECT Name, Password, Access_r FROM Users");
        //Test Fail Login
        $fail_log ="0";
        $enroll = "0";

        while($row = mysql_fetch_array($exist)){
                if ( ($row{'Name'}) == $mail )  {//User Exists
                        if  (($row{'Password'}) == $pass){
                                $enroll = "0";
```

```
                                        $rigth = $row{'Access_r'};
                                        $fail_log = "3";
                                }
                                else{ //Fail Login
                                        $enroll ="0";
                                        $fail_log ="1";
                                }
                        }
                        else {
                                $enroll = "1";//Enroll the new user
                        }
                }

                $print = "1";
        if($_POST['Action']=="login"){
                        if($fail_log == "3")
                                echo "Succefull Login!";
                        else if($fail_log =="1")
                                echo "Wrong Password!";
        }

                if (($enroll == "1") && ($_POST['Action']=="enroll") ) {
                        $result = mysql_query("INSERT INTO Users(Name, Password, Access_r) VAL-
UES('$Name','$Password', '0')");
                }
                else{
                        if($_POST['Action']=="enroll"){
                        echo "User Exists! Try another email";
                        }
                        $print = "0";
                }

                if($result){
                        echo "New User inserted succesfully";
                }
                else{//Try to enroll but failed
                        if ($print !="0")
                                {echo "Insertion Failed";}
                }

                //fail_log = 3 --> Correct user name and password
                if ( isset($_POST['led1']) ) {
                if($_POST['led1'] == 1) {
                exec("gpio write 0 1");// pin 0 in wiring pi is gpio 17
                        $led = exec("gpio read 0");
                        if ($led == 1)
                        echo "led1 on";
                                else
                                echo "Wrong Settings";
                        } else {
                        exec("gpio write 0 0");
                $led = exec("gpio read 0");//assure if command is exec.
                                if ($led== 0)
                                echo "led1 off";
                                else
                                echo "Wrong Settings";
                                  }
                }

                //LED2
```

```
if ( isset($_POST['led2']) ) {
if($_POST['led2'] == 1) {
exec("gpio write 2 1");// pin 2 in wiring pi is gpio 22
        $led = exec("gpio read 2");
    if ($led == 1)
    echo "led2 on";
        else
        echo "Wrong Settings";
    } else {
    exec("gpio write 2 0");
$led = exec("gpio read 2");//assure if command is exec.
        if ($led== 0)
        echo "led2 off";
        else
        echo "Wrong Settings";
          }
}

//LED3
if ( isset($_POST['led3']) ) {
                    if($_POST['led3'] == 1) {
                                        exec("gpio write 3 1");// pin 3 is W.Pi 2?
                                        $led = exec("gpio read 3");
                                        if ($led == 1)
                                                            echo "led3 on";
                                        else
                                                            echo "Wrong Settings";
                    } else {
                                        exec("gpio write 3 0");
                                        $led = exec("gpio read 3");
                                        if ($led== 0)
                                                            echo "led3 off";
                                        else
                                                            echo "Wrong Settings";
                    }
}

//LED4 NOT CONNECTED
if ( isset($_POST['led4']) ) {
                    echo "NOT";
}

//LED5 NOT CONNECTED
if ( isset($_POST['led5']) ) {
          echo "NOT";
}

//IN1 NOT CONNECTED
if(isset($_POST['in1'])) {
          echo "NOT";
}

if(isset($_POST['all'])) {
          $in1 = exec("gpio read 7");
          $led1 = exec("gpio read 0");
          $led2 = exec("gpio read 2");
          $led3 = exec("gpio read 3");
          $led4 = exec("gpio read 4");
          $led5 = exec("gpio read 5");
```

```
                        //IN 1 is stated as 0 since nothing is connected!
                        echo "state"."N"."."."$led1."."."$led2."."."$led3."."."$led4."."."$led5;
                }

}else{ //Post req. incomplete
                echo "Required field(s) is missing";
}
}//Connected to AP

?>
```